공학박사학위논문

Multi-Robot Environmental Learning and Target Tracking with Distributed Gaussian Process

분산 가우시안 과정을 통한 다중 로봇 환경 학습 및 목표물 추적 기법

2022년 **8**월

서울대학교 대학원

항공우주공학과

장 도 현

Multi-Robot Environmental Learning and Target Tracking with Distributed Gaussian Process

분산 가우시안 과정을 통한 다중 로봇 환경 학습 및 목표물 추적 기법

지도교수 김 현 진

이 논문을 공학박사 학위논문으로 제출함

2022년 **5**월

서울대학교 대학원

항공우주공학과

장 도 현

장도현의 공학박사 학위논문을 인준함

2022년 **6**월

Multi-Robot Environmental Learning and Target Tracking with Distributed Gaussian Process

A Dissertation

by

Jang, Dohyun

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2022

Multi-Robot Environmental Learning and Target Tracking with Distributed Gaussian Process

Jang, Dohyun

Department of Aerospace Engineering

Seoul National University

APPROVED:

Youdan Kim

Youdan Kim, Chair, Ph.D.

H. Jin Kim

H. Jin Kim, Ph.D.

Chan Gook Park

Chan Gook Park, Ph.D.

Woojin Kim

Woojin Kim, Ph.D.

Jaehyun Yoo

Jaehyun Yoo, Ph.D.

to my

FAMILY and HARIM

with love

Abstract

Multi-Robot Environmental Learning and Target Tracking with Distributed Gaussian Process

Jang, Dohyun
Department of Aerospace Engineering
The Graduate School
Seoul National University

This dissertation presents an investigation on distributed environment learning techniques in multi-robot systems and applies them to multi-target search and tracking problems. A multi-robot system has the advantages of reliability, efficiency, and scalability, facilitated by the cooperative work of multiple robots. Such a system can be easily applied to data-driven environmental learning. Data-driven environmental learning is a technique for obtaining comprehensive information on a region of interest by acquiring a large amount of sensor data from a specific region of interest. However, distributed-learning algorithms and collaborative algorithms are required to enable multiple robots to perform such tasks.

The first part of this dissertation focuses on the distributed environment learning algorithm. Given noisy sensor measurements obtained at the location of robots with no prior knowledge of the environmental map, Gaussian process regression can be an efficient solution for constructing a map that represents spatial information with confidence intervals. However, because the conventional Gaussian process algorithm operates in a centralized manner, processing information coming from multiple distributed sensors in real time is difficult. In this work, a multi-robot exploration algorithm is proposed, which deals with the following challenges: i) construction of a distributed environmental map using networked sensing platforms, ii) online learning using successive measurements suitable for a multi-robot team, and iii) active sensing and control for multi-agent coordination to determine the highest peak of an unknown environmental field. The effectiveness of the proposed algorithm is demonstrated through simulation and a topographic survey experiment with multiple unmanned aerial vehicles (UAVs). However, this technique lacks path planning in the cooperative search process, resulting in myopic behavior. Accordingly, the second part of this dissertation proposes a multi-robot informative path planning algorithm working in a fully distributed manner. This algorithm tackles the following challenges: i) online distributed learning of environmental map using multiple robots, ii) generation of safe and efficient exploration paths based on the learned map, and iii) maintenance of scalability with respect to the number of robots. Accordingly, the entire process is divided into two stages: environmental learning and path planning. Distributed algorithms are applied to each stage and combined through communication between adjacent robots. The learning algorithm uses a distributed Gaussian process, and the path planning algorithm uses a distributed Monte Carlo tree search. Therefore, a scalable system without a constraint on the number of robots is built. Simulation results demonstrate the performance and scalability of the proposed system. Moreover, a hardware experiment validates the utility of the proposed algorithm in a more realistic scenario.

Finally, the results of environmental learning can be applied to search and tracking problems using multi-robots. Deployment of multiple robots for target search and tracking has many practical applications; however, the challenge of planning for unknown or partially known targets remains difficult to address. With recent advances in deep learning, intelligent control techniques such as reinforcement learning have enabled agents with little to no prior knowledge to learn autonomously from environmental interactions. Such methods can address the exploration–exploitation tradeoff of planning for unknown targets in a data-driven manner, eliminating the reliance on heuristics—typical of traditional approaches—and streamlining the decision-making pipeline with end-to-end training. Accordingly, a multi-agent reinforcement learning technique with target map building based on a distributed Gaussian process is proposed. The distributed Gaussian process to encode belief over the target locations is leveraged to efficiently plan for unknown targets. Further, the performance and transferability of the trained policy is evaluated through simulation, and the method is demonstrated through hardware experiments on a swarm of micro UAVs.

Keywords: multi-robot system, environmental learning, informative path planning, deep reinforcement learning

Student Number: 2019-35974

Table of Contents

List of Tables

List of Figures

Abbreviations and Acronyms

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
AGL	Above Ground Level
CNN	Convolutional Neural Network
CPU	Central Processing Unit
D-UCB	Distributed-Upper Confidence Bound
DDPG	Deep Deterministic Policy Gradient
DGP	Distributed Gaussian Process
DP	Dynamic Programming
DRL	Deep Reinforcement Learning
FEM	Finite Element Method
GM-PHD	Gaussian Mixture-Probabilistic Hypothesis Density
GMM	Gaussian Mixture Model
GP	Gaussian Process
GPR	Gaussian Process Regression
IIR	Infinite Impulse Response
INS	Inertial Navigation System
KF	Kalman Filter
KL	Karhunen–Loève
LASER	Light Amplification by Simulated Emission of Radiation
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MARL	Multi-Agent Reinforcement Learning

MAS	Multi-Agent System
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
ML	Maximum Likelihood
MLP	Multi-Layer Perceptron
MRS	Multi-Robot System
NCS	Networked Control System
POMDP	Partially-Observable Markov Decision Process
RL	Reinforcement Learning
RMSE	Root Mean Square Error
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
RVI	Reduced Value Iteration
SE	Squared-Exponential
ToF	Time-of-Flight
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UCB	Upper Confidence Bound
WSN	Wireless Sensor Network

1 Introduction

In recent years, multi-agent systems (MASs)—as a means of solving complex and difficult problems— have received considerable attention in various academic fields, including mechanical engineering, aerospace engineering, and computer science [1–4]. A multi-agent system comprises autonomous entities known as agents, which are assigned individual tasks to achieve a common goal. Each agent can judge a situation and act on its own through interaction with a neighboring agent or its environment. This flexibility and scalability allowsallow MASs to solve various problems. When these agent entities are replaced by robotic agents, they are referred to as multi-robot systems (MRSs) [5, 6].

A multi-robot system differs from a system comprising virtual computer agents in that it has a physical location. Robots need to be moved to a suitable place occasionally, and changing positions is not only difficult in itself but also incurs time and cost. Moreover , based on a robot's position, a communication channel between robots is established. In particular, small UAVs or robots use communication modules that are relatively light and have low performance due to the limitation of payload and minimization of cost. This makes unrestricted information exchange between robots difficult, which prevents sophisticated perception of the environment and decision on the situation [7, 8]. To implement an algorithm that operates robustly even in such a limited communication network environment, studies on distributed systems are attracting attention.

An example is an application that realizes environmental awareness and learning through a wireless sensor network (WSN) [9]. Instead of installing wireless sensors in a fixed location, installing them on a moving robot can help implement a robotic sensor network system [10, 11]. By using the robotic sensing platform, acquiring continuous sensor data along the path of the robot is possible, rather than just learning the sensor data at a predetermined initial position. Therefore, even with fewer sensors, an environmental map for a larger area can be obtained at a higher resolution. However, if the environmental map needs to be learned in real time, all the sensor information cannot be utilized if communication with the central server is limited. In this case, a distributed algorithm that can exchange and fuse information only through communication with adjacent robots is required.

Path planning and cooperative algorithms are also required for autonomous movement of robots. A robot must move to its destination by avoiding obstacles and optimize the traveled path. The optimization criterion may be the minimum time or shortest distance, or it may be the maximum amount of information that can be obtained. The cooperative algorithm increases the task performance efficiency by considering each other's trajectories . In some cases, multiple robots may need to create a specific formation and maintain it while moving. This cooperative formation control problem has been addressed in various studies [12, 13].

The problem of considering the environmental learning and path planning simultaneously is the multi-robot informative path planning [14–16]. Various papers have previously addressed these issues. However, in most cases, a suitable distributed algorithm was not applied, so the scalability for the number of robots was not considered. Also, due to the low cooperation in the path creation process, it is not possible to efficiently utilize a large number of robots. In order to perform the efficient navigation task of multiple robots, a fully distributed algorithm must be applied to both the environmental learning process and the path planning process.

1.1 Contributions and Outline

The dissertation aims to tackle the problem of distributed environmental learning and target search and tracking in multi-robot sysetms. Because mobile robots use inherently light and weak communication modules, communication distance and bandwidth are limited. To overcome this harsh network environment, a distributed algorithm that works only on local communication is essential. Therefore, I apply the distributed Gaussian process to learning the unknown environment as shown in Chapter 3. Next, Chapter 4 proposes the informative path planning algorithm for multiple robots to obtain additional sensory data helpful to improve GP estimation quality. Extending these environmental learning problems, I have addressed the multi-target search and tracking problem as shown in Chapter 5. Using the signal intensity map learned by distributed GP, I propose a multi-agent deep reinforcement learning technique that can guide multiple agents to point to the targets' location. The contributions of this dissertation are summarized as follows.

Chapter 3: Multi-Robot Environmental Learning with Distributed Gaussian Process

- **Distributed Gaussian process**: I propose a distributed Gaussian process for the learning of unknown environment in a distributed manner. In this chapter, Karhunen–Loève (KL) kernel expansion and an average consensus algorithm are presented for cooperative environmental learning.
- **Succesive data fusion and active sensing**: I propose an environmental learning technique in which multiple robots fuse sensory data obtained from new locations in real time, and design a multi-robot guidance control technique to acquire additional sensory data.

Chapter 4: Informative Path Planning for Multi-Robot Systems

- **Informative path planning**: I propose a distributed search and path planning algorithm based on the amount of information. The confidence of the learned map obtained by the Gaussian process becomes a criterion for measuring the amount of information in the predicted trajectory.

- **Monte Carlo tree search**: A sampling-based Monte Carlo tree search technique is combined with the Gaussian process to compute an optimally predicted trajectory for the robot. A near-optimal prediction trajectory is obtained with fewer sampling times using the upper confidence bound (UCB) algorithm based on probability theory.

- **Distributed Monte Carlo tree search**: Robot agents share the predicted path obtained by Monte Carlo tree search with neighboring agents to influence each other's path regeneration. This method allows multiple robots to collaboratively perform exploration, taking into account collision avoidance and coordination.

Chapter 5: Target Search and Tracking with Reinforcement Learning

- **Multi-agent reinforcement learning**: I combine the distributed GP with deep reinforcement learning to tackle the problem of multi-agent target search and tracking. Even with only partial information about the environment, multiple robots can actively perform search missions by using past experiences.

- **Map-based MADDPG**: I design the network architecture accommodating high-dimensional inputs to incorporate the belief map with MADDPG. This method allows for a zero-shot transfer during deployment. The proposed technique enables multiple robots to actively search for and locate each target even in a distributed network environment.

2 Preliminaries

2.1 Multi-Robot Systems

In a multi-robot system, multiple robots work together with a common goal. As depicted in Fig. 2.1, the multi-robot system consists of multiple robots, environment, and a communication network. To interact with the environment, each robot senses environmental data and acts based on the environmental information.

We consider n robot agents exploring the environment, assuming all robots have the same dynamic model. In the multi-robot system we are dealing with, all robots obtain sensory data periodically from their location through on-board sensors and exchange them with neighboring robots. Each robot i takes the measurement y_k^i of an unknown environmental process $f(\cdot)$ in its position $\mathbf{x}_k^i \in \mathbb{X}_w (i = 1, 2, \cdots, n)$ at time k which has the following relationship:

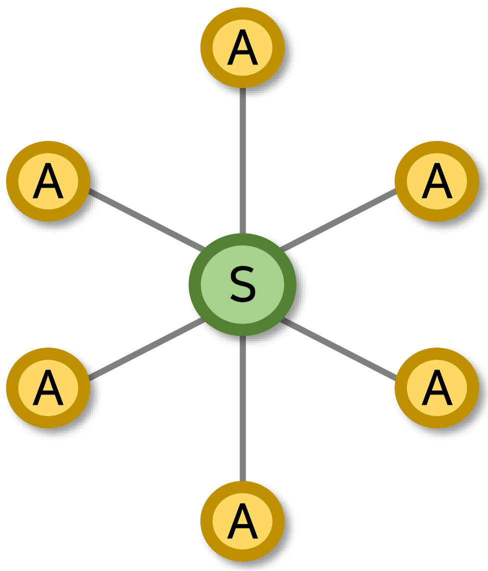
$$y_k^i = f(\mathbf{x}_k^i) + v_k^i, \tag{2.1}$$

where $\mathbb{X}_w$ is the domain of working space and the measurement of $f(\mathbf{x}_k^i)$ is corrupted by the additive white Gaussian noise $v_k^i \sim \mathcal{N}(0, \sigma_v^2)$. The unknown environmental process $f(\cdot)$ will be estimated by Gaussian process regression algorithm.

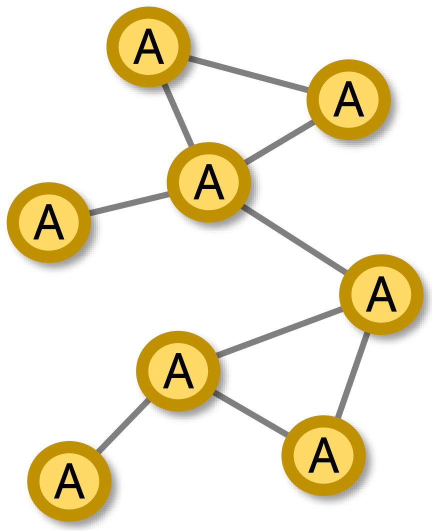
Figure 2.1: Overview of multi-robot systems.

During exploration, robots estimate environmental process by using both measured and shared data obtained from neighbors. To share measured data with other robots, all robots utilize the communication network. Let us briefly distinguish the network characteristics for the two main types of networks: centralized and distributed networks shown in Fig. 2.3. In the centralized network, all the robots send measurement data to a central server, and the global estimate obtained at the server is sent back to all the robots. Therefore, this method consumes much energy for communication and is vulnerable to bottlenecks and malfunction of the central server. On the other hand, the distributed network does not have the central server. Because the distributed schemes operate only through local communication with neighbor robots, it does not have constraints on network size or link failure risks [17–19]. Therefore, it is appropriate to be applied to a large system with many deployed robots.

The distributed network of n robots is defined based on the graph theory. Let $\mathcal{G}(k) = \{\mathcal{V}, \mathcal{E}(k)\}$ be an undirected graph of order n with the non-empty set of nodes $\mathcal{V} = \{v_i | \ i \in \mathcal{N}\}$ and the set



(a) Centralized network



(b) Distributed network

Figure 2.2: Two types of networks. Symbol 'A' means a robot agent and 'S' means a central server. Solid lines are peer-to-peer communication channels.

---

of edges $\mathcal{E}(k) = \{(v_i, v_j)|\ i, j \in \mathcal{N}, \|\mathbf{x}_k^i - \mathbf{x}_k^j\| < d_{comm}\}$ at time k, where $\mathcal{N} = \{1, 2, \cdots, n\}$ is the index set of nodes. (v_i, v_j) is the communication channel from the robot i to j, and d_{comm} is the communication range of the robot. We define a set of neighbors for robot i as $\mathcal{N}_k^i = \{j|\ (v_j, v_i) \in \mathcal{E}(k), j \in \mathcal{N}/i\}$, and we also define $\mathcal{N}^{i+}$ as $\mathcal{N}^i \cup \{i\}$. For arbitrary variable A, A^{i+} means $\{A^j\}_{j \in \mathcal{N}^{i+}}$, and A^{i-} means $\{A^j\}_{j \in \mathcal{N}^i}$ for brevity.

2.2 Gaussian Process

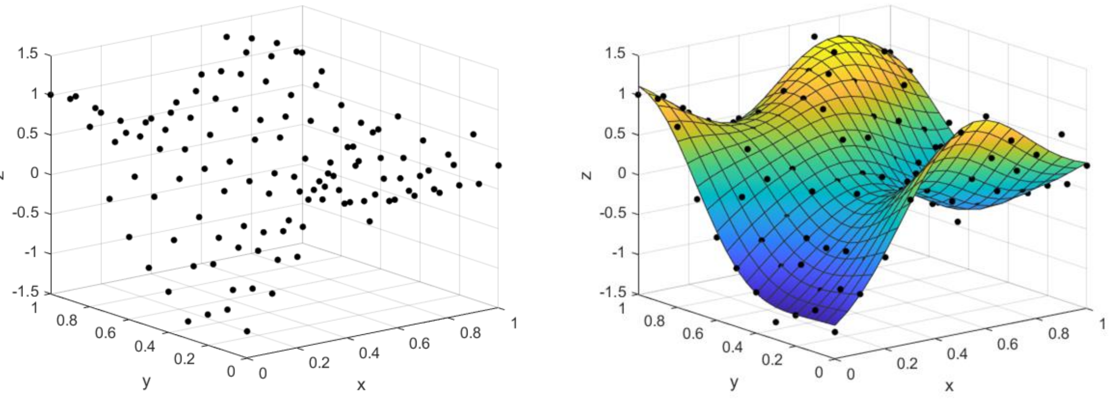
Gaussian process is a data-driven non-parametric learning method designed to solve regression and probabilistic classification problems, taking into account joint Gaussian probability distribution between the sampled dataset [20]. This method can provide the probabilistic prediction over the set $\mathbb{X}_w$ so that it can compute empirical confidence intervals. In (2.1), the unknown process model $f(\cdot)$ is assumed to follow a zero-mean Gaussian process as

$$f(\cdot) \sim \mathcal{GP}(0, \kappa(\mathbf{x}', \mathbf{x}'')), \tag{2.2}$$

where $\kappa(\mathbf{x}', \mathbf{x}'')$ is a *kernel* or *covariance function* for positions $\mathbf{x}', \mathbf{x}'' \in \mathbb{X}_w$. The covariance function is the crucial element in Gaussian process, as it defines the similarity between data points to lean unknown process. There are a number of common covariance functions (e.g., squared exponential, Ornstein-Uhlenbeck, and Matérn), and different covariance functions make different function estimations. In this work, we choose the original *squared exponential* (SE) kernel, which is defined as

$$\kappa(\mathbf{x}', \mathbf{x}'') = \sigma_s^2 \exp(-\frac{1}{2}(\mathbf{x}' - \mathbf{x}'')^\top \Sigma_l^{-1} (\mathbf{x}' - \mathbf{x}'')), \tag{2.3}$$

where σ_s^2 is the signal variance of $f(\cdot)$, and Σ_l is the length scale. The hyper parameters σ_s^2 and Σ_l can be determined by maximizaing the marginal likelihood [20].



(a) Sampled dataset (b) Gaussian posterior process

Figure 2.3: Gaussian process regression.

Formally, let $D_k^i = \{(\mathbf{x}_t^i, y_t^i)\}_{t \in M_k}$ be the training dataset sampled by the robot i, where t is the sampling time. M_k is the set of sampling time indices up to time k. With the dataset D_k^i of size m_k^i, we can simply define the input data matrix as $\mathbf{X}_k^i = [\bar{\mathbf{x}}_1^i, \cdots, \bar{\mathbf{x}}_{m_k^i}^i]^\top \in \mathbb{R}^{m_k^i \times 3}$ and the output data vector as $\mathbf{y}_k^i = [\bar{y}_1^i, \cdots, \bar{y}_{m_k^i}^i]^\top \in \mathbb{R}^{m_k^i \times 1}$. According to the test point $\mathbf{x} \in \mathbb{X}_w$, the posterior distribution over $f(\mathbf{x})$ by robot i is derived as follows:

$$p(f(\mathbf{x})|\mathbf{X}_k^i, \mathbf{y}_k^i, \mathbf{x}) \sim \mathcal{N}(\bar{f}^i(\mathbf{x}), \Sigma^i(\mathbf{x})), \tag{2.4}$$

where

$$\bar{f}^i(\mathbf{x}) = \mathbf{k}^\top(\mathbf{X}_k^i, \mathbf{x})(\mathbf{K}(\mathbf{X}_k^i, \mathbf{X}_k^i) + \sigma_v^2 I)^{-1}\mathbf{y}_k^i, \tag{2.5a}$$

$$\begin{aligned}\Sigma^i(\mathbf{x}) = {} & \kappa(\mathbf{x}, \mathbf{x}) \\ & - \mathbf{k}^\top(\mathbf{X}_k^i, \mathbf{x})(\mathbf{K}(\mathbf{X}_k^i, \mathbf{X}_k^i) + \sigma_v^2 I)^{-1}\mathbf{k}(\mathbf{X}_k^i, \mathbf{x}).\end{aligned} \tag{2.5b}$$

$\mathbf{K}(\mathbf{X}_k^i, \mathbf{X}_k^i)$ is the $m_k^i \times m_k^i$ kernel matrix whose (u, v)-th element is $\kappa(\bar{\mathbf{x}}_u^i, \bar{\mathbf{x}}_v^i)$ for $\bar{\mathbf{x}}_u^i, \bar{\mathbf{x}}_v^i \in \mathbf{X}_k^i$. $\mathbf{k}(\mathbf{X}_k^i, \mathbf{x})$ is the $m_k^i \times 1$ column vector that is also obtained in the same way.

2.3 Karhunen–Loève (KL) Kernel Expansion

Let the usual GP consider n robots. We can simply define the input data matrix for n robots as $\mathbf{X}_k = [\mathbf{X}_k^{1\top}, \cdots, \mathbf{X}_k^{n\top}]^\top \in \mathbb{R}^{mn\times 3}$. For simplicity, it is assumed that m_k^i's are same for all robots, and we omit the subscript k, so $m_k^i = m$ hereafter. With the matrix $\mathbf{X}_k$, the usual GP requires all the sampled data $\mathbf{X}_k$ and inversion of $K(\mathbf{X}_k, \mathbf{X}_k)$ with $O((mn)^3)$ operations. These requirements are impractical when peer-to-peer communication is only used, and the computational burden also increases depending on the data size. For this reason, a new kernel method is needed. The kernel (2.3) can be expanded in terms of eigenfunctions ϕ_e and corresponding eigenvalues λ_e as follows [21]:

$$\kappa(\mathbf{x}', \mathbf{x}'') = \sum_{e=1}^{+\infty} \lambda_e \phi_e(\mathbf{x}')\phi_e(\mathbf{x}''), \tag{2.6}$$

where $\lambda_e\phi_e(\mathbf{x}') = \int_{\mathbb{X}_w} \kappa(\mathbf{x}', \mathbf{x}'')\phi_e(\mathbf{x}'')d\mu(\mathbf{x}'')$. It is difficult to derive the kernel eigenfunctions in a closed-form, but the SE kernel expansion has already been obtained via Hermite polynomials, as mentioned in [22]. Then, the process model f for the position $\mathbf{x} \in \mathbb{X}_w$ is expanded as

$$\begin{aligned} f(\mathbf{x}) &= \sum_{e=1}^{E} a_e\phi_e(\mathbf{x}) + \sum_{e=E+1}^{+\infty} a_e\phi_e(\mathbf{x}) \\ &= f_E(\mathbf{x}) + \sum_{e=E+1}^{+\infty} a_e\phi_e(\mathbf{x}),\end{aligned} \tag{2.7}$$

where $a_e \sim \mathcal{N}(0, \lambda_e)$ for $e = 1, 2, \cdots, \infty$. $f_E(\mathbf{x})$ is the E-dimensional model of $f(\mathbf{x})$ where E is a constant design parameter. This parameter can be tuned by the SURE strategies [23]. As shown in [22], the optimal E-dimensional models can be obtained by a convex combination of the first E-kernel eigenfunctions as the size of sampled dataset increases to infinity.

2.4 Average Consensus Algorithm

Each agent can receive data only from neighbor agents due to the limitations of the communication network. Against the communication constraint, the consensus algorithm enables multiple agents to operate on the same protocol so that each local estimate converges to the global estimate. More details for obtaining a global estimate are addressed in Section 3.2.

We define the following dynamics with the state matrix $\{X_i\}_{i=1}^N$ and the consensus protocol $\{U_i\}_{i=1}^N$:

$$X_i((k+1)T) = X_i(kT) + U_i(kT). \tag{2.2}$$

where $k \in \mathbb{Z}_{\geq 0}$, and T is the sampling time. We replace all "(kT)" by "(k)" for brevity. The *average consensus* for these N state matrices is satisfied when the following two conditions are met [24]:

$$\lim_{k\to\infty} ||X_i(k) - X_j(k)|| = 0, \ \ \forall i, j \in \mathcal{N} \tag{2.3}$$

and

$$\sum_{i\in\mathcal{N}} X_i(k) = \bar{X}(k_0), \ \ \forall k \in (k_0, \infty). \tag{2.4}$$

The variable $\bar{X}(k)$ is the average matrix of all state matrices X_i at k. If only (2.3) is satisfied, *consensus* is achieved.

When the graph $\mathcal{G}(k)$ is connected, the protocol for the system (2.2) to achieve the average consensus is as follows [25]:

$$U_i(k) = -\frac{1}{|\mathcal{N}_i(k)|} \sum_{j\in\mathcal{N}_i(k)} \gamma(X_i(k) - X_j(k)) \tag{2.5}$$

for agent $i \in \mathcal{N}$. $|\mathcal{N}_i(k)|$ is the cardinality of $\mathcal{N}_i(k)$ and $\gamma \in \mathbb{R}$ is the parameter for convergence rate. We will show how the multi-agent team performs distributed Gaussian process regression in the networked control systems using this average consensus protocol.

3
Multi-robot Environmental Learning with Distributed Gaussian Process

This chapter deals with multi-robot cooperative exploration and environmental learning techniques. The given goal is to map the environmental process of the field of interest through a Gaussian process. At this time, it is necessary to use a distributed algorithm for real-time learning through the sensor network of multiple robots. In this chapter, I introduce distributed Gaussian process to enable distributed environmental learning. In addition, by using the mobility of multiple robots, it is possible to acquire more sensor data not only in the initial position of robots but also in a wider area. The acquired data is fused to update the learned environment model in real time.

3.1 Introduction

Multi-robot systems can be better specialized than a single robot for missions in large areas such as crop monitoring, observing climate changes, and terrain surveying [26]. These missions can be viewed as environmental process estimation problems in which each robot measures spatiotemporal data on its own. As such, techniques for multiple robots to move around and acquire local information for constructing a global environmental map are referred to as robotic sensor networks [11].

In robotic sensor networks, robots move to optimal locations where most valuable data can be obtained, and these data are used to generate an environmental map. I need to learn an environmental model from the collected data and estimate the information gain to decide which location to investigate further.

Another challenge in multi-robot systems is the limitation of network resources such as communication distance, transmission throughput, and channel bandwidth [27, 28]. Given these issues, network control systems (NCS) have been developed to facilitate the cooperative work by using mutually agreed protocols for data communication of a large number of robots [24, 25].

In this chapter, I present a distributed multi-robot exploration method for joint sensing and learning of an unknown environmental process with the following challenges: i) distributed learning for an environmental process and communication protocol in networked control systems, ii) online update of an environmental process model with newly acquired sensory data for multi-robot systems, and iii) multi-robot coordination for improving the process estimation performance.

3.1.1 Related work

Gaussian process (GP) regression [20] can be used to construct an environmental process from data [29, 30]. It derives a spatial relationship between sampled data by using a kernel function and performs Bayesian inference for prediction at an unknown location. An online version of GP [31], which is extended from typical batch learning, would be more suitable for real-time robot learning scenarios [32].

However, most online GP approaches have been designed for centralized systems, making it difficult to apply them for a multi-robot NCS subject to the limitation of network resources (e.g., transmission power, throughput, or channel bandwidth). Even though relay communication might be a solution for large-scale data transition [33, 34], it takes too much time and bandwidth

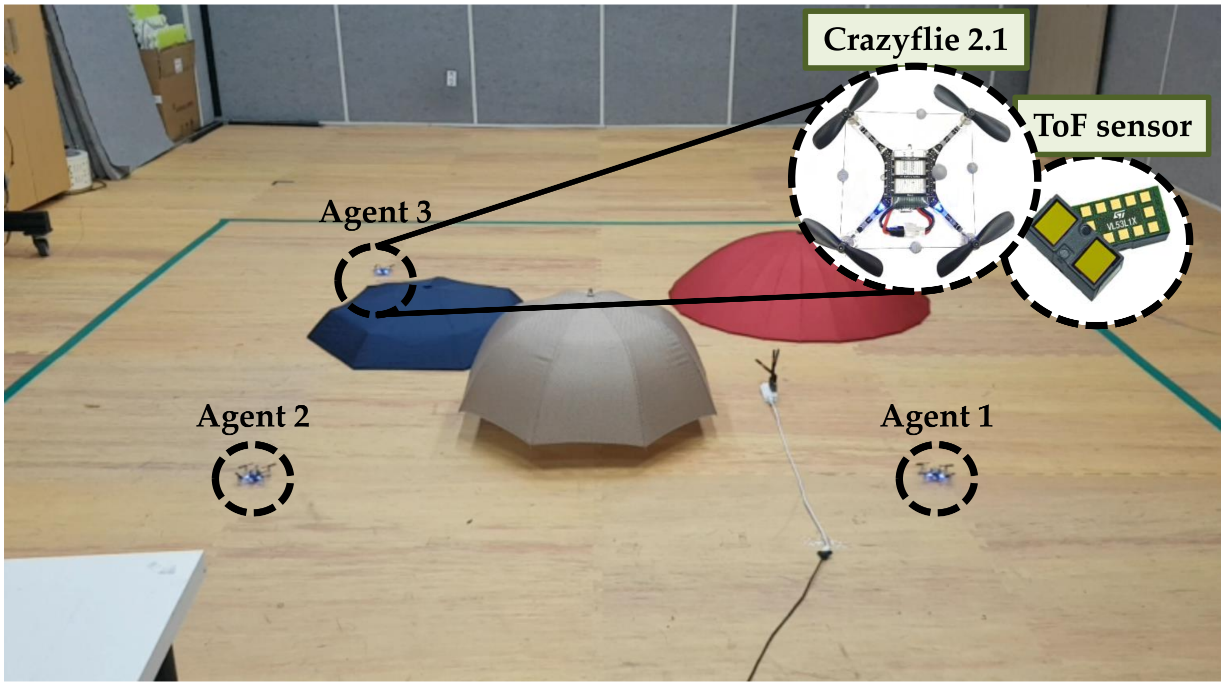


Figure 3.1: Topographic survey experiments with multiple UAVs.

to adapt rapidly changing environment in multi-robot NCS.

In [23], distributed GP regression is introduced for multi-agent systems, in which a finite-dimensional GP estimator is designed by using Karhunen–Loève (KL) kernel expansion. It is more flexible on a network scale because only a connected graph is needed where each node is connected to the other nodes through a sequence of edges (i.e. path), not a centralized network where all agents are directly connected to a central server. In contrast to the decentralized GP's local estimate, the distributed GP yields a global estimate as if using the sensory data of all agents by exchanging estimator information with neighbors. However, this study considers sensor networks only, not moving robot platforms. Thus, the sensor position is fixed, and data cannot be obtained where the sensor is not installed. The main difference with my work lies in the fact that I can update the GP estimate using newly measured sensory data from a multi-robot team.

I am interested in robotic sensor networks that could enable active sensing and constructing the environmental model using mobile robots. [35] represents informative planning for autonomous robot with online sparse GP, which takes advantage of a subset of data. In contrast to my work, it considered only single-agent exploration. In [36], a multi-robot sensor coverage problem is addressed with a mixture of the Gaussian process. These papers employ multi-robot systems and GP but concentrate only on the optimal sensor placement objectives without consideration of collision avoidance.

The idea of combining GP and multi-robot systems has already been explored in several works. [37] presents a decentralized multi-agent exploration with online GP, focusing on multi-agent coordination and physical process construction. In [38], Gaussian radial basis functions and multi-agent systems are utilized to discover peak, considering the communication distance. In contrast to my study, both [37] and [38] only address the local-estimate-based exploration, not the global-estimate-based. With local estimates for multi-robot control, each agent would follow an inefficient path or converge to the local peaks. [39] developed multi-UAS planning with a decentralized GP fusion algorithm, which takes a different approach from my works in implementing a distributed GP. It has an advantage of communication efficiency; however, this technique does not guarantee the global estimate's performance.

3.1.2 Problem Statement

Multiple robots (e.g., ground vehicles or UAVs) explore an unknown environmental process in 3-D space with onboard sensors. Each agent estimates environmental information across an unknown area that has not yet been explored by using both own measurements and shared data received from neighbors. The robot determines its next location to move based on the prediction by the distributed GP, where a search for a place of high uncertainty is prioritized. I consider the following setting:

• The exploration area is finite, and all agents know the boundary of the exploration area. The environmental process is time-invariant.

• Each agent has its localization and navigation capabilities in the global coordinate system.

• Each agent can only communicate directly with agents within the communication range. Communication between the two agents is bidirectional.

3.1.3 Contribution

This chapter focuses on three main contributions to multi-robot exploration in the networked control system.

• I present a distributed GP algorithm with global mean and variance estimation through Karhunen–Loève expansion and an average consensus protocol [23].

• I expand the distributed GP algorithm to be suitable for continuous data collection to enable online GP learning for mobile robots.

• To find the location of the peak value of a scalar function, which has been considered in [38, 40], I propose the maximum variance exploration and maximum mean exploitation for individual agents. For safe and efficient exploration, I apply a collision avoidance and coordination algorithm.

I perform a multi-robot exploration simulation in a virtual environment and conduct a topographic survey experiment using multiple unmanned aerial vehicles (UAVs) with a laser rangefinder. The outline of this chapter is as follows. Sec. 2 summarizes distributed GP. Sec. 3 combines distributed GP and multi-robot coordination. Simulation and real-time experiments are presented in Sec. 4. Finally, Sec. 5 concludes the chapter.

3.2 Distributed Gaussian Process

3.2.1 Multi-Agent Distributed Gaussian Process

I apply E-dimensional approximation to the GP estimator in (2.5) to derive the estimation of $f_E(\mathbf{x})$. According to E-dimensional approximation, the kernel function (2.6) can be described as $\kappa(\mathbf{x}', \mathbf{x}'') \approx \sum_{e=1}^{E} \lambda_e \phi_e(\mathbf{x}')\phi_e(\mathbf{x}'')$. For the input data matrix $\mathbf{X}_k$, kernel matrices included in (2.5) are defined by

$$\mathbf{K}(\mathbf{X}_k, \mathbf{X}_k) = G\Lambda_E G^\top, \tag{3.1a}$$

$$\mathbf{k}(\mathbf{X}_k, \mathbf{x}) = G\Lambda_E \Phi(\mathbf{x}), \tag{3.1b}$$

where $\Phi(\mathbf{x}) := \Big[\phi_1(\mathbf{x}), \cdots, \phi_E(\mathbf{x})\Big]^\top$ and $G := \Big[\Phi(\bar{\mathbf{x}}_1^1) \cdots \Phi(\bar{\mathbf{x}}_m^1), \cdots, \Phi(\bar{\mathbf{x}}_1^n) \cdots \Phi(\bar{\mathbf{x}}_m^n)\Big]^\top$. Λ_E is the diagonal matrix of kernel eigenvalues.

With (2.5a) and (3.1a), the E-dimensional estimator for GP is expressed as follows [23]:

$$\bar{f}_E(\mathbf{x}) \;:=\; \Phi^\top(\mathbf{x}) H_E \mathbf{y}, \tag{3.2}$$

where

$$H_E := \left(\frac{G^\top G}{mn} + \frac{\sigma_v^2}{mn}\Lambda_E^{-1}\right)^{-1} \frac{G^\top}{mn}. \tag{3.3}$$

Because each agent cannot obtain G and $\mathbf{y}$ in (3.2) without a fully connected network, I decompose the associated terms included in (3.3) as follows:

$$\frac{G^\top G}{mn} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{t=1}^{m}\Phi(\bar{\mathbf{x}}_t^i)\Phi^\top(\bar{\mathbf{x}}_t^i) = \frac{1}{n}\sum_{i=1}^{n}\alpha_m^i, \tag{3.4a}$$

$$\frac{G^\top \mathbf{y}}{mn} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{t=1}^{m}\Phi(\bar{\mathbf{x}}_t^i)\bar{y}_t^i = \frac{1}{n}\sum_{i=1}^{n}\beta_m^i, \tag{3.4b}$$

where $\alpha_m^i := \sum_{t=1}^m \Phi(\bar{\mathbf{x}}_t^i)\Phi^\top(\bar{\mathbf{x}}_t^i)/m$ and $\beta_m^i := \sum_{t=1}^m \Phi(\mathbf{x}_t^i)\bar{y}_t^i/m$ are *GP states* after the m-th sensor measurements. Now (3.2) is reformulated in the following distributed form:

$$\bar{f}_E^i(\mathbf{x}) \;:=\; \Phi^\top(\mathbf{x})\left(\alpha_m^i + \frac{\sigma_v^2}{mn}\Lambda_E^{-1}\right)^{-1}\beta_m^i. \tag{3.5}$$

As the results of average consensus protocol in Section 2.4, (3.5) converges to (3.2) after iterative communication. Similarly, the distributed form of $\Sigma(\mathbf{x})$ in (2.5b) is expressed as

$$\Sigma_E(\mathbf{x}) \;:=\; \kappa(\mathbf{x},\mathbf{x}) - \Phi^\top(\mathbf{x}) H_E G \Lambda_E \Phi(\mathbf{x}), \tag{3.6}$$

$$\begin{aligned}\Sigma_E^i(\mathbf{x}) := \kappa(\mathbf{x},\mathbf{x}) - \Phi^\top(\mathbf{x})\left(\alpha_m^i + \frac{\sigma_v^2}{mn}\Lambda_E^{-1}\right)^{-1} \\ \times \alpha_m^i \Lambda_E \Phi(\mathbf{x}).\end{aligned} \tag{3.7}$$

When I compare (2.5) with (3.5) and (3.7), the computational complexity of the distributed algorithm is $O(E^3)$, whereas that of the centralized algorithm is $O((mn)^3)$ due to the matrix inversion [23]. Therefore, the distributed algorithm is more scalable since $E \ll mn$ in general.

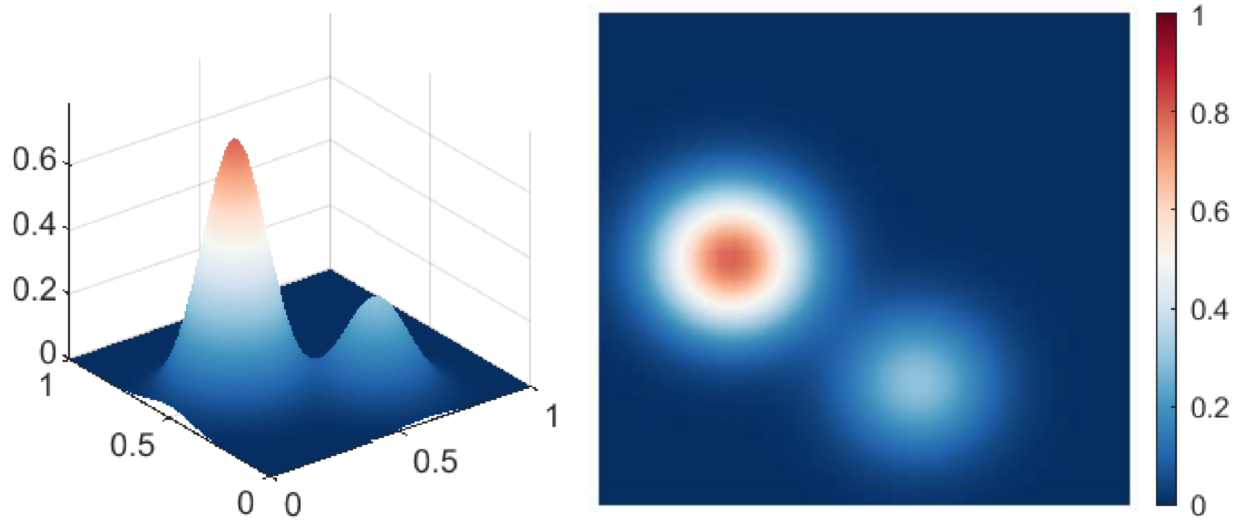


Figure 3.2: Bi-modal environmental phenomenon for simulation. (left) 3D perspective view. (right) top-down view.

Fig. 3.3 shows distributed GP example using 64 stationary agents for unknown environmental model in Fig. 3.2. The overall uncertainty for the workspace decreases over time, and the GP mean estimate converges to the original model.

Fig. 3.4 shows the analysis of convergence rate according to the network. In general, the greater the number of neighbors connected to each agent, the faster the convergence speed of the consensus algorithm. However, when excessive communication channels are connected, the network overload increases, and bottleneck occurs. If communication channels are fully-connected, it is a centralized network.

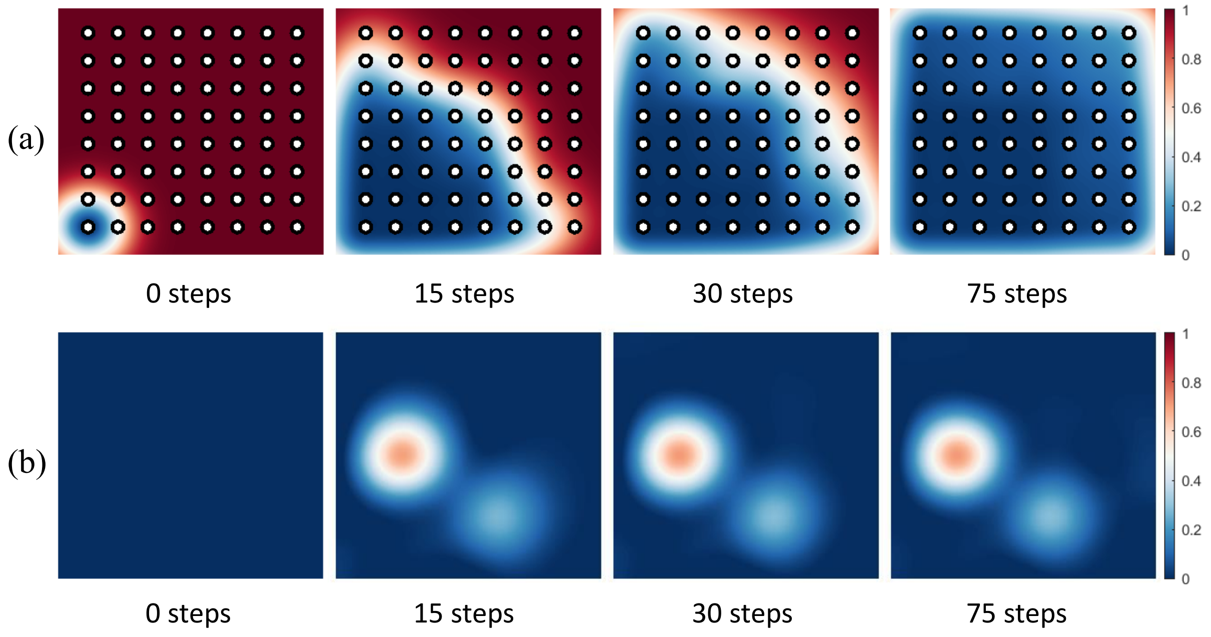


Figure 3.3: The process of the environmental model construction performed by 64 stationary sensors with distributed Gaussian process regression. It shows (a) the uncertainty propagation and (b) the change of the GP estimate with time from $k = 0$ to $k = 75$ in order from the left figure. The environmental model is presented in Fig 3.2. The communication range is equal to the distance between two adjacent agents. All results are obtained by agent $\#1$ located at the bottom-left corner. In addition to the results of agent $\#1$, the results of all the other agents converge to the same.

3.2.2 Online Information Fusion by Moving Agents

If the $(m+1)$-th new training dataset $\{(\bar{\mathbf{x}}^i_{m+1}, \bar{y}^i_{m+1})\}^n_{i=1}$ are obtained, $\{\alpha^i_m\}^n_{i=1}$ and $\{\beta^i_m\}^n_{i=1}$ have to be discarded to include new data so that the consensus process must be restarted from scratch. To avoid repeated restarts and keep the continuity of environmental estimate, I introduce the online information fusion algorithm.

Let me assume that the sensor measurement frequencies of all agents are same for convenience. The update rule of α^i_m and β^i_m is defined as follows:

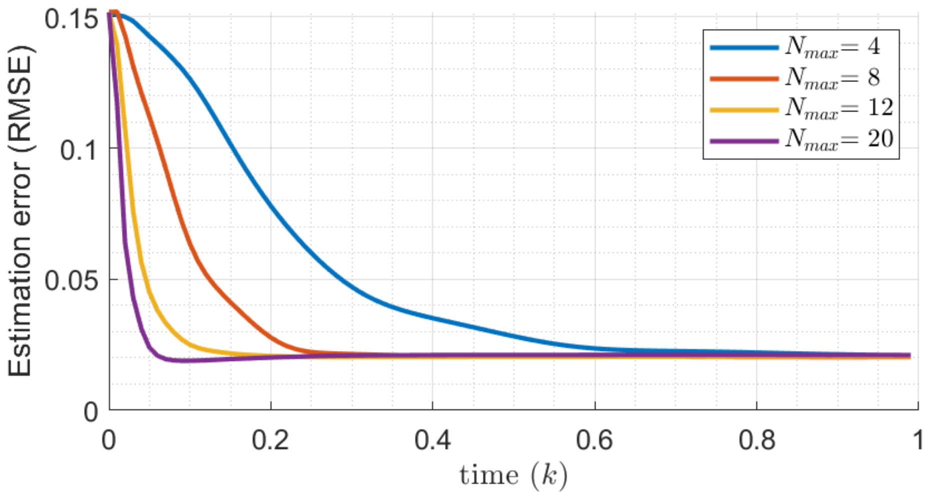


Figure 3.4: Convergence rate according to the maximum number of neighbors.

$$\begin{aligned} \alpha^i_{m+1} &= (1-r)\alpha^i_m + r\Phi(\bar{\mathbf{x}}^i_{m+1})\Phi^\top(\bar{\mathbf{x}}^i_{m+1}), \\ \beta^i_{m+1} &= (1-r)\beta^i_m + r\Phi(\bar{\mathbf{x}}^i_{m+1})y^i_{m+1}, \end{aligned} \tag{3.8}$$

where $\alpha^i_0 = 0$ and $\beta^i_0 = 0$. This rule is an infinite impulse response (IIR) filter. If $r = (m+1)^{-1}$, The update rule reflects all dataset equally, so it is suitable for static environmental learning. If $r > (m+1)^{-1}$, this rule reflects more of the recent data, so it is suitable for dynamic environmental learning. Fig. 3.5 shows distributed GP example using 64 stationary agents to estimate the dynamic environment. The online information fusion algorithm enables learning about dynamic environments. Because the propagation of information takes time, it is not possible to predict the ground truth in real time. However, it keeps up with the changing environment through iterative information exchange.

Theorem 1. *Using the average consensus protocol and update rules in* (3.8)*, new data are successively fused with the existing* $\{\alpha^i_m\}^n_{i=1}$ *and* $\{\beta^i_m\}^n_{i=1}$*, so that* $\{\alpha^i_{m+1}\}^n_{i=1}$ *and* $\{\beta^i_{m+1}\}^n_{i=1}$ *converge towards* (3.4a) *and* (3.4b)*, respectively, in a distributed manner.*

Proof. See The Appendix. □

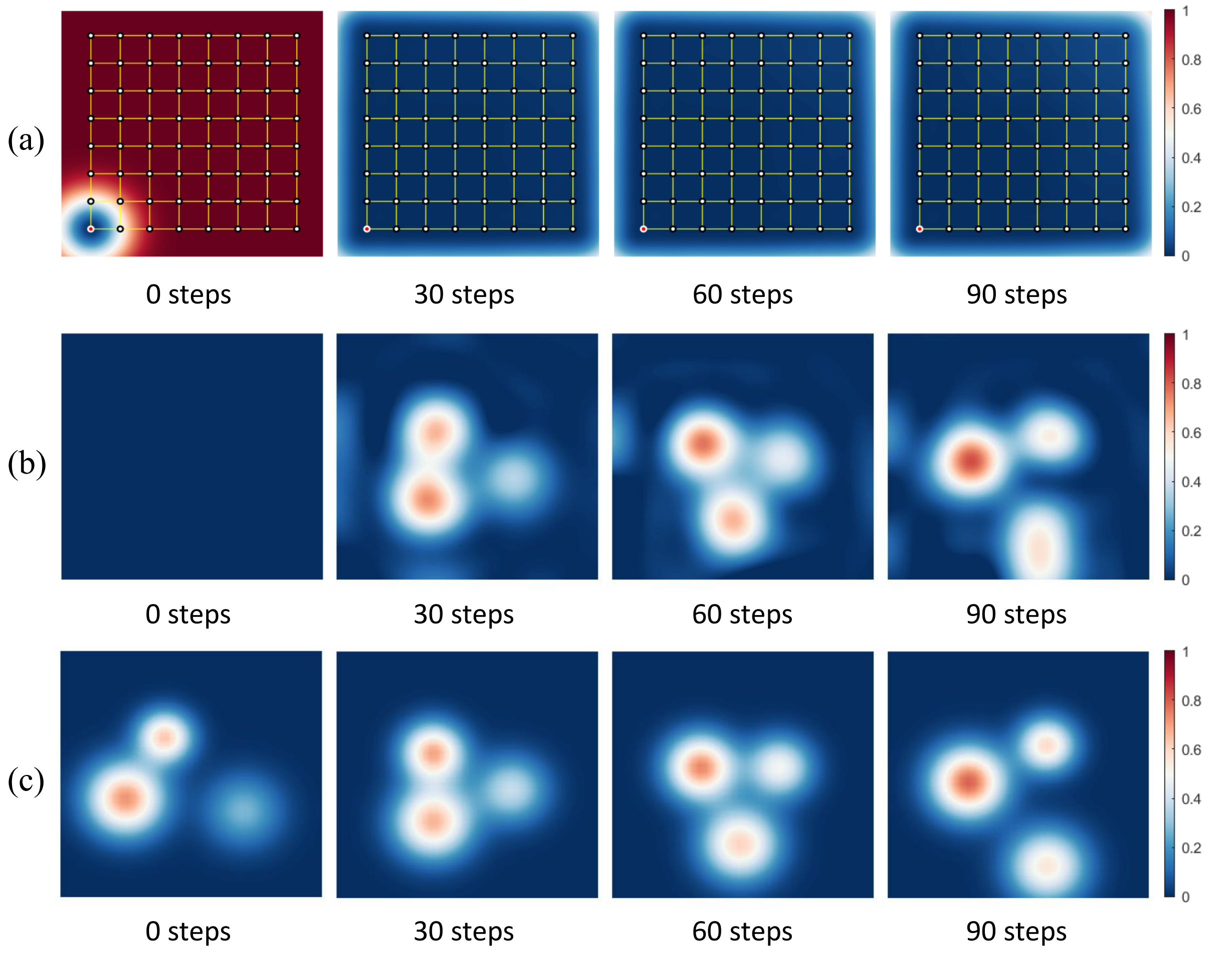


Figure 3.5: The process of the dynamic environmental model construction performed by 64 stationary sensors with distributed Gaussian process regression. It shows (a) the uncertainty propagation, (b) the change of the GP estimate, and (c) the change of dynamic environment with time from $k = 0$ to $k = 90$ in order from the left figure. Yellow lines are communication links between agents. All results are obtained by agent $\#1$ located at the bottom-left corner. With the online information fusion algorithm, dynamic environment can be estimated for all agents.

Algorithm 3.1: Multi-Robot Exploration for agent #*i*

```
1:  $\alpha_i(k < k_0) \leftarrow 0, \beta_i(k < k_0) \leftarrow 0$
2:  $k \leftarrow k_0, m \leftarrow 1$
3:  while True do
4:      /*Sensing*/
5:      if mod(k,SensingPeriod) $= 0$ then
6:          $y_i(k) \leftarrow f(\mathbf{x}_i(k)) + \nu$
7:          $\alpha_i(k) \leftarrow \frac{m-1}{m}\alpha_i(k-1) + \frac{1}{m}\Phi(\mathbf{x}_i(k))\Phi^T(\mathbf{x}_i(k))$
8:          $\beta_i(k) \leftarrow \frac{m-1}{m}\beta_i(k-1) + \frac{1}{m}\Phi(\mathbf{x}_i(k))y_i(k)$
9:          $m \leftarrow m + 1$
10:     end
11:     /*Communication*/
12:     $\Delta\alpha_i(k) \leftarrow -\sum_{j\in\mathcal{N}_i}\gamma(\alpha_i(k) - \alpha_j(k))$
13:     $\Delta\beta_i(k) \leftarrow -\sum_{j\in\mathcal{N}_i}\gamma(\beta_i(k) - \beta_j(k))$
14:     $\alpha_i(k+1) \leftarrow \alpha_i(k) + \Delta\alpha_i(k)$
15:     $\beta_i(k+1) \leftarrow \beta_i(k) + \Delta\beta_i(k)$
16:     /*Multi-Robot Coordination $(Algorithm 3.2)$*/
17:     $\mathbf{x}_i(k+1) \leftarrow \pi(\mathbf{x}_i(k), \alpha_i(k), \beta_i(k))$
18:     $k \leftarrow k + 1$
19: end
```

3.3 Multi-Agent Active Sensing and Control

3.3.1 Exploration and Exploitation

In the previous section, I introduced the distributed GP to construct an unknown environmental model by letting multiple agents explore. In this section, I establish a multi-robot exploration strategy to improve GP performance with additional data collection and exploitation strategies to improve problems when focusing only on exploration.

To make up for the lack of information in the initial location, each robot would move in the direction of the region where the uncertainty of GP estimation is high (exploration). Each robot sets a searching area like (3.9) based on its current location $\mathbf{x}_i$ and calculates the distributed GP variance (3.7).

$$X_c = \{\mathbf{x}_c = l\angle\theta_c + \mathbf{x}_i |\ \theta_c \in [0, 2\pi)\} \tag{3.9}$$

where l is the search range. If the uncertainty around the robot is lower than a parameter $\sigma^2_{\text{threshold}}$, the search for nearby areas of the robot is not likely to be effective for improving the overall amount of information. In this case, the robot should broaden the search area and examine the GP uncertainty over a wider area (line 15 of Algorithm 3.2). If there is a region where the uncertainty exceeds $\sigma^2_{\text{threshold}}$, the robot moves toward the direction decided by the lines 12-13,23-24,29 of Algorithm 3.2. If the robot cannot find an area having the uncertainty above a certain threshold after trying expanding the search area, it enters the exploitation process (lines 17-20,27-28 of Algorithm 3.2).

The goal of the robots is to find the highest value for the environmental process. GP increases the uncertainty as the test location moves away from the data acquisition location. Therefore, exploration techniques that chase only uncertainty have a limit in the detailed description of the peak. To better describe the model around the peak, the exploitation step is needed [41]. Considering the confidence region of GP, exploration and exploitation can be performed at the same time by adding the mean value estimate and variance estimate, such as (3.10) (lines 26-27

Algorithm 3.2: Multi-Robot Control Policy π for agent #*i*

```
    input : $\mathbf{x}_i, \alpha_i, \beta_i$
    output : $\mathbf{x}_{i,next}$
 1: $l \leftarrow l_{init}$
 2: $ExplorationFlag \leftarrow True$
 3: $\Theta_{init} \leftarrow [0, 2\pi)$
 4: while $True$ do
 5:   /*Boundary condition*/
 6:   $\Theta_{bc} \leftarrow \{\theta_{bc} \in \Theta_{init} | \, (l\angle\theta_{bc} + \mathbf{x}_i) \in X_{map}\}$
 7:   /*Collision Avoidance condition*/
 8:   $\Theta_{ca} \leftarrow \{\theta_{ca} \in \Theta_{init} | \, ||(l_{init}\angle\theta_{ca} + \mathbf{x}_i) - \mathbf{x}_j|| > l_{ca},$
 9:          $\forall j \in \mathcal{N}_i\}$
10:   /*Motion Candidates*/
11:   $\Theta_c \leftarrow \Theta_{bc} \cap \Theta_{ca}$
12:   if $\max_{\theta_c \in \Theta_c} \Sigma_{E,i}(l\angle\theta_c + \mathbf{x}_i) > \sigma^2_{\text{threshold}}$ then
13:     break
14:   else
15:     $l \leftarrow l + l_{init}$
16:     if $l > l_{\text{threshold}}$ then
17:       $ExplorationFlag \leftarrow False$
18:       break
19:     end
20:   end
21: end
22: if $ExplorationFlag$ then
23:   /*Exploration*/
24:   $\theta_{d,i} \leftarrow \arg\max_{\theta_c \in \Theta_c}[w_i(l\angle\theta_c + \mathbf{x}_i) \times \Sigma_{E,i}(l\angle\theta_c + \mathbf{x}_i)]$
25: else
26:   /*Exploitation*/
27:   $\theta_{d,i} \leftarrow \arg\max_{\theta_c \in \Theta_c}[\hat{f}_{E,i}(l\angle\theta_c + \mathbf{x}_i) + \eta\Sigma_{E,i}(l\angle\theta_c + \mathbf{x}_i)]$
28: end
29: $\mathbf{x}_{i,next} \leftarrow l_{init}\angle\theta_{d,i}$
```

of Algorithm 3.2).

$$\theta_{d,i} = \underset{\theta_c \in \Theta_c}{\arg\max}(\hat{f}_{E,i}(l\angle\theta_c + \mathbf{x}_i) + \eta\Sigma_{E,i}(l\angle\theta_c + \mathbf{x}_i)) \tag{3.10}$$

$\eta \in \mathbb{R}_{\geq 0}$ defines the confidence interval of GP model.

Local maxima can still happen in a rare situation when all the following conditions hold : i) the exploration range of individual agents, $l_{\text{threshold}}$ (in Algorithm 3.2), is very small, so the distance between all agents and a global maximum is greater than $l_{\text{threshold}}$ ($l_{\text{threshold}}$ can be limited by computational resource), ii) the variance of global maximum is small enough, so the agent can no longer access it, and iii) the estimation result of a global maximum by GP is incorrectly lower than local maxima.

3.3.2 Collision Avoidance and Coordination

For safe multi-robot exploration, I calculate the collision avoidance condition and the coordination weight. Robot i is assumed to know the locations of neighbor robots $j \in \mathcal{N}_i$ within the communication range l_{comm}.

In (3.9), the region farther than the collision-free distance l_{ca} is set as collision-free area Θ_{ca} (lines 7-8 of Algorithm 3.2). The workspace boundary condition is also applied in the same way (lines 5-6 of Algorithm 3.2).

For efficient coordination of multiple robots during the exploration, each agent gives the weights to the directions as far away from its neighbors as possible (lines 23-24 of Algorithm 3.2). The weight function $w_i(\mathbf{x_c})$ for agent i is defined as follows:

$$w_i(\mathbf{x}_c) = \sum_{j \in \mathcal{N}_i} (||\mathbf{x}_j - \mathbf{x}_c||)/(||\mathbf{x}_i - \mathbf{x}_c||) \tag{3.11}$$

It gives a linear weight with respect to $||\mathbf{x}_j - \mathbf{x}_c||$, which is scaled to 1 when the ratio of $||\mathbf{x}_j - \mathbf{x}_c||$ to $||\mathbf{x}_i - \mathbf{x}_c||$ is 1.

3.4 Simulation Result

This section presents a simulation on the virtual environmental model, as shown in Fig. 3.2. This bi-modal environmental model is unknown a priori, and each robot should obtain the sensory data from the current robot location. Since the communication range l_{comm} is 1, some agents may not be able to communicate with each other. The goal of this simulation is to find the highest peak in the entire workspace.

3.4.1 Simulation: robotic sensor networks for 4 agents

In this simulation, I perform distributed GP and active sensing for four agents. They conduct exploration to obtain an estimate of the overall environmental map, considering collision avoidance and coordination, then complete the simulation after rendering the highest peak. I set $\sigma_s^2 = 1$ and $\Sigma = \text{diag}([0.02, 0.02])$ for the Gaussian kernel (2.3). To find a proper E for E-dimensional estimator (3.2) and (3.6), I tested the change in estimation error over E as shown in Fig. 3.6. In general, the larger E, the smaller the estimation error. However, E larger than a certain level increases computation time and memory usage. Therefore, in this simulation, I set E to 80 appropriately.

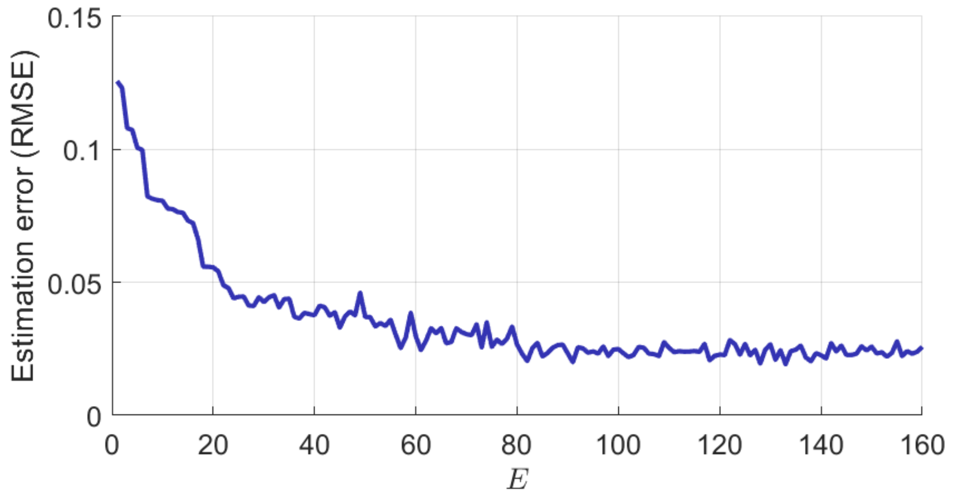


Figure 3.6: GP estimation error according to the hyperparameter.

Fig. 3.7 represents the progress over time from $k = 0$ to $k = 1500$. As shown in Fig. 3.7(a), four agents searched the map together and gathered at the peak position without colliding with one another. All agents moved as far as possible from neighbors to improve exploration efficiency. Also, they avoid being close to the places in which the estimate is already reliable. Fig. 3.7(b) shows the result of the online GP variance estimate for the agent $\#1$ starting from the left-bottom corner, with the result of uncertainty lowering in all regions over time. Similarly, Fig. 3.7(c) shows the GP mean value estimate of the agent $\#1$, which gradually converges to the estimation result similar to Fig. 3.2. The exploitation process starts after $k = 1000$, and the shape of the peak becomes valid. By the average consensus, all the distributed GP estimates of each agent converge (see Fig.3.8).

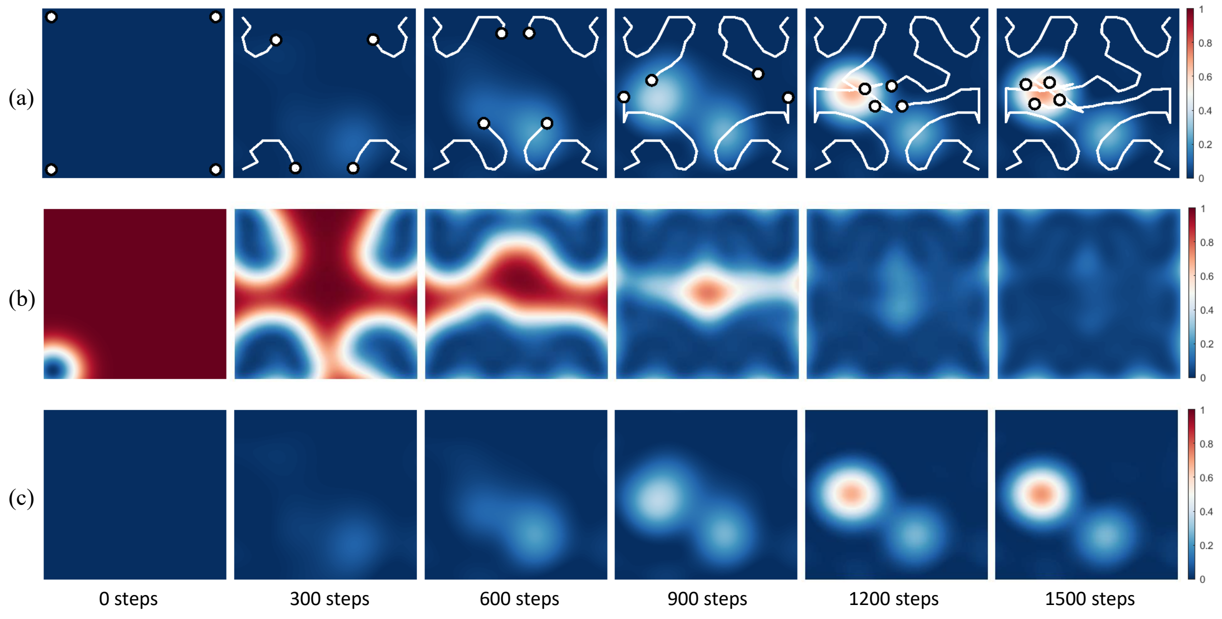


Figure 3.7: Robotic sensor networks simulation for four agents to search for the highest peak location. It shows (a) the environmental process estimate and the trajectories of the agents, (b) the uncertainty propagation, and (c) the change of the GP estimate with time from $k = 0$ to $k = 1500$ in order from the left figure. The environmental model is presented in Fig 3.2. All results are estimates of agent $\#1$ initially located at bottom-left corner.

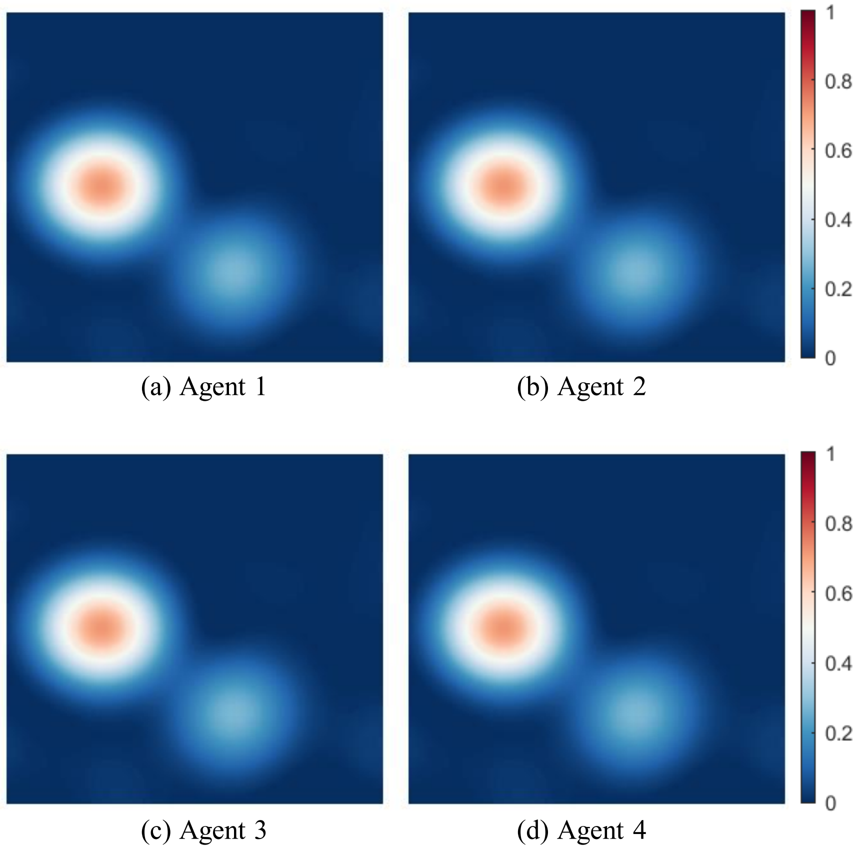


Figure 3.8: Final results of distributed GP for each agent. All GP mean estimates converge to the same because of the average consensus algorithm.

3.5 Experimental Result

3.5.1 Experimental Setup

I validate my algorithm by experiment with multiple UAVs. Fig. 3.1 imitates the rough terrain to be used for the survey experiment. There are three hills with different height in a 4 m×4 m workspace marked with green lines, and the peak values are 39 cm (the beige), 29 cm (the red), and 19 cm (the blue), respectively. Three Crazyflie 2.1 nanocopters shown in Fig. 3.1 measure height above ground level (AGL) using VL53L1x Time-of-Flight (ToF) sensor, which is mounted on the bottom of the Crazyflie. VL53L1x is a laser rangefinder that can measure a distance up to 4 meters, and the measurement error is from $\pm 1\%$ to $\pm 0.15\%$. The Time-of-Flight sensor noise is considered in GP calculation. VICON motion capture system and inertial navigation system (INS) in Crazyflie are used for indoor navigation. They fly at a constant altitude and perform topographic surveys over the entire workspace based on distance values measured with VL53L1x. The entire system is implemented in Robot Operating System (ROS), and the overall algorithm runs at 50 Hz including the online distributed GP. Because of the limited onboard computation resource of Crazyflie nanocopters, GP computation of all the individual UAVs is performed on a laptop (Intel i7-7500U CPU). The communication network is implemented based on the distance between agents, so such setting is not different from the distributed control scenario.

3.5.2 Experiment: robotic sensor networks for 3 agents

In this experiment, all UAVs perform an exploration to estimate the topographic map of the workspace, taking into account collision avoidance and coordination during exploration and complete the experiment at the highest peak (the center of the beige-colored umbrella). I set $\sigma_s^2 = 1$ and $\Sigma = \text{diag}([0.02, 0.02])$ for the Gaussian kernel (2.3), and I set $E = 80$ for E-dimensional estimator (3.2) and (3.6).

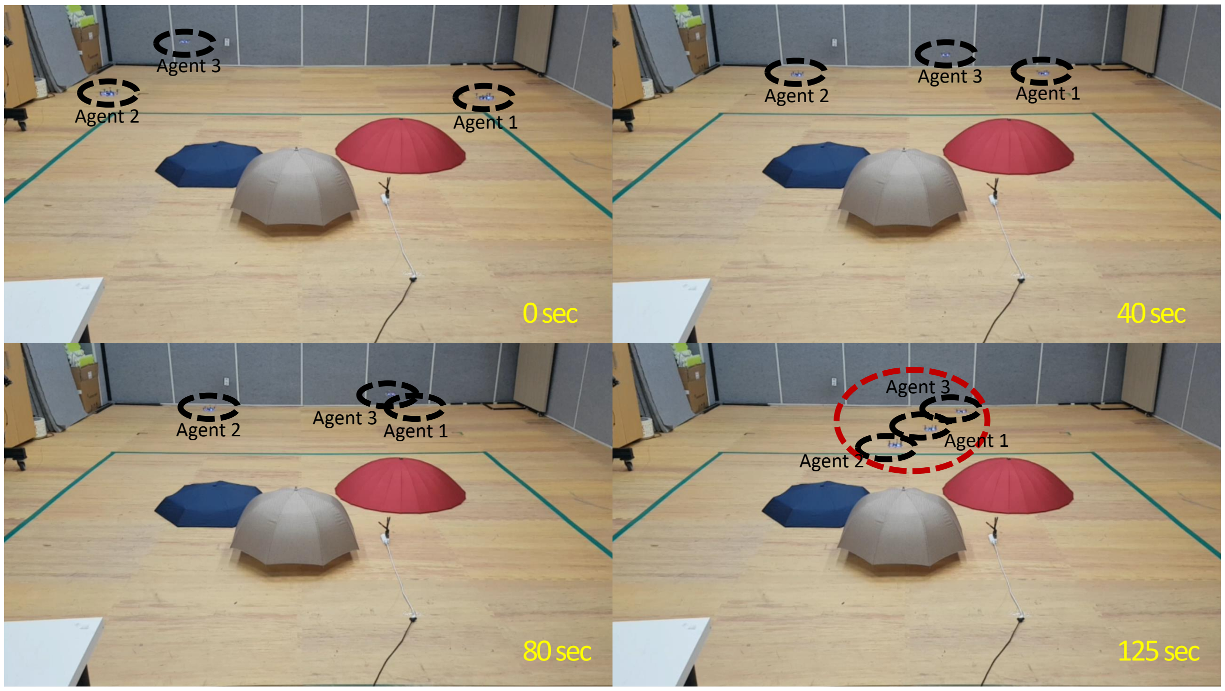
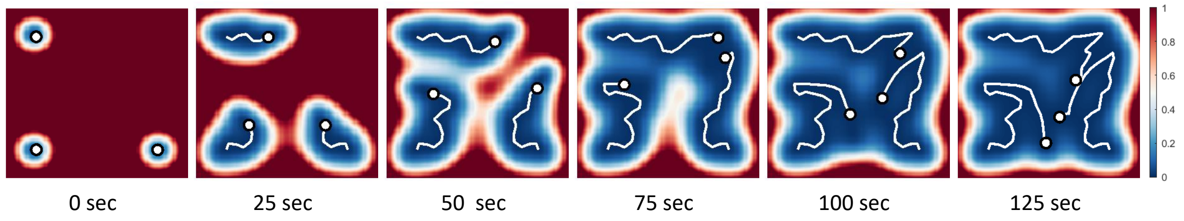


Figure 3.9: 3 UAVs robotic sensor networks experiment to search the highest peak location. Snapshots from 0 to 125 seconds are shown. All UAVs complete the area exploration and at the end gather at the highest peak.

Fig. 3.9 shows some snapshots taken during the experiment. All UAVs travel around the workspace, construct the topographic map using a laser rangefinder, and gather at the highest peak in the end.

(a)



(b)

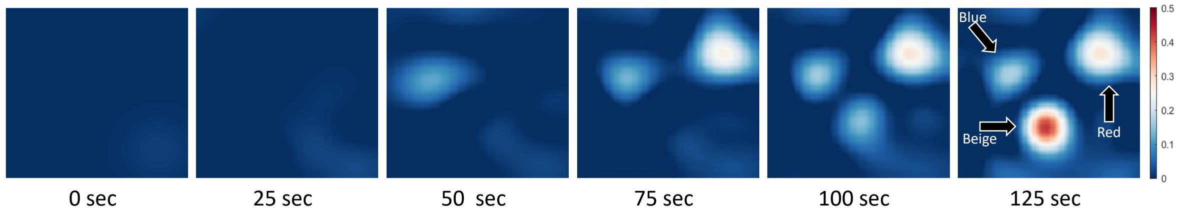


Figure 3.10: Experiment - robotic sensor networks for three UAVs to search for the highest peak location. It shows (a) the uncertainty propagation and (b) the change of the GP estimate with time from 0 to 125 seconds in order from the left figure. All results are estimate of the agent $\#1$.

Fig. 3.10 represents the progress of this experiment during $t = 125$ seconds. Fig. 3.10(a) shows the result of the online GP variance estimate for the agent $\#1$, with the result of uncertainty lowering in all regions over time except that the boundaries still had high uncertainty because of flight zone restriction of UAVs. Similarly, Fig. 3.10(b) shows that the GP mean value estimate of the agent $\#1$ gradually converges to the estimation result similar to Fig. 3.1. The exploitation process starts after about 100 seconds, and the shape of the beige region becomes sharp in Fig. 3.10(b).

3.6 Summary

In this chapter, I present a distributed GP and multi-robot sensor networks to obtain a global environmental model estimate. With the kernel expansion and average consensus algorithm, I implement online distributed GP in a networked control system. Using GP estimate, I establish a multi-agent exploration and exploitation algorithm with collision avoidance and coordination. Then, multi-robot exploration simulation is performed in a virtual environment, and an experiment is performed to construct a topographic map with multiple UAVs. This study is limited to a stationary environmental model, and I am working on expanding this work into a dynamic environment as future work.

4 Informative Path Planning for Multi-Robot Systems

This chapter introduces an information-based path planning technique for environmental learning of multi-robot systems. If a multi-robot team is used for environmental learning, there is an advantage in that a lot of sensor data can be obtained along the path of multiple robots. To maximize this advantage, robots should be located to a position with a lot of information to acquire the most valuable sensor data. In this chapter, I use the Monte Carlo tree search technique to find the path with the largest amount of information. A scalable system that operates in a distributed network environment is constructed using a distributed Gaussian process. Also, by using distributed Monte Carlo tree search, multiple robots coordinate together by using only local communication.

4.1 Introduction

Robotic sensor networks, which combine local sensing capabilities of various sensors with the mobility of robots, can provide more versatility than conventional stationary sensor networks due to their capability to extend the sensing range and improve the resolution of sensory data maps [42]. These networks have been studied extensively in survey of global environment [43–46], industrial environment perception [47], radio signal search [48], and so on.

To construct sensor networks, I first deploy many sensors in a working space. Then, I establish communication channels with the central server to collect and fuse data acquired from all sensors. Since the wireless communication range of sensors is limited, sensors usually make an indirect connection with the central server, such as a mesh network that connects all sensors and the central server by relay channels.

However, the relay network requires a routing table that must be rebuilt every time the robot network is reconfigured, which is cumbersome for robotic sensor networks. This problem is particularly noticeable in UAVs or small robots since they need to use relatively weak communication modules to reduce power consumption.

Decentralizing the system can be a proper solution to network problems by removing the dependency of robots on the central server. For example, if a robot can infer the entire sensory map only from the local information directly provided by surrounding robots, the search task can be completed without the help of the central server. This chapter applies decentralization to the environmental learning phase and the path planning phase, respectively. With an online information fusion algorithm, I build a distributed autonomous system of multiple robots to search and learn even dynamic environments that change over time.

4.1.1 Literature Review

The first part of my work is multi-robot environmental learning in a distributed manner. For environmental learning, some useful techniques exist such as Gaussian mixture model (GMM)

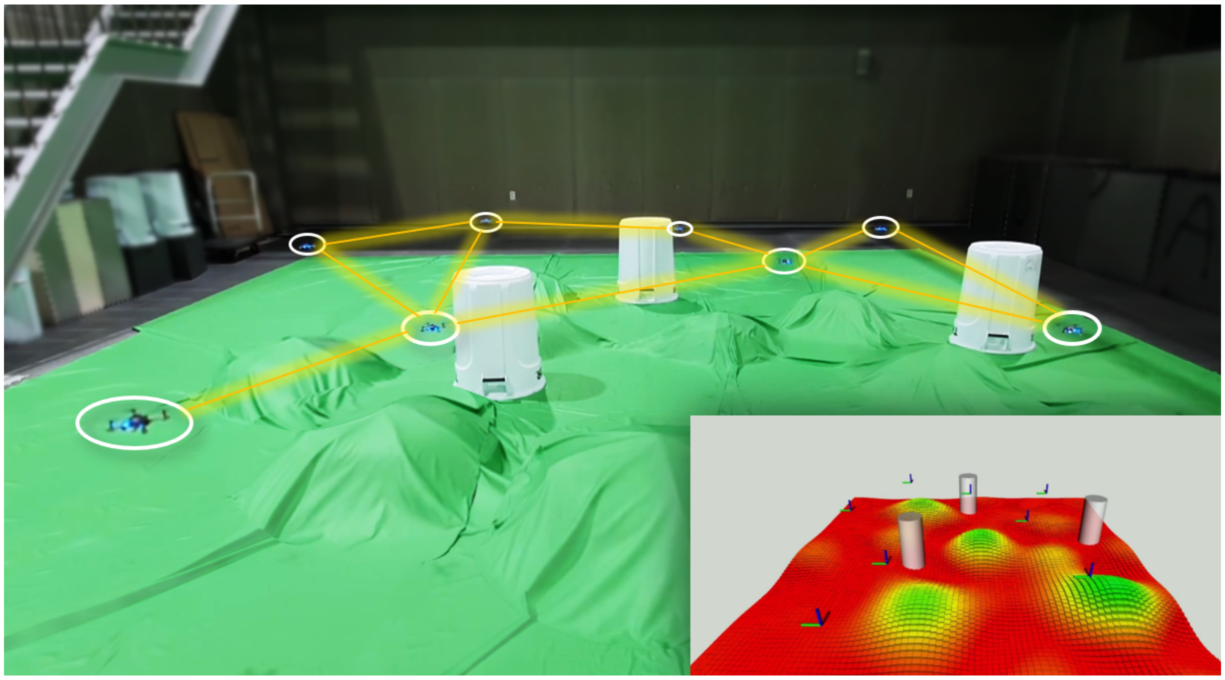


Figure 4.1: Topographic survey using multiple UAVs. All UAVs can only communicate with their neighbors.

[36, 49, 50], finite element method (FEM) [51], and GP regression [45, 46, 48]. In particular, GP is a popular approach that derives a spatial relationship between sampled data using a kernel and performs Bayesian inference for prediction in an unknown region.

However, most GP-related researches focus on centralized systems, making it difficult to expand to large-scale multi-robot systems due to network resource limitations such as channel bandwidth and transmit power. Distributed multi-agent Gaussian regression is introduced in [23], which designs a finite-dimensional GP estimator by using KL expansion [21]. In contrast to the decentralized GP presented in [37], the distributed GP provides a common copy of the global estimate to all agents by exchanging the estimated information with their neighbors. This chapter extends [52], which shows that distributed GPs can construct environmental models using mobile robots in order to take the distributed path planning into account.

The second part of my work is informative path planning in a distributed multi-robot system. As an initial study of informative path planning, the problem of optimal sensor placement has been investigated to create an environmental map in a given space by properly placing a finite number of sensors [36, 50, 53]. Since then, by applying GPs and information theory, the research of optimal sensor placement has grown into the informative path planning research as presented in [35, 40, 46, 49, 54–56]. Some studies have combined GP with conventional planning algorithms such as rapidly-exploring random tree (RRT) [57], dynamic programming (DP) [35], or Monte Carlo tree search (MCTS) [58].

Besides the above approaches that mainly focus on informative path planning for single agents, many studies have applied informative planning for multi-robot systems. In [37, 48], although both studies deal with decentralized multi-robot exploration using GP, these algorithms are not scalable as they consider only two robots. [59] introduced the combination of the Kalman filter (KF) and the reduced value iteration (RVI) method for the parallelized active information gathering. While this technique is scalable to a large number of robots, it is noted that the environmental model has to be known, and only discrete environments can be represented since the model is expressed in KF. Considering the scalability for multi-robot systems, I extend the MCTS path planning in a distributed manner to be compatible with the distributed GP.

4.1.2 Contribution

To achieve my goal of fully distributed multi-robot informative planning, I divide the whole process into two phases: environmental learning and path planning. During these phases, I focus on three main contributions as follows.

- I propose a distributed informative path planning algorithm using a distributed MCTS combined with GP. In addition, I introduce the trajectory merging method to consider predicted trajectories of other agents.
- I build a fully distributed exploration and learning architecture using only local peer-to-peer communication for system scalability, as shown in Table 4.1.
- I perform a multi-robot exploration simulation with a virtual environment setting and hardware experiment for topographic survey as shown in Fig. 4.1.

The outline of this chapter is as follows. Sec. 2 briefly describes a multi-robot system setup and preliminaries. Sec. 3 combines environmental learning and MCTS in the distributed system. Simulations for the synthetic environment and real-world dataset are presented in Sec. 4. Finally, Sec. 5 concludes the chapter.

Table 4.1: Scalability comparison between centralized and distributed systems for multi-agent tasks. See text for symbols.

	Centralized	**Distributed (ours)**
GP Computation Complexity	$O((mn)^3)$ ([37, 48])	$O(E^3)$ ([52])
MCTS Planner Action Cardinality	$\|\mathcal{A}\|^n$ ([60])	$\|\mathcal{A}\|$ ([61])
Communication Complexity	$O(n^2)$ ([49, 60])	$O(n)$ ([37, 48, 52])

4.2 Multi-Robot System Setup and Preliminaries

I focus on the environment learning problem in multi-robot systems by considering a target domain as a 3-dimensional compact set $\mathbb{X}_w \subset \mathbb{R}^3$. Multiple robots (e.g., ground vehicles or UAVs with onboard sensors) explore an unknown area and estimate environmental information using both self-measurements and shared data received from neighbors. All robots can discover obstacles nearby using the range sensor and only communicate with adjacent robots within the communication distance.

4.2.1 Multi-Robot System Setup

Each robot has its process modules, *Distributed GP* and *Distributed MCTS*, for the distributed monitoring task. During these processes, they share GP variables and predicted trajectories through

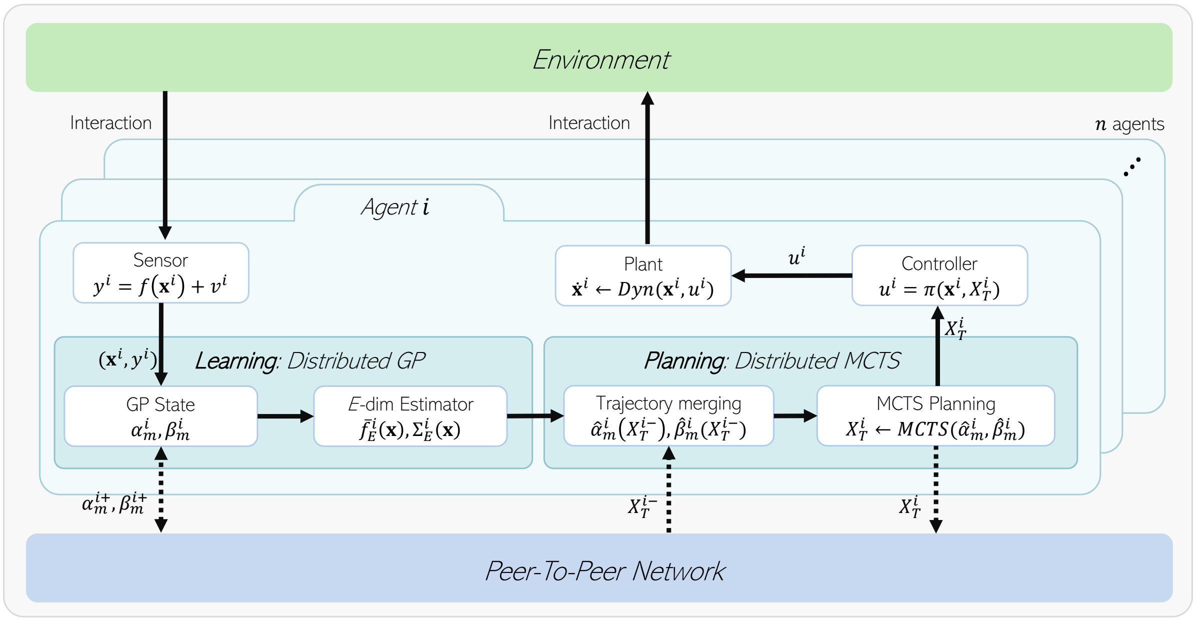

Environment
Interaction
Interaction
n agents
Agent i
Sensor
$y^i = f(\mathbf{x}^i) + v^i$
Plant
$\dot{\mathbf{x}}^i \leftarrow Dyn(\mathbf{x}^i, u^i)$
u^i
Controller
$u^i = \pi(\mathbf{x}^i, X_T^i)$
X_T^i
$(\mathbf{x}^i, y^i)$
Learning: Distributed GP
Planning: Distributed MCTS
GP State
α_m^i, β_m^i
E-dim Estimator
$\bar{f}_E^i(\mathbf{x}), \Sigma_E^i(\mathbf{x})$
Trajectory merging
$\hat{\alpha}_m^i(X_T^{i-}), \hat{\beta}_m^i(X_T^{i-})$
MCTS Planning
$X_T^i \leftarrow MCTS(\hat{\alpha}_m^i, \hat{\beta}_m^i)$
$\alpha_m^{i+}, \beta_m^{i+}$
X_T^{i-}
X_T^i
Peer-To-Peer Network

Figure 4.2: Structure of the distributed exploration and environmental model learning system. Each agent has its own distributed GP and distributed MCTS planner modules that operate through peer-to-peer communication with each other.

a peer-to-peer communication network. This operation process is summarized in Fig. 4.2. The controller design process is not covered in this work.

4.2.2 Informative Path Planning

To obtain the better description of a spatial process model, robots perform informative path planning. It maximizes the *information gain* $\mathbb{I}(;)$, which is the mutual information between the process f and measurements $\mathbf{y}$:

$$\mathbb{I}(\mathbf{y}; f) = \mathrm{H}(\mathbf{y}) - \mathrm{H}(\mathbf{y}|f), \tag{4.1}$$

where $\mathrm{H}(\cdot)$ is the *entropy* of a random variable. Let $X^i_{1:k}$ be the possible trajectory of robot i and $X_{1:k} = \{X^1_{1:k}, \cdots, X^n_{1:k}\}$ be the possible trajectories of all robots. Then, the multi-robot team's global objective function $J(X_{1:k})$ is defined as follows:

$$J(X_{1:k}) = \mathbb{I}(\mathbf{y}_{1:k}; f). \tag{4.2}$$

$\mathbf{y}_{1:k}$ is the measurements corresponding to $X_{1:k}$. As a result, the optimal trajectories for all agents are defined as follows:

$$\begin{aligned} X^*_{1:k} &= \arg\max_{X_{1:k}} J(X_{1:k}) \\ &= \arg\max_{X_{1:k}} \mathbb{I}(\mathbf{y}_{1:k}; f) \\ &= \arg\max_{X_{1:k}} (\mathrm{H}(\mathbf{y}_{1:k}) - \mathrm{H}(\mathbf{y}_{1:k}|f)). \end{aligned} \tag{4.3}$$

In this study, the entropy $\mathrm{H}(\cdot)$ is obtained using GP. With the result of GP estimation (2.5), the optimal trajectory generation for each agent will be addressed in Section 4.3.

4.3 Path Planning: Distributed Monte Carlo Tree Search

Using the distributed model learning discussed in the previous section, all agents create a local environmental map that converges to the global environmental map even in a distributed network. To find the most promising search trajectories with the learned map, all agents should consider every possible action. However, because the cardinality of possible action set grows exponentially with respect to the number of robots, the distributed planning strategy is needed in multi-robot path planning [59]. In [61], the decentralized MCTS approach is studied to alleviate the cardinality of possible action set from $|\mathcal{A}|^n$ to $|\mathcal{A}|$, where $\mathcal{A}$ represents the discrete action space of each robot. I apply this advantage to my GP-based informative planning of multiple robots. With the distributed MCTS, each robot calculates a promising trajectory by communication with neighboring agents only. This process is shown in Fig. 4.3. This section introduces the trajectory merging method to reflect the neighbor's path in each agent's tree search process. The contents of this section are summarized in Algorithm 4.1 and 4.2.

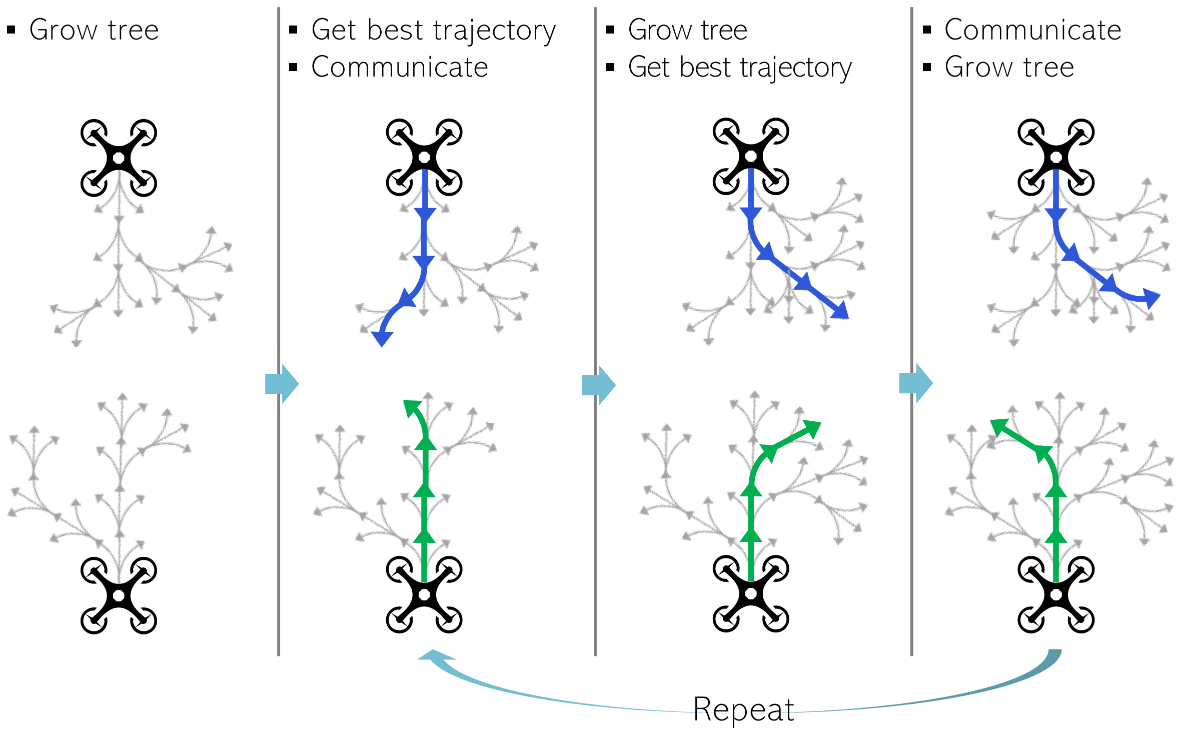


Figure 4.3: Illustration of the distributed MCTS process. Nearby robots exchange their predicted trajectories. These trajectories are used to temporarily update GPs and grow search trees. This process is repeated until the time budget is met.

4.3.1 Trajectory Merging

For each agent i, $X^i_{k+1:k+T} = (\hat{\mathbf{x}}^i_{k+1}, \cdots, \hat{\mathbf{x}}^i_{k+T})$ denotes the predicted trajectory with the prediction length T, or it can be represented by X^i_T for brevity. Assuming that the agent i receives the predicted trajectories of neighboring agents $X^{i-}_T = \{X^j_T\}_{j\in\mathcal{N}_i}$, I modify the GP state α^i_m as follows:

$$\begin{aligned}\hat{\alpha}^i_m\,(X^{i-}_T) = {} & \frac{mn}{mn + \mathrm{n}(X^{i-}_T)}\alpha^i_m \\ & + \frac{\mathrm{n}(X^{i-}_T)}{mn + \mathrm{n}(X^{i-}_T)} \sum_{j\in\mathcal{N}_i}\sum_{t=1}^{T} \Phi(\hat{\mathbf{x}}^j_{k+t})\Phi^\top(\hat{\mathbf{x}}^j_{k+t}).\end{aligned} \tag{4.4}$$

$\mathrm{n}(X^{i-}_T)$ is the number of sensing points included in X^{i-}_T. With (4.4) and the E-dimensional estimator in (3.7), I define the trajectory-merged GP estimator as follows:

$$\begin{aligned}\hat{\Sigma}^i_E(\mathbf{x}) := {} & \kappa(\mathbf{x},\mathbf{x}) \\ & -\Phi^\top(\mathbf{x})\left(\hat{\alpha}^i_m + \frac{\sigma_v^2}{mn + \mathrm{n}(X^{i-}_T)}\Lambda_E^{-1}\right)^{-1} \\ & \times\hat{\alpha}^i_m\Lambda_E\Phi(\mathbf{x}).\end{aligned} \tag{4.5}$$

In this way, predicted trajectories of neighboring agents are temporarily included in the acquired data set of the GP estimator. Because (4.4) and (4.5) are temporary values for tree search in distributed MCTS, they do not affect α^i_m and disappear after getting new predicted trajectories. This process is summarized in the Distributed MCTS block of Fig. 4.2. With this result, the path planning process will be explained in the next section.

4.3.2 Informational Reward Function

As I mentioned in (4.2), the information gain $\mathbb{I}$ is the objective function I have to maximize. With the definition in (4.3), the optimal trajectory considering neighboring paths is defined as follows.

$$\begin{aligned}X^{i*}_T &= \operatorname*{arg\,max}_{X^i_T\in\mathbb{X}^i_k} J(X^i_T \cup X^{i-}_T \cup X_{1:k}) \\ &= \operatorname*{arg\,max}_{X^i_T\in\mathbb{X}^i_k} \mathbb{I}(\mathbf{y}^i_T \cup \mathbf{y}^{i-}_T \cup \mathbf{y}_{1:k}; f),\end{aligned} \tag{4.6}$$

$\mathbf{y}_T^i$ and $\mathbf{y}_T^{i-}$ are the measurements corresponding to X_T^i and X_T^{i-}, respectively. $\mathbb{X}_k^i$ is the domain of possible trajectories for agent i. As shown in (4.1), information gain is represented with entropies as follows:

$$\begin{aligned}\mathbb{I}(\mathbf{y}_T^i \cup \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k}; f) = \mathrm{H}(\mathbf{y}_T^i \cup \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k}) \\ -\mathrm{H}(\mathbf{y}_T^i \cup \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k} | f).\end{aligned} \tag{4.7}$$

Using the mesuarement model (2.1) and the entropy calculation for the normal distribution [62], conditional entropy becomes

$$\mathrm{H}(\mathbf{y}_T^i \cup \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k} | f) = \tfrac{1}{2} \log |2\pi e \sigma_v^2 I|. \tag{4.8}$$

$\mathrm{H}(\mathbf{y}_T^i \cup \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k})$ is decomposed using conditional entropy, and it is obtained by calculating the entropy of GP as follows:

$$\begin{aligned}\mathrm{H}(\mathbf{y}_T^i \cup \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k}) &= \mathrm{H}(\mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k}) + \mathrm{H}(\mathbf{y}_T^i | \mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k}) \\ &\approx \mathrm{H}(\mathbf{y}_T^{i-} \cup \mathbf{y}_{1:k}) + \tfrac{1}{2} \log |2\pi e \hat{\Sigma}_E^i(X_T^i)|.\end{aligned} \tag{4.9}$$

As a result, with the trajectory-merged GP estimator in (4.5), the optimal trajectory for agent i is defined as follows:

$$\begin{aligned}X_T^{i*} &= \underset{X_T^i \in \mathbb{X}_k^i}{\arg\max}\, J(X_T^i \cup X_T^{i-} \cup X_{1:k}) \\ &\approx \underset{X_T^i \in \mathbb{X}_k^i}{\arg\max}\, \tfrac{1}{2} \log |2\pi e \hat{\Sigma}_E^i(X_T^i)| \\ &= \underset{X_T^i \in \mathbb{X}_k^i}{\arg\max}\, \mathcal{R}^i(X_T^i).\end{aligned} \tag{4.10}$$

I call $\mathcal{R}^i(\cdot)$ the *informational reward function*, which is utilized in the tree search algorithm.

4.3.3 Tree Search with D-UCB Alogrithm

Using the informational reward function defined in 4.3.2, the tree search algorithm iteratively explores and evaluates predictive path candidates according to the discounted upper confidence

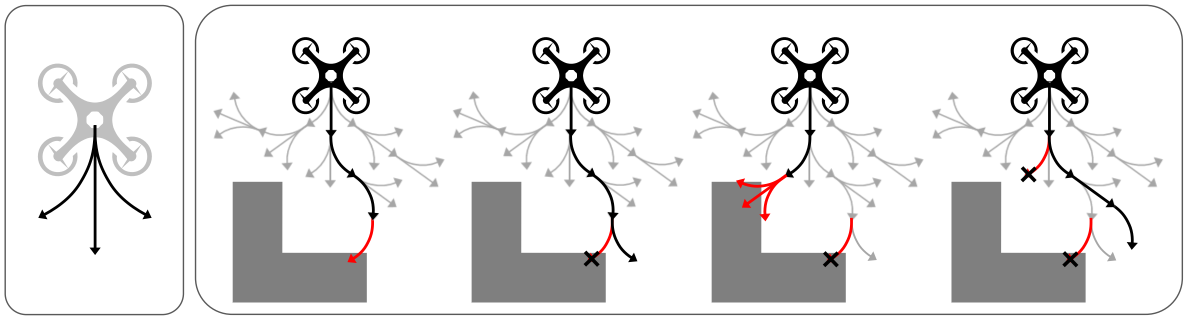


Figure 4.4: Node closing method for obstacle avoidance in MCTS. (left) finite action space that the robot can take. (right) tree expansion and node closing process.

bound (D-UCB) rule to find the optimal path. UCB rule assigns the probabilistic search priority to the action candidates [63].

The tree structure consists of nodes s and edges (s, a) for all legal actions $a \in \mathcal{A}(s)$. Each edge contains a set of variables $\{N_s^a, W_s^a, \tau_s^a, C_s^a\}$ where N_s^a is the visit count, W_s^a is the total action value, τ_s^a is the number of iterations for tree search (shown in line 5 of Algorithm 4.2), and C_s^a is a closing variable which will be discussed. I follow the tree search process in Algorithm 4.1 and the distributed MCTS with GP in Algorithm 4.2. The MCTS process can be divided into four main steps as follows.

Selection (lines 4, 12-22 of Algorithm 4.1): The selection phase focuses on finding a leaf node s_{leaf}. Following the D-UCB rule, the selected action at node s is defined as follows [62]:

$$a_t = \underset{a \in \mathcal{A}(s)}{\arg\max} \left(\frac{W_s^a}{N_s^a} + U_s^a \right), \tag{4.11}$$

where

$$U_s^a = \sqrt{\frac{\ln \sum_{a' \in \mathcal{A}(s)} N_s^{a'}}{N_s^a}}. \tag{4.12}$$

The first term on the right-hand side of (4.11) means exploration term for the tree search, and the second term means exploitation term. As shown in Algorithm 4.2, each agent periodically receives the predicted trajectories of adjacent agents, which are utilized in the tree search process. It means that the tree, obtained by using previously given trajectories, may not be optimal when new neighboring trajectories are received. Therefore, adopting the discount factor γ makes the previously visited nodes less influential on the current UCB value.

Expansion (line 5 of Algorithm 4.1) : The expansion phase expands the selected node with uniformly sampled action from the action space $\mathcal{A}(s)$ if the depth of the selected node does not exceed the search depth T. When the expanded node collides with an obstacle, the algorithm closes this edge ($C_s^a \leftarrow 1$) and returns to the selection phase.

Simulation (lines 9, 23-27 of Algorithm 4.1) : In the simulation phase, it calculates the informational reward $\mathcal{R}_i(X_T^i)$ of the selected trajectory. If the selected node's depth is less than the search depth T, it performs random walks. After that, the reward is calculated with the predicted trajectory X_T^i as shown in Section 4.3.2.

Backpropagation (line 10 of Algorithm 4.1) : The edge variables are updated in a backward pass. The visit counts are incremented, $N_s^a \leftarrow N_s^a \gamma^{\tau - \tau_s^a} + 1$, and the total action value is updated, $W_s^a \leftarrow W_s^a \gamma^{\tau - \tau_s^a} + R_i$. As described in the *selection* step, the discount factor γ is applied to reduce the weight of the previous value.

4.4 Result and Analysis

This section presents environmental learning simulations on the various situations and a hardware demo. The simulation is conducted on a synthetic environment, and the experiment is on a artificial terrain environment. These environmental models are unknown a priori, and each robot obtains the sensory data from the current location. Furthermore, since the communication range is finite, some agents may not be able to communicate with each other.

4.4.1 Simulation - synthetic environment learning

I perform the fully distributed informative planning simulation for multiple agents. They conduct exploration to obtain an estimate of the environmental map, considering collision avoidance and coordination. They can communicate only with neighbors within the range of 10 m (the map size

Algorithm 4.1: Tree Search with Obstacle Avoidance

```
1  Function TreeSearch($T_k^i, \tau$):
2      $s_0 \leftarrow$ getRootNode($T_k^i$);
3      for $t \leftarrow 1$ to $N_{iteration}$ do
4          $s_{leaf} \leftarrow$ selection($s_0, \tau$);
5          $(s_t, a) \leftarrow$ expansion($s_{leaf}, \tau$);
6          if collisionCheck($s_t$) then
7              $C_{s_{leaf}}^{a} \leftarrow$ true;
8              continue;
9          $r_t \leftarrow$ simulation($s_t$);
10         backprop($s_t, r_t$);
11     return $T_k^i$;
12 Function selection($s_{leaf}, \tau$):
13     while not leafNodeFound do
14         if not $depth(s_{leaf}) > T$ then
15             if $C_{s_{leaf}}^{a} =$ true $\forall a \in \mathcal{A}(s_{leaf})$ then
16                 $C_{s_{leaf-1}}^{a_{leaf-1}} \leftarrow$ true;
17                 back to the parent node;
18                 $s_{leaf} \leftarrow s_{leaf-1}$;
19             else
20                 $a \leftarrow$D-UCB($s_{leaf}, \tau$)                    ▷ eq. (4.11);
21                 $s_{leaf} \leftarrow getNode(s_{leaf}, a)$;
22     return $s_{leaf}$;
23 Function simulation($s_t$):
24     $X_T^i \leftarrow$ trajectory from $s_0$ to $s_t$;
25     for $j \leftarrow$ depth($s_t$) to $T$ do
26         $X_T^i \leftarrow \{X_T^i$, randomWalk($\mathbf{x}_{k+j}^i$)$\}$;
27     return $\mathcal{R}^i(X_T^i)$;
```

Algorithm 4.2: Distributed MCTS with GP for agent i

```
   Input: $\mathbf{x}_k^i, \alpha_{m-1}^i, \beta_{m-1}^i, X_T^{i-}$
   Output: $X_T^i$
1  $y_k^i \leftarrow$ getMeasurement($\mathbf{x}_k^i$)                          ▷ eq. (2.1);
2  $(\alpha_m^i, \beta_m^i) \leftarrow$ updateGP($\alpha_{m-1}^i, \beta_{m-1}^i, \mathbf{x}_k^i, y_k^i$)          ▷ eq. (3.8);
3  $(\alpha_m^i, \beta_m^i) \leftarrow$ GPconsensus($\alpha_m^i, \beta_m^i$)                   ▷ eq. (18) in [52];
4  $T_k^i \leftarrow$ initializeTree($\mathbf{x}_k^i$);
5  for $\tau \leftarrow 1$ to $N_{search}$ do
6      $(\hat{\alpha}_m^i, \hat{\beta}_m^i) \leftarrow$ TrajectoryMerging($X_T^{i-}$)           ▷ eq. (4.4);
7      $T_k^i \leftarrow$ TreeSearch($T_k^i, \hat{\alpha}_m^i, \hat{\beta}_m^i, \tau$)               ▷ Algorithm 4.1;
8      $X_T^i \leftarrow$ getBestTraj($T_k^i$);
9      $X_T^{i-} \leftarrow$ communicateTraj($X_T^i$);
10 return $X_T^i$;
```

is 20 m $\times$ 20 m) and move at 1 m/s constantly. I set $\sigma_s^2 = 1$ and $\Sigma_l = \text{diag}([0.02, 0.02])$ for the Gaussian kernel (2.3), and I set $E = 80$ for E-dimensional estimator (3.2) and (3.6).

The progress over time from 0 to 50 seconds is shown in Fig. 4.5. As shown in Figs. 4.5(a)-(b), twelve agents search the map together and generate the GP estimate, where the ground truth model is presented in 4.5(c). The agents scatter naturally and find the next locations to be updated based on the variance map. Also, as they avoid the places where the estimate is already reliable, they can minimize the redundant actions that can decrease the exploration efficiency. Through Fig. 4.5(b), it can be confirmed that the information of all agents is diffused through a communication link.

Although Figs. 4.5(a)-(b) show results from agent $\#1$ only, all the distributed GP estimates of each agent converge to the same by the average consensus as shown in Fig. 4.6. In other words, all the agents construct a global GP estimation map in a distributed manner despite sharing the local measurements with their neighbors only.

I conduct more simulations in various environments to investigate the factors that affect search performance. In the tree search algorithm, the search depth T and the number of iterations $N_{iterations}$ are the factors that directly affect the search result. The simulation results in Figs.

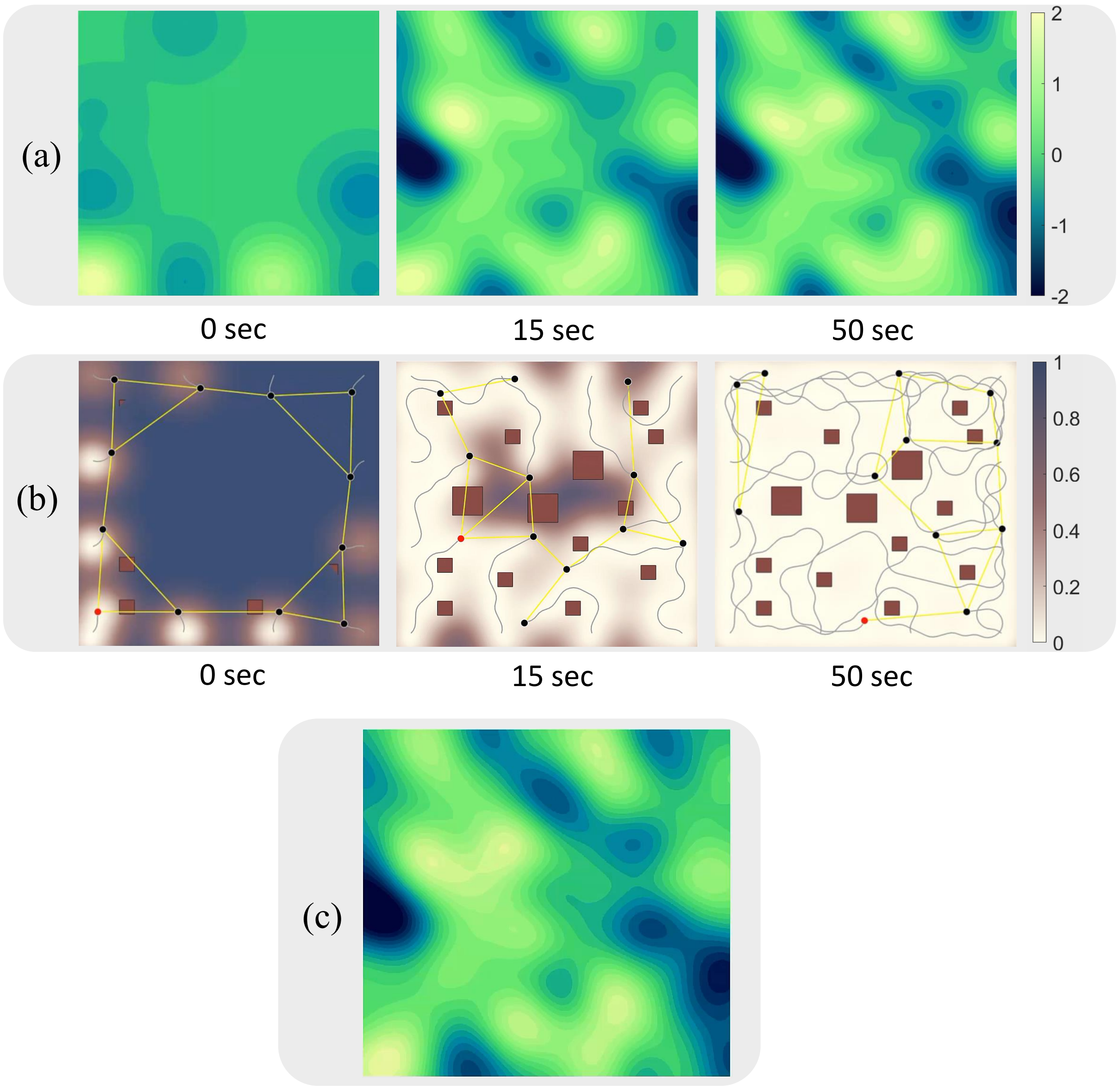


Figure 4.5: Simulation A. The process of the environmental model construction by 12 agents with fully distributed informative planning. (a) Change of GP estimate and (b) uncertainty propagation over time from 0 to 50 seconds in order from the left figure. (c) Ground truth of environmental model. Yellow lines in (b) indicate communication links between agents. All the presented results are obtained by agent $\#1$ (red dot in (b)).

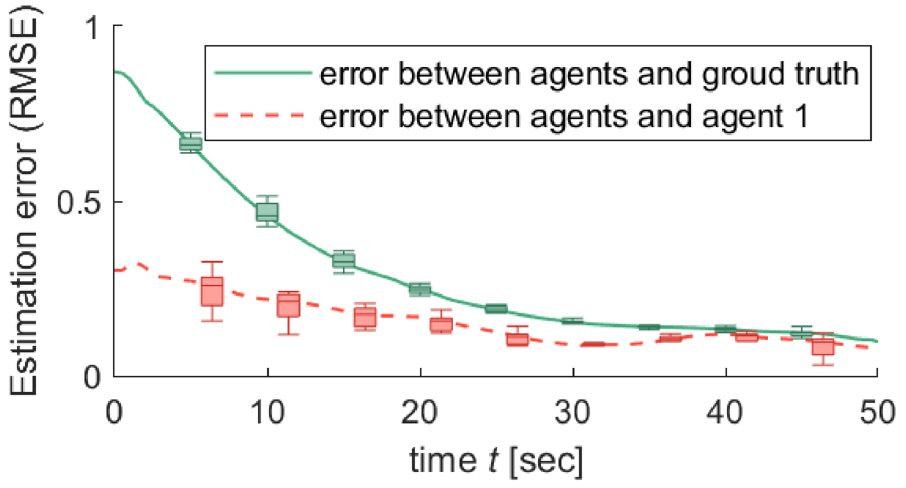
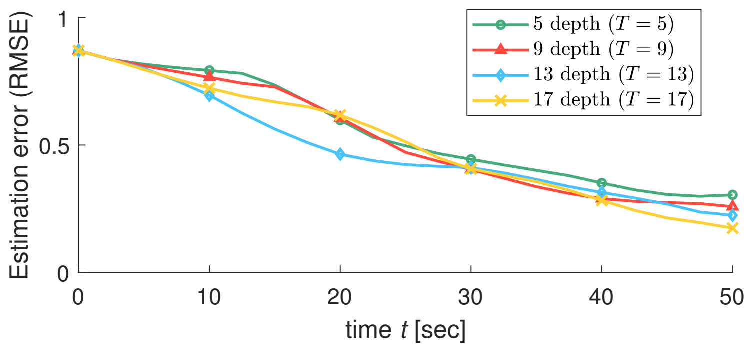


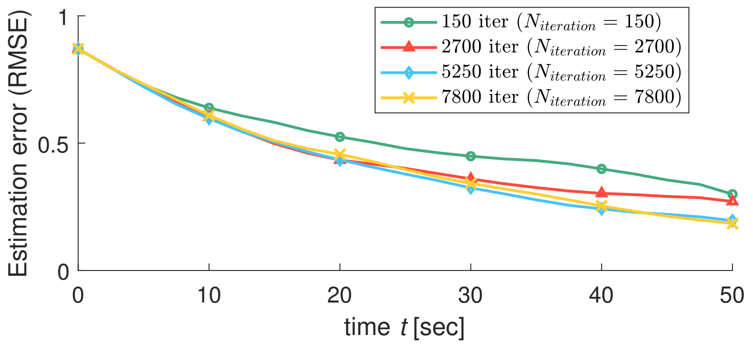
Figure 4.6: Simulation A. Environmental model estimation error. The solid green line shows the box plot of RMSE values between all agents and the ground truth. The dashed red line shows the box plot of RMSE values between all agents and the agent #1. This graph means that all GP estimation results converge to the same result with only local communication.

4.7(a)-(b) show that the search performance is proportional to both T and $N_{iterations}$. The deeper the search, the longer paths are considered. As the number of searching iterations increases, the probability of finding an optimal route increases. Fig. 4.7(c) shows that the search performance can be improved as the number of agents increases through a distributed algorithm. From these results, I can see that my algorithm is scalable for a large number of robots as well.

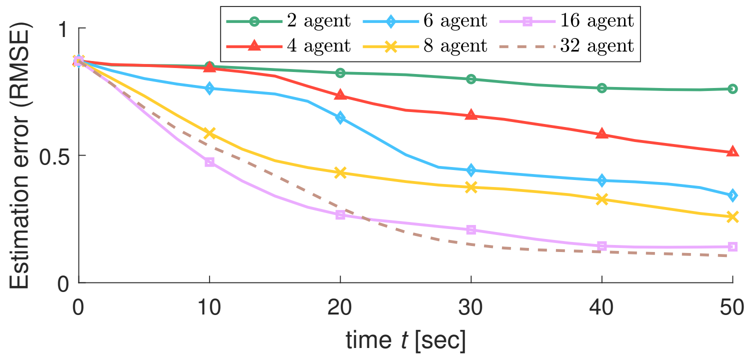
Fig. 4.8 compares the search performance of distributed systems, centralized systems, and independent systems. In the independent system, agents explore the area without communication. The centralized system has a central server that gathers all the information regardless of the communication range, and the server calculates paths for all agents. Because the action space of centralized system $(\mathrm{n}(\mathcal{A})^n)$ is much bigger than that of distributed system $(\mathrm{n}(\mathcal{A}))$, the centralized system requires greater $N_{iterations}$ than the distributed system. Even with limited communication and much fewer iterations, the distributed system performs similarly to the centralized system.



(a) Search depths



(b) Number of iterations



(c) Number of agents

Figure 4.7: Simulation B. Environmental model construction with different conditions. (a) 6 agents' exploration results with different search depths. (b) 8 agents' exploration results with the different number of iterations. (c) Exploration results with the different number of agents.

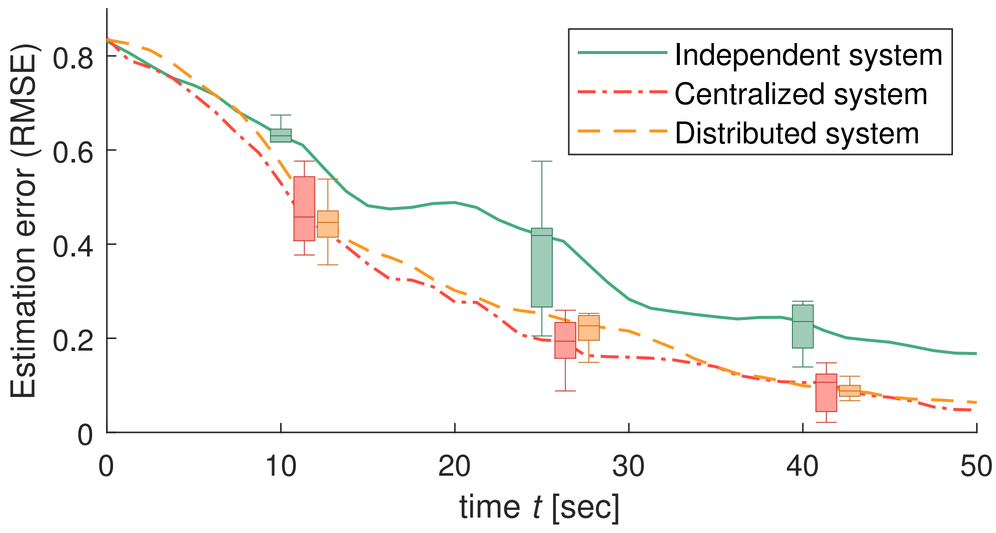
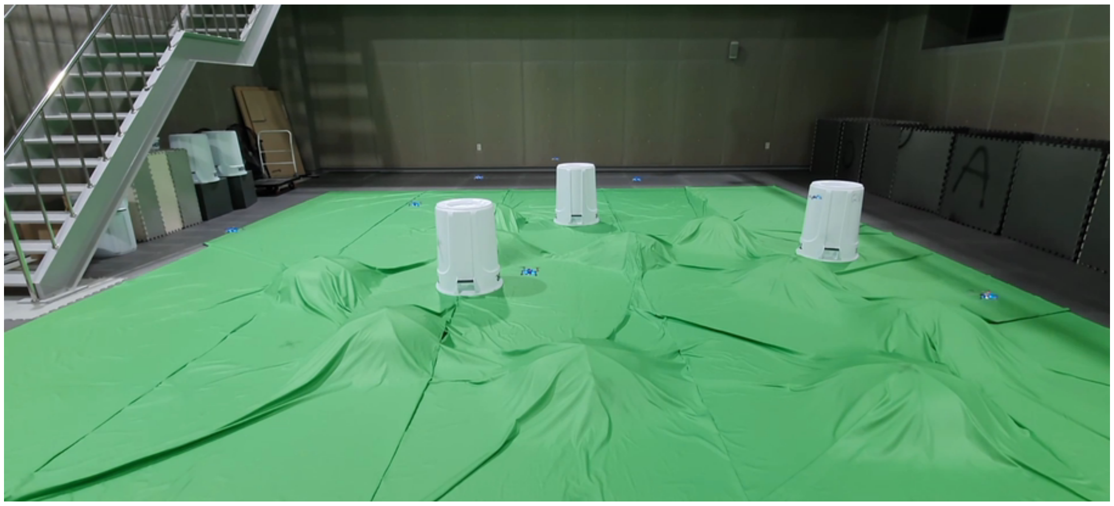


Figure 4.8: Simulation C. 4 agents' exploration results with different type of systems for 10 trials (with different environments).

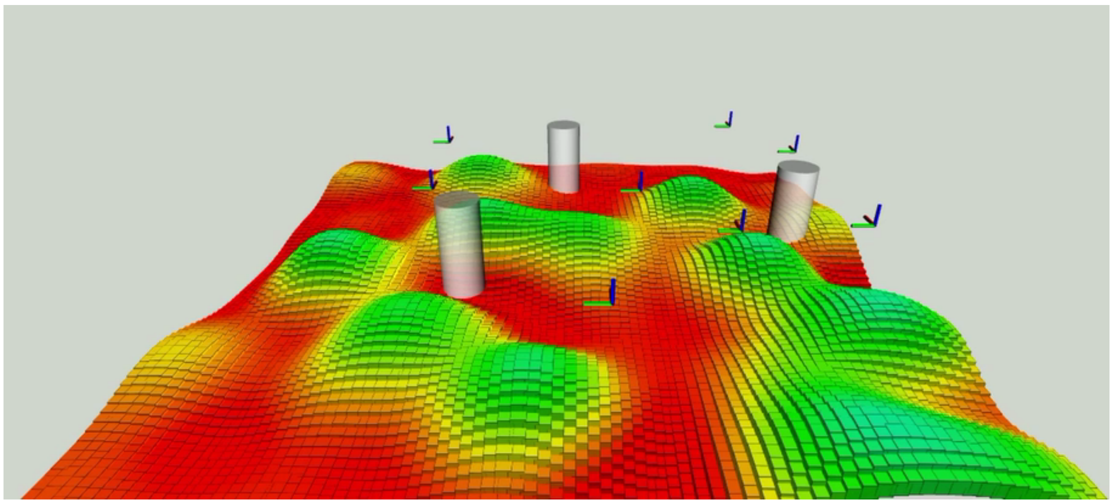
4.4.2 Experiment - topographic survey using multiple UAVs

I conduct a hardware demonstration with multiple UAVs. Fig. 4.1 represents the experimental setup for the topographic surey on the 6 m $\times$ 6 m rough terrain. There are eight Crazyflie 2.1 quadrotors to measure height AGL using VL53L1x ranging sensor. VL53L1x has the measurement error from $\pm 1\%$ to $\pm 0.15\%$, which error is considered in GP estimation. OptiTrack motion capture system is used for indoor navigation. The entire system is implemented in ROS, and the computation of GP and planning for each agent is run on the ground station due to the limited computational resource of Crazyflie 2.1.

In this scenario, a team of UAVs flies over the search area and gathers height data from the current UAV's location. Because their communication distance is limited, sometimes they can be disconnected from one another. Fig. 4.9 shows snapshots taken during the experiment. The UAV team explores the unknown environment, measures the height AGL, and estimates the terrain with only local communication and distributed GP regression.



(a) Unknown target environment and a team of UAVs.



(b) The result of topographic survey with UAVs.

Figure 4.9: Experiment results. The three white pillars are the obstacles. After 40 seconds, the UAV team complete the area exploration and topographic survey.

4.5 Summary

This chapter presents fully distributed robotic sensor networks to obtain a global environmental model estimate. I combine the Gaussian process with the Monte Carlo tree search in a distributed manner for peer-to-peer communication. My method allows multiple robots to collaboratively perform exploration, taking into account collision avoidance and coordination. I validate my algorithm in various scenarios through simulations and experiments. The results confirm that multiple agents can successfully explore the environment, and it is scalable with the increasing number of agents in the distributed network.

5

Target Search and Tracking with Reinforcement Learning

This chapter introduces multi-target search and tracking techniques using multiple robots. To find a multi-target whose initial location is unknown, Gaussian process-based signal intensity map learning is preceded. Based on the learned signal intensity map, I aim to locate each robot near targets. However, it is not possible to know the initial position of targets, nor to accurately estimate target positions from the learned map. This problem makes it difficult to track multiple targets using the conventional path planning algorithms. In this chapter, I introduce deep reinforcement learning with a signal intensity map as an input, leading to the most efficient search and tracking scheme based on past experiences. The learned robots can be transferred immediately even to a new environment.

5.1 Introduction

Multi-agent systems have garnered increasing attention recently within the robotics community. These systems have great potential to address practical applications in large-scale and unknown environments, including but not limited to search and rescue, reconnaissance, and surveillance [46, 49, 64, 65]. Specifically, I am interested in the challenge of multi-agent target search and tracking, which requires processing the combined signal intensity from multiple targets measured by agents at various locations and time intervals. In general, the signal intensity field is unknown beforehand in these situations [66]. Planning efficiently over the unknown signal field is not straightforward and leads to the exploration-exploitation dilemma [62]. Reinforcement learning (RL) is a promising approach to intelligently deal with this dilemma by learning from past experiences in a data-driven manner [67].

However, there are multiple challenges to applying RL to a multi-agent system in unknown signal fields. While extending RL from single agent to multi-agent is straightforward in theory, it may be infeasible in practice. The observation-action space scales exponentially to the number of agents which hinders optimization (curse of dimensionality). Scalability issues aside, the number of agents or the connection between agents can vary in multi-agent systems, which is difficult to accommodate with existing RL algorithms. In addition, partial-observability of the target search and tracking mission renders it incompatible with RL, which assumes a fully-observable Markov decision process (MDP).

There exists a body of research on multi-agent reinforcement learning (MARL). These methods formulate the problem from a generalized Markov game perspective, such as decentralized partially-observable MDP (POMDP), and they assume a sparse interaction between agents to avoid the scalability issue [68, 69]. This stems from the fact that each agent, in most cases, can rely only on its information and act independently of other agents unless coordination between agents is required. Thus, existing approaches mainly focus on partial-observability arising from agent-centric observation and sparse interactions rather than from the problem setting itself. However, this is not the case for distributed multi-agent target search and tracking, where partial-observability appears directly in the form of an unknown signal intensity environment.

In this chapter, I propose to augment deep reinforcement learning (DRL) with distributed GP multi-agent consensus-based map building to tackle the partial-observability of the search and tracking mission. This hybrid approach transforms the original problem of POMDP into MDP by incorporating the belief of the environment map into the observation input of the RL agent. An added benefit of this approach is in combining the strengths of GP and RL, marrying the interpretability afforded by the GP with the generalizability and transferability of DRL algorithms. One of the downsides of RL is that it can be difficult to understand the factors that lead to the agents' actions, more so in a multi-agent setting. Thus, explicitly constructing a belief of the environment provides useful insight for diagnosing and refining the technique.

5.1.1 Related Works

Various studies have tackled the problem of collecting environmental information using multiple robots equipped with sensors. In [51], FEM was used for persistent environmental monitoring with multiple robots. In [70], multiple UAVs generated a target probability distribution map using GMM for target search. A more common approach than these methods is to use GP. GP is a non-parametric regressor based on training data, which is advantageous for inferring spatial information using a kernel. In [45, 46, 71], the sensor data gathered from each robot was processed with GP, and multiple robots performed tasks such as wind map building and ocean monitoring. In [36, 49], environmental modeling was performed with a mixture of GP combining GMM and GP.

However, it is not straightforward how to extend conventional GP to distributed settings since GP requires that all data be collected and processed on a single device such as a server. Thus, decentralized techniques have been proposed to apply GP to distributed robot networks [23, 39, 52]. In [23], a consensus algorithm was applied to a network of sensors to obtain an estimate of the entire map through local communications alone. In [52], a multi-robot system performed environmental learning in a distributed network based on the distributed GP and consensus algorithm.

Recent studies on multi-robot systems have integrated intelligent controllers with environmental learning algorithms to achieve higher autonomy. One such approach is RL which lies at the intersection of approximate dynamic programming and stochastic optimal control. Recent advances in data-driven techniques such as deep learning have extended the reach of RL to various domains [72–75]. DRL has been successfully applied to complex skills and high-dimensional inputs both in simulation and physical systems [76,77]. DRL may overcome the shortcomings of traditional methods in balancing the exploration-exploitation tradeoff of multi-agent target search and tracking mission by leveraging a diverse set of prior experiences to learn a generalized strategy that balances the tradeoff in an end-to-end manner.

Multi-agent multi-target capture has been considered in MARL domain [78]. However, such works mostly focus on the theoretical aspect and only consider grid-world examples which do not apply to robotic systems. My work resembles previous work which applied MARL to UAVs [79]. It defined the multi-UAV target tracking problem and applied an existing MARL algorithm under the multi-particle simulation environment. Unlike prior work, I consider a more realistic setting where target information is initially unknown and must be gathered from exploration. To do so, I propose a GP-based cooperative map building and demonstrate my method not just in simulation but with multi-UAV experiments as well.

5.1.2 Contributions

To achieve my objective, I focus on the following three main contributions:

- I build a belief for the target locations to circumvent the partial-observability arising from the unknown signal intensity field.
- I decentralize the map building process with distributed GP, which improves the scalability of the system by alleviating the computational and network burden.
- I modify an existing MARL algorithm to accommodate the belief map as an image input and facilitate policy transfer.

I perform various multi-agent target search and tracking simulations to evaluate the performance and transferability of my method and demonstrate on a swarm of micro-UAVs for experimental validation.

The outline of the chapter is as follows. Section 2 briefly describes the problem statement and preliminaries. Section 3 details a method for cooperative map building and map-based MARL algorithm to tackle multi-agent target search and tracking mission. Simulations and multi-UAV experiments for the various scenarios are presented in Section 4. Section 5 concludes the chapter.

5.2 Problem Statement and Preliminaries

5.2.1 Problem Statement

I focus on the target search and tracking problem with N multiple agents by considering a target domain as a 2-dimensional compact set $\mathbb{X}_c \subset \mathbb{R}^2$. There are M targets deployed in the target domain, which locations are unknown in advance. Each agent has a sensor that measures signal power, which is radiated in all directions from each target. Since this signal power is inversely proportional to the travel distance, it can be a clue to estimating the target position. *The main objective* is to locate all targets and position at least one agent for each target while avoiding agent-agent and agent-obstacle collision.

5.2.2 Multi-Agent Systems

I define the agent index set as $\mathcal{N} = \{1, 2, \cdots, N\}$ for N agents. The i-th agent's location at timestep k is represented as $\mathbf{x}_k^i \in \mathbb{X}_c$ ($i \in \mathcal{N}$). Each agent can communicate with neighbor agents via peer-to-peer communication. I define a set of neighbors for robot i at time k as $\mathcal{N}_k^i = \{j| \; \|\mathbf{x}_k^i - \mathbf{x}_k^j\| < d_{comm}, j \in \mathcal{N}/\{i\}\}$, where d_{comm} is the communication distance. $\mathcal{N}^{i+}$ is a shorthand for $\mathcal{N}^i \cup \{i\}$.

The target index set is represented as $\mathcal{M} = \{1, 2, \cdots, M\}$ for M multiple targets. Each target radiates a signal in all directions, and the signal intensity field $\varphi(\mathbf{x})$ at position $\mathbf{x}$ is defined by the sum of all signal intensities as follows:

$$\varphi(\mathbf{x}) = \sum_{m\in\mathcal{M}} A^m h(\|\mathbf{p}^m - \mathbf{x}\|), \quad \mathbf{x} \in \mathbb{X}_c, \tag{5.1}$$

where $A^m \in \mathbb{R}_{\geq 0}$ and $\mathbf{p}^m \in \mathbb{X}_c$ are the maximum signal intensity and positions of m-th target, respectively. $h(\cdot)$ represents a monotonic-decreasing intensity function where $h(d) \to 0$ as $d \to +\infty$.

Each agent has a signal power sensing unit that takes the measurement y_k^i of $\varphi(\cdot)$ at its current position $\mathbf{x}_k^i$ ($i \in \mathcal{N}$), which has the following relationship:

$$y_k^i \;=\; \varphi(\mathbf{x}_k^i) + n_k^i, \tag{5.2}$$

where the measurement of $\varphi(\mathbf{x}_k^i)$ is corrupted by the additive white Gaussian noise $n_k^i \sim \mathcal{N}(0, \sigma_n^2)$.

5.2.3 Gaussian Process for Target Search

With the signal power measurement in (5.2), GP is utilized to estimate target locations. GP is a data-driven non-parametric regressor, which can provide probabilistic inference over the set $\mathbb{X}_c$, taking into account joint probability distribution between the sampled dataset [20]. In (5.2), the unknown signal intensity field $\varphi(\cdot)$ is assumed to be represented as the following GP:

$$\varphi(\cdot) \sim \mathcal{GP}(0, \kappa(\mathbf{x}', \mathbf{x}'')), \tag{5.3}$$

where $\kappa(\mathbf{x}', \mathbf{x}'')$ is a *kernel* for positions $\mathbf{x}', \mathbf{x}'' \in \mathbb{X}_c$, and I use the original SE kernel which is defined as

$$\kappa(\mathbf{x}', \mathbf{x}'') = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}' - \mathbf{x}'')^\top \Sigma_l^{-1}(\mathbf{x}' - \mathbf{x}'')), \tag{5.4}$$

where σ_f^2 is the signal variance of $\varphi(\cdot)$, and Σ_l is the length-scale matrix. The hyper parameters σ_f^2 and Σ_l can be optimized by the maximum likelihood (ML) method [20].

When the agent i samples the data $\mathbf{x}_k^i$ and y_k^i at time k, these are stored sequentially in the GP input dataset $X_k^i = \{\bar{\mathbf{x}}_1^i, \cdots, \bar{\mathbf{x}}_{w_k^i}^i\}$ and the GP output dataset $\mathbf{y}_k^i = \{\bar{y}_1^i, \cdots, \bar{y}_{w_k^i}^i\}$, respectively.

w_k^i is the size of dataset at time k. For simplicity, I assume that w_k^i's are identical for all agents, such that $w_k^i = w$ hereafter. For any test position $\mathbf{x} \in \mathbb{X}_c$, the posterior distribution over $\varphi(\mathbf{x})$ by agent i is derived as follows:

$$p(\varphi(\mathbf{x})|X_k^i, \mathbf{y}_k^i, \mathbf{x}) \sim \mathcal{N}(\bar{\varphi}^i(\mathbf{x}), \Sigma^i(\mathbf{x})), \tag{5.5}$$

where,

$$\bar{\varphi}^i(\mathbf{x}) = \mathbf{k}^\top(X_k^i, \mathbf{x})(\mathbf{K}(X_k^i, X_k^i) + \sigma_n^2 I)^{-1}\mathbf{y}_k^i, \tag{5.6a}$$

$$\begin{aligned}\Sigma^i(\mathbf{x}) &= \kappa(\mathbf{x}, \mathbf{x}) \\ &\quad - \mathbf{k}^\top(X_k^i, \mathbf{x})(\mathbf{K}(X_k^i, X_k^i) + \sigma_n^2 I)^{-1}\mathbf{k}(X_k^i, \mathbf{x})\end{aligned} \tag{5.6b}$$

$\mathbf{K}(X_k^i, X_k^i)$ is $\mathbb{R}^{w\times w}$ matrix whose (u, v)-th element is $\kappa(\bar{\mathbf{x}}_u^i, \bar{\mathbf{x}}_v^i)$ for $\bar{\mathbf{x}}_u^i, \bar{\mathbf{x}}_v^i \in X_k^i$. $\mathbf{k}(X_k^i, \mathbf{x})$ is $\mathbb{R}^{w\times 1}$ column vector that is also obtained in the same way.

5.2.4 Deep Reinforcement Learning

RL is defined by a sequential decision making problem over MDP. MDP is a tuple of ($\mathcal{O}$, $\mathcal{A}$, $\mathcal{P}$, r), where $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability $P(o_{t+1}|o_t, a_t)$, and $r(o_t, a_t)$ is the immediate reward function. The goal of a RL agent is to find the optimal policy $\pi^*(a|o)$ that maximizes the expected γ-discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(o_t, a_t)]$. The discount factor $\gamma \in [0, 1]$ provides convergence guarantee for the Bellman operator and incentivizes the agent to take actions early.

With powerful function approximators, deep learning has pushed RL beyond the tabular setting. I opt for an off-policy, actor-critic variant of the DRL algorithm that can handle continuous observation and action spaces. Specifically, I opt for deep deterministic policy gradient (DDPG) algorithm [80] and its multi-agent variant MADDPG [81]. Off-policy methods reuse past experience from the replay buffer $\mathcal{D} = \{(o_t, a_t, r_t, o_{t+1})\}$ leading to better sample efficiency compared to their on-policy counterparts. Actor-critic methods consist of an actor which is a policy that maps observations to actions and a critic that evaluates the value of an observation-action pair under the given policy. The critic is updated via the temporal difference loss defined by the Bellman operator and the actor is updated by taking an ascending gradient on the critic.

5.3 Method

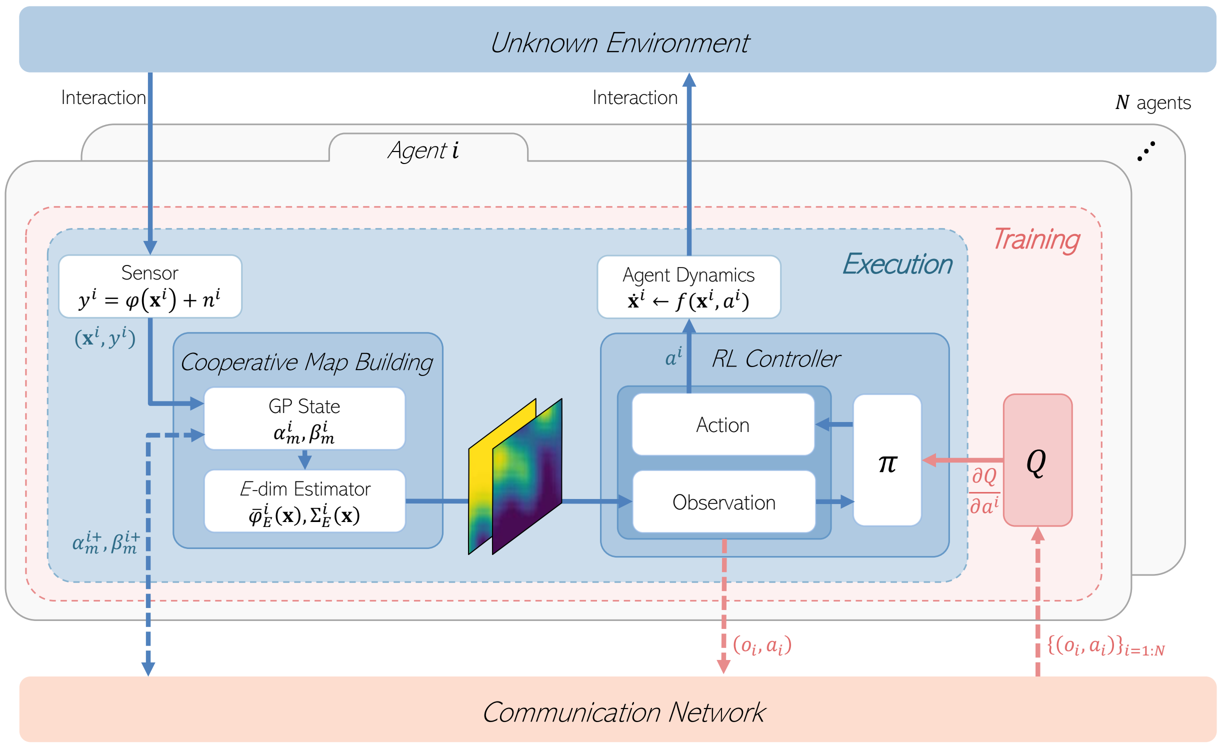


Figure 5.1: Schematics for target search and tracking with distributed GP and RL.

The overview of multi-agent target search and tracking with distributed GP and MADDPG is shown in Fig. 5.1. N agents, connected through a local communication network, operate while interacting with an environment with unknown targets. There are two main modules: *cooperative map building* and *RL controller*. Cooperative map building constructs a belief of the signal intensity map based on each agent's current position and sensor measurement. The actor network π inputs the observation including the generated GP map and agent information to output the corresponding action. This action is then applied to the system and the observation is transitioned according to the agent dynamics and interaction with the unknown target environment.

RL controller is trained in a centralized manner and executed in a decentralized manner. During training, critic network Q is evaluated with the observation-action pairs of all agents to update the actor network π. The reward function for the RL controller is a product of two sub-goal rewards, $r = r_{target} \times r_{collision}$. The sub-goal rewards are as follows:

$$r_{target} = \left[\prod_{m=1}^{M} 0.1 + 0.9 e^{-d^m / d_{char}} \right]^{\frac{1}{M}} \tag{5.7a}$$

$$r_{collision} = \begin{cases} 0, & \text{agent-agent/obstacle collision.} \\ 1, & \text{otherwise.} \end{cases} \tag{5.7b}$$

Both sub-goal rewards are bounded to the range $[0, 1]$. r_{target} is additionally normalized by the number of targets. d^m is the distance between m-th target and its corresponding optimally assigned agent, derived from the linear assignment problem [82]. d_{char} is the environment-dependent characteristic distance that is set to some appropriate value. $r_{collision}$ is a binary reward that heavily penalizes any agent-agent or agent-obstacle collision.

5.3.1 Multi-Agent Consensus-Based Map Building

Let the GP training dataset collected from all agents be $X_k = \cup_{i=1}^{N} X_k^i$ and $\mathbf{y}_k = \cup_{i=1}^{N} \mathbf{y}_k^i$. In the distributed multi-agent system, the conventional GP in Chapter 5.2.3 cannot incorporate X_k and $\mathbf{y}_k$ due to the lack of a central server. For this reason, *multi-agent consensus-based map building* is required.

The first step for consensus-based map building is to apply the E-dimensional approximation to the GP with Karhunen–Loève (KL) expansion [21]. This process is described in detail in [83]. According to E-dimensional approximation, the kernel (5.4) can be represented in terms of eigenfunctions ϕ_e and corresponding eigenvalues λ_e as $\kappa(\mathbf{x}', \mathbf{x}'') \approx \sum_{e=1}^{E} \lambda_e \phi_e(\mathbf{x}') \phi_e(\mathbf{x}'')$, and the E-dimensional estimator for (5.6a) on the training dataset X_k and $\mathbf{y}_k$ is expressed as follows [23]:

$$\bar{\varphi}_E(\mathbf{x}) := \Phi^\top(\mathbf{x}) F_E \mathbf{y}, \tag{5.8}$$

where,

$$\Phi(\mathbf{x}) := \Big[\phi_1(\mathbf{x}), \cdots, \phi_E(\mathbf{x})\Big]^\top, \tag{5.9a}$$

$$F_E := \left(\frac{G^\top G}{wN} + \frac{\sigma_n^2}{wN}\Lambda_E^{-1}\right)^{-1} \frac{G^\top}{wN}, \tag{5.9b}$$

$$G := \Big[\Phi(\bar{\mathbf{x}}_1^1)\cdots\Phi(\bar{\mathbf{x}}_w^1), \cdots, \Phi(\bar{\mathbf{x}}_1^N)\cdots\Phi(\bar{\mathbf{x}}_w^N)\Big]^\top. \tag{5.9c}$$

The second step for consensus-based map building is to rewrite (5.8) into a distributed form, since constructing G and $\mathbf{y}$ in (5.8) and (5.9) requires centralized processing. The associated terms included in (5.9b) can be decomposed as follows:

$$\frac{G^\top G}{Nw} = \frac{1}{N}\sum_{i=1}^{N}\alpha_w^i, \quad \frac{G^\top \mathbf{y}}{Nw} = \frac{1}{N}\sum_{i=1}^{N}\beta_w^i, \tag{5.10}$$

where $\alpha_w^i := \sum_{t=1}^{w}\Phi(\bar{\mathbf{x}}_t^i)\Phi^\top(\bar{\mathbf{x}}_t^i)/w$ and $\beta_w^i := \sum_{t=1}^{w}\Phi(\mathbf{x}_t^i)\bar{y}_t^i/w$ are *GP states* after the w-th sensor measurements.

As a result, (5.8) is modified to the following distributed form:

$$\bar{\varphi}_E^i(\mathbf{x}) \ := \ \Phi^\top(\mathbf{x})\left(\alpha_w^i + \frac{\sigma_n^2}{wN}\Lambda_E^{-1}\right)^{-1}\beta_w^i. \tag{5.11}$$

Using the average consensus algorithm [24] for the GP states, (5.11) converges to (5.8) through repeated communication. Similarly, the distributed form of $\Sigma(\mathbf{x})$ in (5.6b) is represented as follows:

$$\begin{aligned}\Sigma_E^i(\mathbf{x}) := \kappa(\mathbf{x},\mathbf{x}) - \Phi^\top(\mathbf{x})\left(\alpha_w^i + \frac{\sigma_n^2}{wN}\Lambda_E^{-1}\right)^{-1} \\ \times\alpha_w^i\Lambda_E\Phi(\mathbf{x}).\end{aligned} \tag{5.12}$$

When comparing distributed GPs with the conventional GPs in terms of (5.6), the computational complexity of the distributed algorithm is $O(E^3)$, which is less than that of the centralized algorithm, $O((wN)^3)$, since $E \ll wN$ holds in practice.

The final step in building the consensus map is to incorporate the newly acquired data into (5.11) and (5.12) to maintain the continuity of GP estimate. Data is collected as each agent navigates the environment. Applying the online information fusion algorithm of [83], the update rule

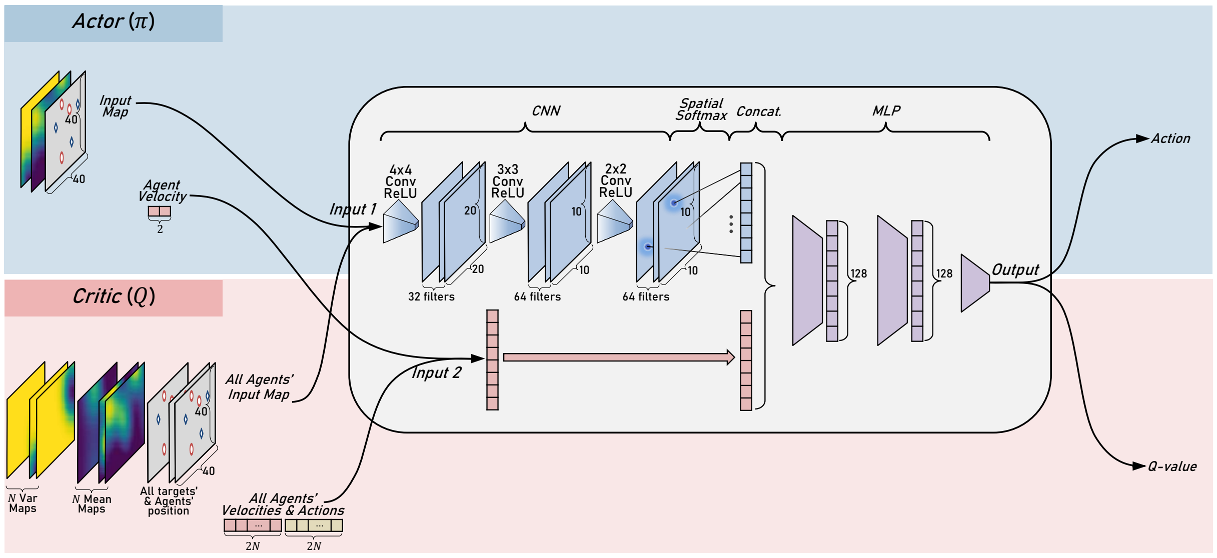


Figure 5.2: Network architecture for map-based MADDPG actor and critic.

of α_w^i and β_w^i is defined as

$$\begin{aligned} \alpha_{w+1}^i &= \tfrac{w}{w+1}\alpha_w^i + \tfrac{1}{w+1}\Phi(\bar{\mathbf{x}}_{w+1}^i)\Phi^\top(\bar{\mathbf{x}}_{w+1}^i), \\ \beta_{w+1}^i &= \tfrac{w}{w+1}\beta_w^i + \tfrac{1}{w+1}\Phi(\bar{\mathbf{x}}_{w+1}^i)y_{w+1}^i, \end{aligned} \tag{5.13}$$

where $\alpha_0^i = 0$ and $\beta_0^i = 0$. In the theorem 1, it has been proven that new data are successively merged with the existing $\{\alpha_w^i\}_{i=1}^N$ and $\{\beta_w^i\}_{i=1}^N$, such that $\{\alpha_{w+1}^i\}_{i=1}^N$ and $\{\beta_{w+1}^i\}_{i=1}^N$ converge towards (5.10) in a distributed manner.

5.3.2 Map-Based MADDPG

I choose MADDPG as the base RL algorithm for the multi-agent target search and tracking mission. MADDPG overcomes the non-stationary dynamics arising from the partial-observability of multi-agent settings by adopting centralized critics Q_{ψ_i} that inputs the combined set of observation and action spaces of all agents. MADDPG also proposes network ensembles for the decentralized actors π_{θ_i} to improve robustness and prevent the overfitting of policies with respect to other agents' actions which is especially problematic in competitive settings. To accommodate heterogeneous agents and individualized objectives, each agent i has a pair of centralized critic Q_{ψ_i} and decentralized actor π_{θ_i}. This configuration also allows for centralized training and decentralized deployment. During training, the critic is updated by minimizing the temporal difference loss, and the actor is updated by gradient ascent on the critic. Temporal difference loss for the centralized critic of the i-th agent parameterized by network weights ψ_i is as follows:

$$\mathcal{L}(\psi_i) = \mathbb{E}_{\mathcal{D}}[(Q^{\pi_i}_{\psi_i}(o^1_t, \cdots, o^N_t, a^1_t, \cdots, a^N_t) - l^i_t)^2], \tag{5.14}$$

where

$$l^i_t = r^i_t + \gamma Q^{\pi_i}_{\bar{\psi}_i}(o^1_{t+1}, \cdots, o^N_{t+1}, \pi_{\bar{\theta}_1}(o^1_{t+1}), \cdots, \pi_{\bar{\theta}_N}(o^N_{t+1})).$$

The target q-value l_i is evaluated with the target actor and target critic networks with delayed parameters $\bar{\theta}_i$ and $\bar{\psi}_i$. Policy gradient for the i-th agent decentralized actor parameterized by network weights θ_i is as follows:

$$\begin{aligned} &\nabla_{\theta_i}\mathcal{J}(\pi_{\theta_i}) \\ &= \mathbb{E}_{\mathcal{D}}[\nabla_{a^i_t} Q^{\pi_i}_{\psi_i}(o^1_t, \cdots, o^N_t, a^1_t, \cdots, a^N_t)|_{a^i_t=\pi_{\theta_i}(o^i_t)} \nabla_{\theta_i}\pi_{\theta_i}], \end{aligned} \tag{5.15}$$

where $\nabla_{a^i_t}$ denotes the action gradient of critic Q_{ψ_i} with respect to the i-th agent's action. Note that only the i-th action is generated by the actor while the rest of the actions are sampled from the replay buffer $\mathcal{D}$.

I adapt the standard MADDPG implementation for the multi-agent target search and tracking task. Since my approach incorporates GP map as the input for the RL agent, I use convolutional neural nets (CNNs) to extract features from image inputs which are then fed into the multi-layer perceptron (MLP) to form the actor and critic networks (Fig. 5.2). However, the combination of high-dimensional inputs with the multi-agent setting hinders RL. Specifically, it hinders representation learning which captures and condenses relevant information from the high-dimensional inputs. In practice, I make some modifications to aid representation learning. To reduce the number of network parameters, I adopt the spatial softmax module from [84]. For typical CNNs, the flatten layer contains most of the network parameters. Spatial softmax replaces the flatten layer with a channel-wise soft-argmax operation that extracts 2D positions, applying a strong bottleneck without introducing additional parameters. In addition, it has been known to be beneficial to match the modality of various inputs [85]. For multi-agent target search and tracking mission, the agent has access to 1D agent-centric information such as relative position of other agents and obstacles as well as the GP map (Fig. 5.2). Thus, I encode agent-centric information as an image and concatenate them alongside the GP mean and standard deviation channels forming a 3-channel image input for each agent.

The algorithm is further streamlined to improve computational efficiency. In the multi-agent target search and tracking mission considered in this chapter, agents are homogeneous and share the same objective (reward function). Thus, I can share a single centralized critic network for all agents. I also share a single actor network for all agents. However, unlike sharing a critic which inputs the combined observations and actions of all agents, sharing an actor requires additional justification. Assuming homogeneous agents and shared objective, the optimal action for any given observation is invariant to the choice of actor: $a^* = \pi^*_{\theta_1}(o) = \cdots = \pi^*_{\theta_N}(o), \forall o \in O$. In other words, the actors of different agents will converge to the same optimal policy in theory ($\pi^* = \pi^*_{\theta_1} = \cdots = \pi^*_{\theta_N}$). Thus, I can share a single actor for all agents without the loss of optimality. A beneficial side-effect of this approach is the transferability to a varying number of agents during deployment. In standard MADDPG, there are multiple actors and it is not apparent which actor to exclude or duplicate when the number of agents decreases or increases during deployment. A single shared actor alleviates this ambiguity. I also do not use network ensembles for the shared actor. Network ensembles for actors are important in competitive settings but less so in cooperative settings, and policy robustness can still be maintained without ensembles since a single shared actor is trained with experience from all agents.

5.4 Results

5.4.1 Hyperparameters

I conduct simulation and hardware experiments to evaluate the proposed method on multi-agent target search and tracking mission. Simulation environment is based on the OpenAI multi-agent particle environment [86] with the workspace of 2$\times$2. Radii of various entities such as the agent, target, obstacle were set to 0.05, 0.1, and 0.1. In addition, velocity and acceleration limit for the agents were set to 0.1 and 0.5, respectively. For signal intensities in (5.1), $A^m = 1$ was set for all $m \in \mathcal{M}$. The intensity function was set to $h(d) = \exp(-0.5d^2/0.06)$. Likewise, the hardware experiment setup was configured to replicate the simulation environment but with a larger workspace of 4m$\times$4m.

The following hyperparameters were used throughout the experiments unless otherwise specified. For GP kernel (5.4), I set $\sigma_f^2 = 1$ and $\Sigma_l = \text{diag}([0.05, 0.05])$. The dimension E of the approximated model was 40. A communication range of 2 was used. The training environment included 3 agents, 3 targets, and 2 obstacles (A3T3O2) with randomized location of various entities at the start of every episode (domain randomization). The discount factor of 0.95 and the maximum episode length of 100 were used. Actor and critic networks in Fig. 5.2 were configured with three convolutional layers of (number of filters, kernel size, stride) tuple (32,4,2), (64,3,2), and (64,2,1) followed by two fully-connected layers of hidden layer dimension 128. The spatial soft-argmax module was applied to the last convolutional layer to extract a 128-dimensional feature vector which was then concatenated with agent velocity (decentralized actor) or velocities and actions of all agents (centralized critic) to be fed into the fully-connected network. A batch size of 1024 and a learning rate of 0.001 were used to update the networks every 100 steps.

5.4.2 Multi-Sensor Consensus-Based Map Building

In order to better visualize the inner workings of the consensus-based map building module under the distributed network environment, it is applied to an example setting of a multi-sensor array. In

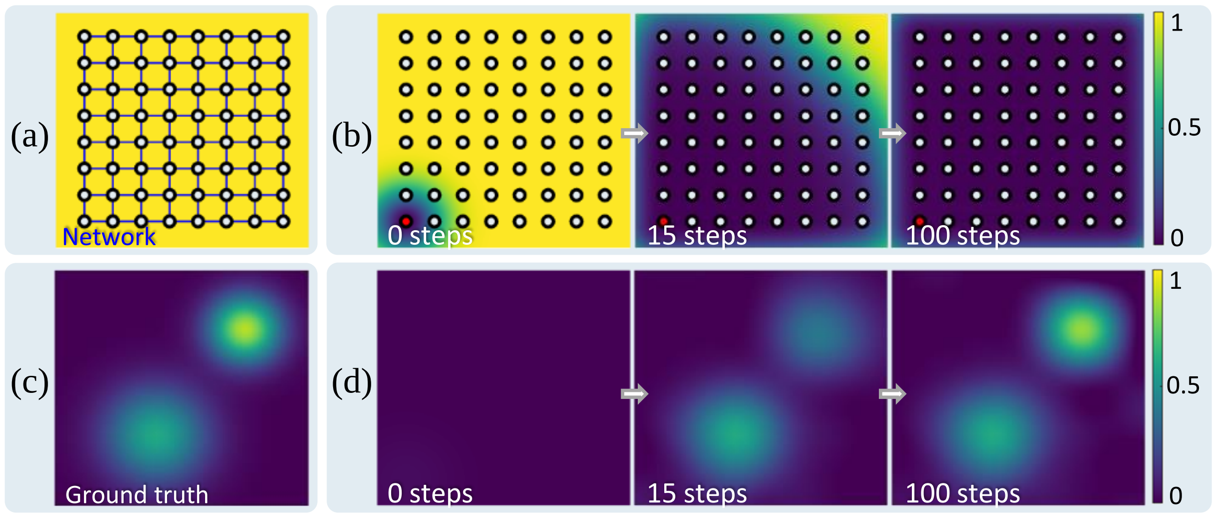


Figure 5.3: An example of consensus-based distributed map building for multi-sensor array.

Fig. 5.3-(a), there are 64 uniformly distributed sensors throughout the map. They are connected with neighbors via local communication and each sensor measures only once at the beginning. Relying solely on local information exchange and the consensus algorithm with neighboring sensors, map information is shared with the entire network. Fig. 5.3-(b) shows the result of information propagation from the perspective of the agent in the lower left (red dot). Fig. 5.3-(c) is the ground truth data, and Fig. 5.3-(d) represents the process of building the map over time. As a result of the average consensus algorithm, all sensors' GP maps converge to the same global map.

5.4.3 Map-Based MADDPG

To demonstrate the importance of representation learning in image-based MARL, I compare the learning curve of CNN with and without the spatial softmax module. To eliminate the exploration burden on the MADDPG agent and test the effect of the spatial softmax module on representation learning in isolation, I assume that the cooperative map building module can recover and provide the MADDPG module with the ground-truth map. Also, obstacles were removed from the training environment and the episode length was reduced for fast training and comparison.

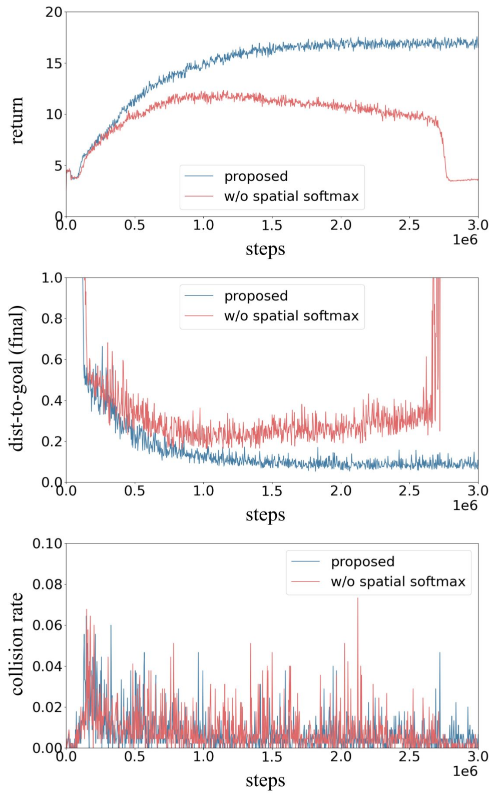


Figure 5.4: Network architecture ablation for map-based MADDPG.

Fig. 5.4 compares the episodic return, distance-to-goal at final timestep, and collision rate of the proposed network and naïve convolutional neural net for 3M training steps.

In terms of the episodic return and distance-to-goal, the proposed network was up to 50% more performant while the collision rate remained more or less identical for both cases. There was also a marked difference in learning stability as well, where the naïve CNN, in many training instances, has seen catastrophic failure occurs after convergence. This may be attributed to the overfitting of the actor due to a lack of network ensembles which is further compounded by the challenging representation learning under high-dimensional inputs.

5.4.4 Simulation Experiments

I evaluate the performance of the proposed algorithm (consensus-based map building + map-based MADDPG) in simulation and check the transferability of the learned actor to a varying number of entities at test time. I consider the following three cases: varying number of agents and targets ($A = T = n$), varying number of targets ($T = n$), and varying number of obstacles ($O = n$). The first scenario assumes that the number of targets with unknown location is known and the matching number of agents can be deployed. In the second scenario, even the number of targets is unknown and a fixed number of available agents are deployed, most likely resulting in a mismatch between the number of agents and targets. The last scenario represents the varying threat level of the test environment. Various performance metrics for the three scenarios are shown in Table 5.1. r_{avg} is the average episodic step reward, d_{final} is the distance-to-target at the final timestep and cr_{aa}, cr_{ao} is the agent-agent and agent-obstacle collision rate, respectively. Note that the metrics for each configuration are averaged over 10 episodes with randomized entity locations.

Table 5.1: Performance metric for various configurations.

config. *	r_{avg} †	d_{final} **	cr_{aa} ††	cr_{ao} ††
A3T3O2	0.3648	0.270	0.025	0.017
A1T1O2	0.3781	0.308	0.000	0.012
A2T2O2	0.3128	0.324	0.000	0.006
A4T4O2	0.2875	0.371	0.075	0.014
A5T5O2	0.2114	0.334	0.241	0.016
A3T1O2	0.3238	0.257	0.073	0.029
A3T2O2	0.3414	0.287	0.043	0.013
A3T4O2	0.3612	0.267	0.017	0.006
A3T5O2	0.4110	0.177	0.003	0.004
A3T3O1	0.3210	0.341	0.001	0.004
A3T3O3	0.3457	0.291	0.001	0.023

* blue font denotes deviation from the training environment

† average episodic step reward

** distance-to-target at the final timestep

†† agent-agent ('aa') or agent-obstacle ('ao') collision rate

In the first scenario, r_{avg} decreased as the number of agents and targets increased. Specifically, while performance was somewhat maintained for d_{final}, increasing rate of agent-agent collision negatively impacted r_{avg}. Since the map size has remained unchanged, increasing the number of agents increased the agent density, resulting in a more frequent collision between agents. For the special case of A1T1O2 with no agent-agent collision ($cr_{aa} = 0$), r_{avg} even outperformed the baseline. In the second scenario, the trends were reversed where the performance improved as the number of targets increased. Since the agent density remains a constant, r_{avg} was directly tied with contention between agents. With fewer targets than agents, two or more agents fighting for the same target led to a higher chance of collision between agents. Varying the number of obstacles (scenario 3) did not meaningfully impact the result. There was a slight increase in cr_{ao} as the number of obstacles increased but this did not have a meaningful impact on r_{avg}. I expected complete collision avoidance by penalizing obstacle collisions, however, temporary collisions were unavoidable. I think the reason is that agents learned to take a slight penalty in order to quickly approach the target.

5.4.5 Multi-UAV Setup

I demonstrate my algorithm with multi-UAV experiments. Fig. 5.8 represents the experiment setup for multi-agent target search and tracking mission. There are three targets (blue circle) and two obstacles (red cone) in a 4 m$\times$4 m workspace marked by black lines. Three Crazyflie 2.1 nano quadcopters measure the combined signal intensities of all targets. VICON motion capture system and onboard INSes were used for indoor localization. The entire system was implemented in ROS, and computations from the cooperative map building and RL controller modules were run on the laptop (Intel i7-7500U CPU) since Crazyflie 2.1 does not have an onboard microcomputer. The local communication links were implemented based on the distance between agents to mimic distributed control systems.

5.4.6 Multi-UAV Experiments

Hardware experiments were performed on various transfer scenarios: one baseline scenario, two scenarios with varying numbers of targets, and two scenarios with varying numbers of obstacles. These scenarios only consider a subset of transfer scenarios presented in the simulation experiments due to time and physical constraints. Hardware demonstration was only performed once for each scenario.

Each snapshot in Figs. 5.5-5.7 displays a captured footage and three supplementary plots (simplified top-down view, belief of the estimated target position denoted by the distributed GP mean and variance) at various times. Indigo indicates low values and yellow indicates high values in GP plots. The result of the baseline experiment is shown in Fig. 5.5. For the GP variance plot, the uncertainty about the target gradually decreases as agents explore. In the case of GP mean plot, it is initialized to zero at the beginning of the episode and when an agent discovers a target, the estimated target location turns yellow. In summary, the performance was maintained for the baseline environment of A3T3O2 despite the domain gap between simulation and hardware. Considering that there are subtle differences in system dynamics between simulation and the real world, the performance degradation was minimal.

The hardware demonstration for reducing or increasing the number of targets is shown in Fig. 5.6. Due to the domain gap, it took more time for the agent to converge to the target compared to the baseline scenario. When there were fewer targets than agents, two agents would converge to a single target or a surplus agent would keep searching for a non-existent third target. When there were more targets than agents, the agents would surround the targets or one agent would oscillate between two targets. In Fig. 5.7, the hardware demonstrations for reducing or increasing the number of obstacles are shown. In the case of one obstacle, it is less difficult than the baseline and the agent would quickly converge to the target. In the case of three obstacles, it is more difficult than the baseline and took more time to find the target. Overall, the trained agent could successfully zero-shot transfer to minor changes in the number of targets and obstacles.

5.5 Summary

This chapter addressed the challenge of multi-agent target search and tracking under an unknown environment. The decentralized map building module was proposed to construct a belief map and locate unknown targets with a distributed network of agents. The network architecture for the MARL agent was devised to accommodate high-dimensional inputs to incorporate the belief map with RL and allow for a zero-shot transfer during deployment. The proposed method was validated with simulated and hardware experiments in various scenarios and was able to search and track multiple targets under an unknown environment whilst avoiding obstacles. I also evaluated the zero-shot transferability of the method to a varying number of environment entities during deployment and demonstrated that performance is maintained. These results indicate that the proposed algorithm is applicable to a broader class of autonomous multi-agent target search and tracking operations and can be adapted for practical applications.

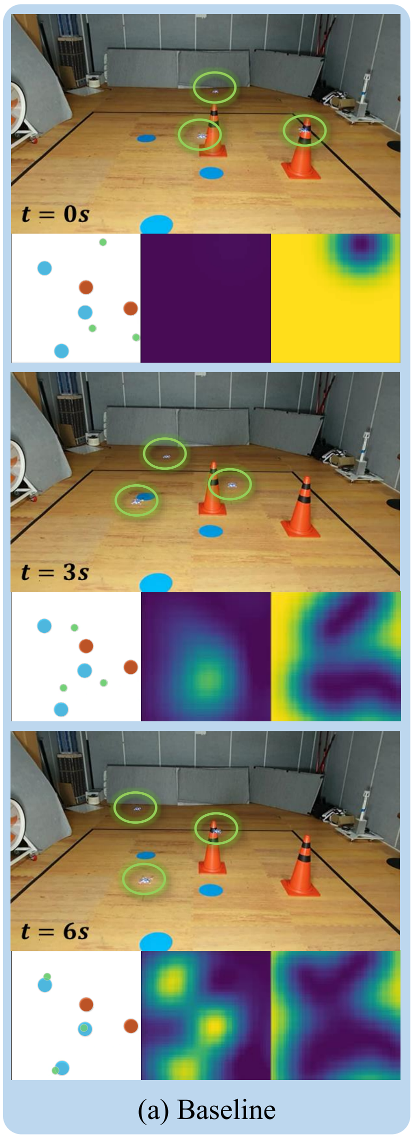


Figure 5.5: Multi-agent target search and tracking demonstration with micro-UAVs: each column represents a different zero-shot transfer setting with a varying number of targets and obstacles. From top to bottom, snapshots of the episode are arranged in chronological order. The three supplementary plots of each snapshot denote the top-down view, GP mean, and GP variance from left to right.

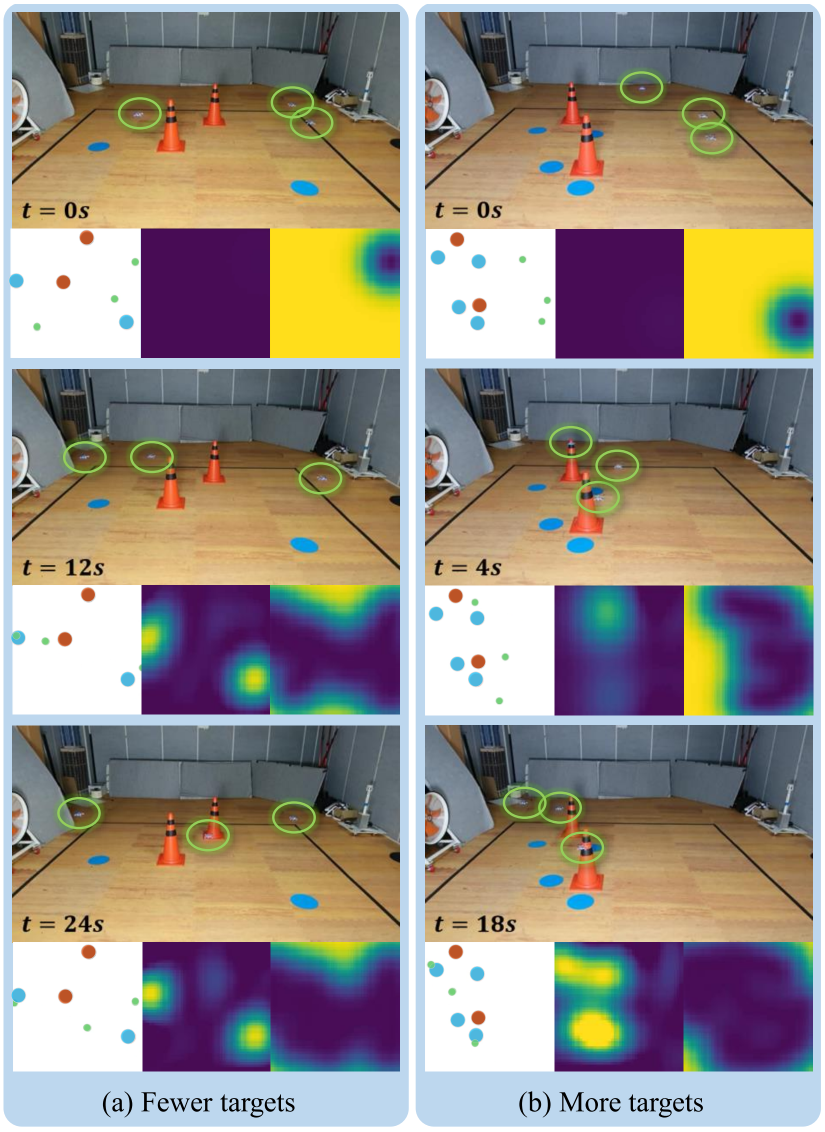


Figure 5.6: Multi-agent target search and tracking demonstration with micro-UAVs: each column represents a different zero-shot transfer setting with a varying number of targets and obstacles. From top to bottom, snapshots of the episode are arranged in chronological order. The three supplementary plots of each snapshot denote the top-down view, GP mean, and GP variance from left to right. Note that zero-shot transfer is successful for minor(±1) changes in the number of targets and the UAVs were able to find and converge to unknown targets.

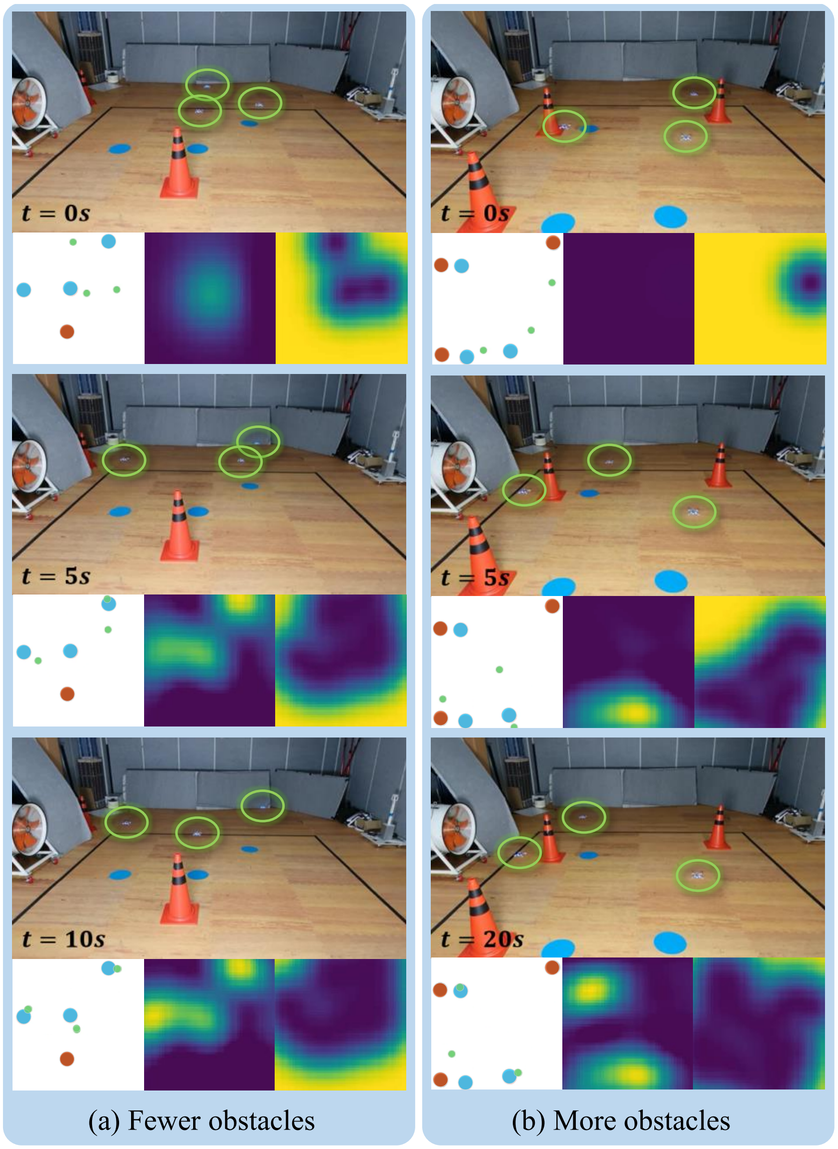


Figure 5.7: Multi-agent target search and tracking demonstration with micro-UAVs: each column represents a different zero-shot transfer setting with a varying number of targets and obstacles. From top to bottom, snapshots of the episode are arranged in chronological order. The three supplementary plots of each snapshot denote the top-down view, GP mean, and GP variance from left to right. Note that zero-shot transfer is successful for minor(± 1) changes in the number of obstacles and the UAVs were able to find and converge to unknown targets.

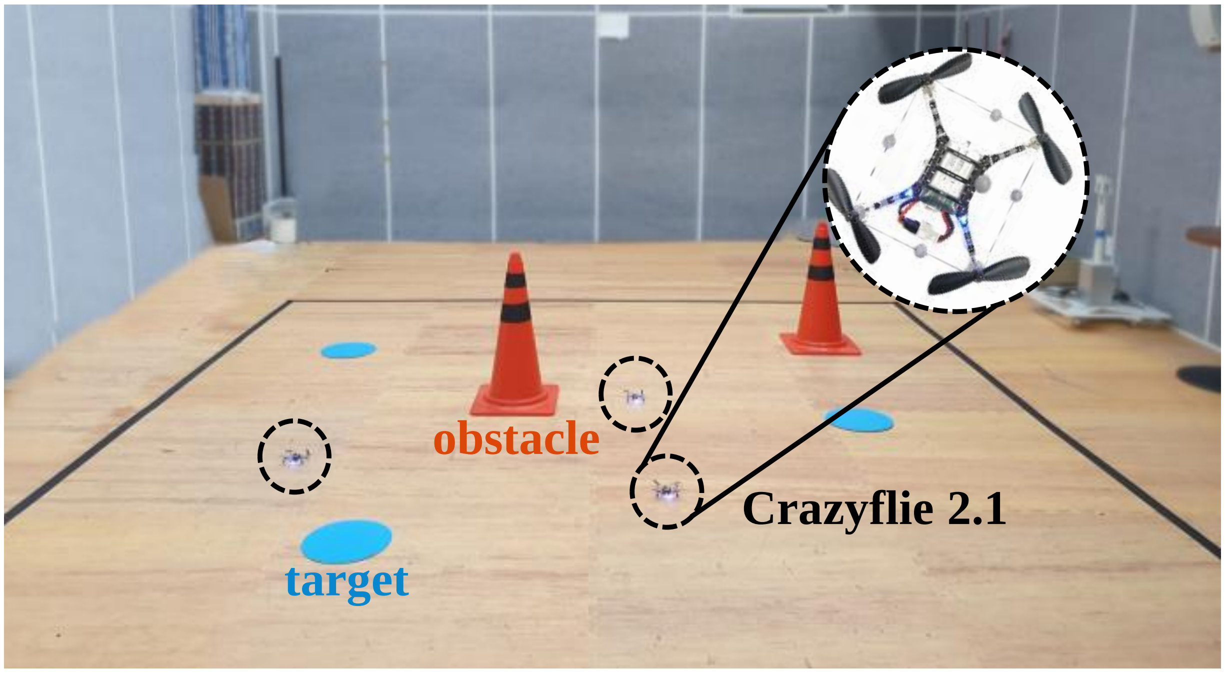


Figure 5.8: Multi-UAV experiment setup.

6 Conclusion

In the dissertation, I tackle the problem of dealing with distributed environmental learning and target search and tracking in multi-robot sysetms. Most systems using multiple robots inherently contain network problems due to limitations in communication distance and bandwidth. To overcome a harsh network environment, I utilize a distributed algorithm that works only with local communication. I apply the distributed Gaussian process in the learning of unkown environment. Using GP states and update model, newly sensed data is successfully fused to current GP estimates online. I have proposed the informative path planning algorithms for multiple robots to obtain additional sensory data useful to improve GP estimation quality. Extending these environmental learning problems, I have addressed the multi-target search and tracking problem. Using the signal intensity map learned by GP, I have designed a deep reinforcement learning technique that can control multiple agents to point to the targets' location. In the following, I present a concise summary of each chapter addressed in this dissertation.

Chapter 3: Multi-Robot Environmental Learning with Distributed Gaussian Process. I proposed a multi-robot environmental learning technique that can operate even in a distributed network. With the kernel expansion and average consensus algorithm, I implement online distributed GP in a networked control system. Using GP estimate, I establish a multi-agent exploration and exploitation algorithm with multi-robot coordination. All real-time environment learning results obtained by multiple robots converge to global estimation results, helping multiple robots to consistently perform cooperative tasks.

Chapter 4: Informative Path Planning for Multi-Robot Systems. I proposed a distributed search and path planning technique that works even in a distributed network. I combine the Gaussian process with the Monte Carlo tree search in a distributed manner for peer-to-peer communication. The Monte Carlo tree search technique combined with GP induces multiple robots to actively move to the path with the largest amount of information. My method allows multiple robots to collaboratively perform exploration, taking into account collision avoidance and coordination. Based on the informative path planning with a predictive trajectory of a certain distance, all robots have a non-myopic behavioral policy.

Chapter 5: Target Search and Tracking with Reinforcement Learning. To address the challenge of multi-agent target search and tracking under an unknown environment, I have combined the distributed GP with deep reinforcement learning. The network architecture accommodating high-dimensional inputs was devised to incorporate the belief map with RL and allow for a zero-shot transfer during deployment. The proposed technique enables multiple robots to actively search for and locate each target even in a distributed environment. In addition, the fact that it can be transferred to various environments and can immediately adapt to changes in the number of agents or targets is a differentiated advantage from existing multi-agent reinforcement learning algorithms.

A possible future study is to address the problem of search and tracking of moving targets. The distributed GP is also capable of learning about the changing environment, confirming that it is possible to track the location of moving targets. However, due to the limitations of the proposed deep reinforcement learning algorithm, policy learning that tracks a target based on a changing environment map is impossible. In order to overcome this problem, a technique for predicting positions of targets more precisely based on the Gaussian mixture probability hypothesis density (GM-PHD) filter is being studied. The predicted positions of targets can be used to leverage the policy learning for tracking moving targets.

A

Proof of Theorem 1

Proof. The GP states $\alpha_m^i(k)$ and $\beta_m^i(k)$ in (3.5) are state variables at time k. We assume the m-th training data $(\bar{\mathbf{x}}_m^i, \bar{y}_m^i)$ is obtained at time k_m, where $m \geq 1$. The E-dimensional estimator for GP (3.2) has to meet the following condition:

$$\begin{aligned}
\frac{G^\top G}{mn} &= \frac{1}{mn}\sum_{i=1}^{n}\sum_{t=1}^{m}\Phi(\bar{\mathbf{x}}_t^i)\Phi^\top(\bar{\mathbf{x}}_t^i) = \frac{1}{n}\sum_{i=1}^{n}\alpha_m^i(k_m), \\
\frac{G^\top \mathbf{y}}{mn} &= \frac{1}{mn}\sum_{i=1}^{n}\sum_{t=1}^{m}\Phi(\bar{\mathbf{x}}_t^i)\bar{y}_t^i = \frac{1}{n}\sum_{i=1}^{n}\beta_m^i(k_m).
\end{aligned} \tag{A.1}$$

When all agents obtain m-th measurement $\{\bar{y}_m^i\}_{i=1}^n$ at k_m, the left terms in (A.1) can be expressed as follows:

$$\begin{aligned}\frac{G^\top G}{mn} &= \frac{1}{mn}\sum_{i=1}^{n}\sum_{t=1}^{m}\Phi(\bar{\mathbf{x}}_t^i)\Phi^\top(\bar{\mathbf{x}}_t^i)\\ &= \frac{1}{mn}\sum_{i=1}^{n}(\sum_{t=1}^{m-1}\Phi(\bar{\mathbf{x}}_t^i)\Phi^\top(\bar{\mathbf{x}}_t^i)+\Phi(\bar{\mathbf{x}}_m^i)\Phi^\top(\bar{\mathbf{x}}_m^i))\\ &= \frac{1}{n}\sum_{i=1}^{n}(\tfrac{m-1}{m}\alpha_{m-1}^i(k_{m-1})+\tfrac{1}{m}\Phi(\bar{\mathbf{x}}_m^i)\Phi^\top(\bar{\mathbf{x}}_m^i))\\ &= \frac{1}{n}\sum_{i=1}^{n}\alpha_m^i(k_m),\end{aligned} \tag{A.2}$$

$$\begin{aligned}\frac{G^\top \mathbf{y}}{mn} &= \frac{1}{mn}\sum_{i=1}^{n}\sum_{t=1}^{m}\Phi(\bar{\mathbf{x}}_t^i)y_t^i\\ &= \frac{1}{mn}\sum_{i=1}^{n}(\sum_{t=1}^{m-1}\Phi(\bar{\mathbf{x}}_t^i)y_t^i+\Phi(\bar{\mathbf{x}}_m^i)y_m^i)\\ &= \frac{1}{n}\sum_{i=1}^{n}(\tfrac{m-1}{m}\beta_{m-1}^i(k_{m-1})+\tfrac{1}{m}\Phi(\bar{\mathbf{x}}_m^i)y_m^i)\\ &= \frac{1}{n}\sum_{i=1}^{n}\beta_m^i(k_m).\end{aligned} \tag{A.3}$$

Using the update rule (3.8), (A.1) is satisfied at all times $k \geq k_1$. With the help of average consensus algorithm, $\{\alpha_m^i(k)\}_{i=1}^n$ and $\{\beta_m^i(k)\}_{i=1}^n$ move towards their average value, i.e., $\lim_{k\to\infty}\alpha_m^i(k) = \sum_{i=1}^n \alpha_m^i(k_m)$ and $\lim_{k\to\infty}\beta_m^i(k) = \sum_{i=1}^n \beta_m^i(k_m)$. As a result, $\{\alpha_m^i(k_m)\}_{i=1}^n$ and $\{\beta_m^i(k_m)\}_{i=1}^n$ become equal to the leftmost terms in (A.1). $\square$

References

[1] D. S. Drew, “Multi-agent systems for search and rescue applications,” *Current Robotics Reports*, vol. 2, no. 2, pp. 189–200, 2021.

[2] A.-M. Zou, K. D. Kumar, and Z.-G. Hou, “Distributed consensus control for multi-agent systems using terminal sliding mode and chebyshev neural networks,” *International Journal of Robust and Nonlinear Control*, vol. 23, no. 3, pp. 334–357, 2013.

[3] A. Zidan, M. Khairalla, A. M. Abdrabou, T. Khalifa, K. Shaban, A. Abdrabou, R. El Shatshat, and A. M. Gaouda, “Fault detection, isolation, and service restoration in distribution systems: State-of-the-art and future trends,” *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2170–2185, 2016.

[4] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

[5] T. Arai, E. Pagello, L. E. Parker *et al.*, “Advances in multi-robot systems,” *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[6] A. Gautam and S. Mohan, “A review of research in multi-robot systems,” in *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*. IEEE, 2012, pp. 1–5.

[7] A. Serra-Gómez, B. Brito, H. Zhu, J. J. Chung, and J. Alonso-Mora, “With whom to communicate: learning efficient communication for multi-robot collision avoidance,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 11 770–11 776.

[8] J. Gregory, J. Fink, E. Stump, J. Twigg, J. Rogers, D. Baran, N. Fung, and S. Young, "Application of multi-robot systems to disaster-relief scenarios with limited communication," in *Field and Service Robotics*. Springer, 2016, pp. 639–653.

[9] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. John Wiley & Sons, 2010.

[10] G. A. Hollinger, S. Choudhary, P. Qarabaqi, C. Murphy, U. Mitra, G. S. Sukhatme, M. Stojanovic, H. Singh, and F. Hover, "Underwater data collection using robotic sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 899–911, 2012.

[11] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[12] Z. Lin, L. Wang, Z. Han, and M. Fu, "Distributed formation control of multi-agent systems using complex laplacian," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1765–1777, 2014.

[13] M. Jafarian, E. Vos, C. De Persis, A. J. Van Der Schaft, and J. M. Scherpen, "Formation control of a multi-agent system subject to coulomb friction," *Automatica*, vol. 61, pp. 253–262, 2015.

[14] N. Cao, K. H. Low, and J. M. Dolan, "Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms," *arXiv preprint arXiv:1302.0723*, 2013.

[15] K.-C. Ma, Z. Ma, L. Liu, and G. S. Sukhatme, "Multi-robot informative and adaptive planning for persistent environmental monitoring," in *Distributed Autonomous Robotic Systems*. Springer, 2018, pp. 285–298.

[16] A. Dutta, A. Ghosh, and O. P. Kreidl, "Multi-robot informative path planning with continuous connectivity constraints," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3245–3251.

[17] H. Long, Z. Qu, X. Fan, and S. Liu, "Distributed extended kalman filter based on consensus filter for wireless sensor network," in *Proceedings of the 10th World Congress on Intelligent Control and Automation*. IEEE, 2012, pp. 4315–4319.

[18] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 8179–8184.

[19] F. S. Cattivelli and A. H. Sayed, "Distributed nonlinear kalman filtering with applications to wireless localization," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 3522–3525.

[20] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[21] B. C. Levy, "Karhunen loeve expansion of gaussian processes," in *Principles of Signal Detection and Parameter Estimation*. Springer, 2008, pp. 1–47.

[22] H. Zhu, C. K. Williams, R. Rohwer, and M. Morciniec, "Gaussian regression and optimal finite dimensional linear models," 1997.

[23] G. Pillonetto, L. Schenato, and D. Varagnolo, "Distributed multi-agent gaussian regression via finite-dimensional approximations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2098–2111, 2018.

[24] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," 2003.

[25] A. Jadbabaie, J. Lin, and A. S. Morse, “Coordination of groups of mobile autonomous agents using nearest neighbor rules,” *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[26] G. Pajares, “Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs),” *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281–330, 2015.

[27] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, “Kalman filtering with intermittent observations,” *IEEE transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.

[28] V. Gupta, B. Hassibi, and R. M. Murray, “On sensor fusion in the presence of packet-dropping communication channels,” in *Proceedings of the 44th ieee conference on decision and control*. IEEE, 2005, pp. 3547–3552.

[29] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[30] N. Chen, Z. Qian, I. T. Nabney, and X. Meng, “Wind power forecasts using gaussian processes and numerical weather prediction,” *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 656–665, 2013.

[31] H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen, “Online sparse gaussian process regression using fitc and pitc approximations,” *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 703–708, 2015.

[32] D. Nguyen-Tuong, J. Peters, and M. Seeger, “Local gaussian process regression for real time online model learning,” *Advances in neural information processing systems*, vol. 21, 2008.

[33] Z. Jin and R. M. Murray, “Multi-hop relay protocols for fast consensus seeking,” in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1001–1006.

[34] S. Manfredi, “Design of a multi-hop dynamic consensus algorithm over wireless sensor networks,” *Control Engineering Practice*, vol. 21, no. 4, pp. 381–394, 2013.

[35] K.-C. Ma, L. Liu, and G. S. Sukhatme, “Informative planning and online learning with sparse gaussian processes,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4292–4298.

[36] W. Luo, C. Nam, G. Kantor, and K. Sycara, “Distributed environmental modeling and adaptive sampling for multi-robot sensor coverage,” in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 1488–1496.

[37] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, “Decentralized multi-agent exploration with online-learning of gaussian processes,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4222–4229.

[38] J. Choi, S. Oh, and R. Horowitz, “Distributed learning and cooperative control for multi-agent systems,” *Automatica*, vol. 45, no. 12, pp. 2802–2814, 2009.

[39] R. Allamraju and G. Chowdhary, “Communication efficient decentralized gaussian process fusion for multi-uas path planning,” in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 4442–4447.

[40] G. Flaspohler, V. Preston, A. P. Michel, Y. Girdhar, and N. Roy, “Information-guided robotic maximum seek-and-sample in partially observable continuous environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3782–3789, 2019.

[41] S. Ishii, W. Yoshida, and J. Yoshimoto, “Control of exploitation–exploration meta-parameter in reinforcement learning,” *Neural networks*, vol. 15, no. 4-6, pp. 665–687, 2002.

[42] F. Deng, S. Guan, X. Yue, X. Gu, J. Chen, J. Lv, and J. Li, “Energy-based sound source localization with low power consumption in wireless sensor networks,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4894–4902, 2017.

[43] M. Coombes, W.-H. Chen, and C. Liu, “Boustrophedon coverage path planning for uav aerial surveys in wind,” in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 1563–1571.

[44] Q. Feng, H. Cai, Y. Yang, J. Xu, M. Jiang, F. Li, X. Li, and C. Yan, “An experimental and numerical study on a multi-robot source localization method independent of airflow information in dynamic indoor environments,” *Sustainable Cities and Society*, vol. 53, p. 101897, 2020.

[45] J. Patrikar, B. G. Moon, and S. Scherer, “Wind and the city: Utilizing uav-based in-situ measurements for estimating urban wind fields,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1254–1260.

[46] K.-C. Ma, L. Liu, and G. S. Sukhatme, “An information-driven and disturbance-aware planning method for long-term ocean monitoring,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2102–2108.

[47] J. Zhang, R. Liu, K. Yin, Z. Wang, M. Gui, and S. Chen, “Intelligent collaborative localization among air-ground robots for industrial environment perception,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9673–9681, 2018.

[48] A. Quattrini Li, P. K. Penumarthi, J. Banfi, N. Basilico, J. M. O’Kane, I. Rekleitis, S. Nelakuditi, and F. Amigoni, “Multi-robot online sensing strategies for the construction of communication maps,” *Autonomous Robots*, vol. 44, no. 3, pp. 299–319, 2020.

[49] Y. Shi, N. Wang, J. Zheng, Y. Zhang, S. Yi, W. Luo, and K. Sycara, “Adaptive informative sampling with environment partitioning for heterogeneous multi-robot systems,” in *2020*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 11 718–11 723.

[50] D. Gu, “Distributed em algorithm for gaussian mixtures in sensor networks,” *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1154–1166, 2008.

[51] M. L. Elwin, R. A. Freeman, and K. M. Lynch, “Distributed environmental monitoring with finite element robots,” *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 380–398, 2019.

[52] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim, “Multi-robot active sensing and environmental model learning with distributed gaussian process,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5905–5912, 2020.

[53] A. Krause, A. Singh, and C. Guestrin, “Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies.” *Journal of Machine Learning Research*, vol. 9, no. 2, 2008.

[54] P. R. Silveira, D. d. F. Naiff, C. M. Pereira, and R. Schirru, “Reconstruction of radiation dose rate profiles by autonomous robot with active learning and gaussian process regression,” *Annals of Nuclear Energy*, vol. 112, pp. 876–886, 2018.

[55] A. Meliou, A. Krause, C. Guestrin, and J. M. Hellerstein, “Nonmyopic informative path planning in spatio-temporal models,” in *AAAI*, vol. 10, no. 4, 2007, pp. 16–7.

[56] L. Bottarelli, M. Bicego, J. Blum, and A. Farinelli, “Orienteering-based informative path planning for environmental monitoring,” *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 46–58, 2019.

[57] K. Yang, S. Keat Gan, and S. Sukkarieh, “A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav,” *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.

[58] W. Chen and L. Liu, “Pareto monte carlo tree search for multi-objective informative planning,” *arXiv preprint arXiv:2111.01825*, 2021.

[59] B. Du, K. Qian, C. Claudel, and D. Sun, “Parallelized active information gathering using multisensor network for environment monitoring,” *IEEE Transactions on Control Systems Technology*, 2021.

[60] B. Kartal, E. Nunes, J. Godoy, and M. Gini, “Monte carlo tree search for multi-robot task allocation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[61] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, “Dec-mcts: Decentralized planning for multi-robot active perception,” *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[62] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Information-theoretic regret bounds for gaussian process optimization in the bandit setting,” *IEEE transactions on information theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

[63] A. Garivier and E. Moulines, “On upper-confidence bound policies for switching bandit problems,” in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 174–188.

[64] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, “Lsar: Multi-uav collaboration for search and rescue missions,” *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019.

[65] H. Wang, C. Zhang, Y. Song, and B. Pang, “Master-followed multiple robots cooperation slam adapted to search and rescue environment,” *International Journal of Control, Automation and Systems*, vol. 16, no. 6, pp. 2593–2608, 2018.

[66] J. Lim, J. H. Park, and H. J. Kim, “Bayesian online learning for information-based multi-agent exploration with unknown radio signal distribution,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2621–2626, 2017.

[67] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[68] C. Yu, M. Zhang, F. Ren, and G. Tan, “Multiagent learning of coordination in loosely coupled multiagent systems,” *IEEE transactions on cybernetics*, vol. 45, no. 12, pp. 2853–2867, 2015.

[69] L. Zhou, P. Yang, C. Chen, and Y. Gao, “Multiagent reinforcement learning with sparse interactions by negotiation and knowledge transfer,” *IEEE transactions on cybernetics*, vol. 47, no. 5, pp. 1238–1250, 2016.

[70] P. Yao, H. Wang, and H. Ji, “Gaussian mixture model and receding horizon control for multiple uav search in complex environment,” *Nonlinear Dynamics*, vol. 88, no. 2, pp. 903–919, 2017.

[71] J. Binney, A. Krause, and G. S. Sukhatme, “Optimizing waypoints for monitoring spatiotemporal phenomena,” *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 873–888, 2013.

[72] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[73] Y. Sun and Y. Zhang, “Conversational recommender system,” in *The 41st international acm sigir conference on research & development in information retrieval*, 2018, pp. 235–244.

[74] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, “Deep direct reinforcement learning for financial signal representation and trading,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[75] S. Choi, S. Kim, and H. Jin Kim, “Inverse reinforcement learning control for trajectory tracking of a multirotor uav,” *International Journal of Control, Automation and Systems*, vol. 15, no. 4, pp. 1826–1834, 2017.

[76] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[77] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[78] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2681–2690.

[79] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," *IEEE access*, vol. 7, pp. 146 264–146 272, 2019.

[80] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proceedings of the 4th International Conference on Learning Representations*, 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[81] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[82] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[83] D. Jang, J. Yoo, C. Y. Son, and H. J. Kim, "Fully distributed informative planning for environmental learning with multi-robot systems," *arXiv preprint arXiv:2112.14433*, 2022.

[84] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[85] X.-H. Chen, S. Jiang, F. Xu, Z. Zhang, and Y. Yu, “Cross-modal domain adaptation for cost-efficient visual reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[86] I. Mordatch and P. Abbeel, “Emergence of grounded compositional language in multi-agent populations,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

국문초록

본 논문은 다중 로봇 시스템에서의 분산 환경 학습 기법을 다루며, 이를 다중 표적 탐색 및 추적 문제에 적용한다. 다중 로봇 시스템은 기본적으로 로봇의 협동 작업을 통해 신뢰성, 효율성 및 확장성의 장점을 얻는다. 이러한 다중 로봇 시스템은 데이터 기반 환경 학습에 적용하기 용이하다. 데이터 기반 환경 학습은 특정 관심 영역에서 다수의 센서 데이터를 획득하여 관심 영역에 대한 포괄적인 정보를 얻는 기법이다. 그러나 다중 로봇이 이러한 작업을 수행하기 위해서는 분산 학습 알고리즘과 협업 알고리즘이 필수적으로 요구된다.

본 논문은 첫째로 분산 환경 학습 알고리즘을 중점적으로 다룬다. 사전에 알려지지 않은 환경 과정에 대하여, 다중 로봇이 현재 위치에서 잡음이 포함된 센서 데이터를 획득한다면, 가우시안 프로세스 회귀 알고리즘을 통해 신뢰 구간을 바탕으로 공간 정보 지도를 구성할 수 있다. 그러나 기존의 가우시안 프로세스 알고리즘은 중앙 집중형으로 동작하기 때문에, 공간상에 넓게 분포한 다수의 센서로부터 오는 정보를 실시간으로 처리하기 어렵다. 이 논문에서는 다음과 같은 도전 과제를 해결할 수 있는 다중 로봇 탐색 알고리즘을 제안한다: i) 네트워크 센싱 플랫폼을 사용한 분산 환경 지도 구축, ii) 다중 로봇 팀에 적합하도록, 연속적인 센서 데이터 측정 및 융합을 사용한 온라인 학습, iii) 알려지지 않은 환경 과정의 최고점을 탐색하기 위한 다중 로봇 능동 감지 및 제어 기법. 이러한 알고리즘의 효율성을 다수의 UAV를 사용한 시뮬레이션과 지형 조사 실험을 통해 검증한다.

그러나 이러한 기법은 협력 탐색 과정에서의 경로 계획이 부재한 관계로, 로봇들의 근시안적인 행동을 초래한다. 따라서 다음 장에선 완전히 분산화 된 방식으로 동작하는 다중 로봇의 정보량 기반의 경로 계획 알고리즘을 제안한다. 이 알고리즘은 다음과 같은 도전 과제의 해결을 목표로 한다: i) 다중 로봇을 사용한 온라인 분산 환경 지도 학습, ii) 학습된 지도를 기반으로 안전하고 효율적인 탐색 경로 생성, iii) 로봇 수의 변화에 대한 확장성 유지. 이를 위해 전체 과정을 환경 학습과 경로 계획의 두 단계로 나눈다. 각 단계에 분산화된 알고리즘을 적용하고, 오직 인접 로봇 간의 통신을 통해 두 알고리즘을 결합한다. 환경 학습 알고리즘은 분산 가우시안 프로세스를 사용하고, 경로 계획 알고리즘은 분산 몬테카를로 트리 탐색을 이용한다. 그

결과, 로봇 수의 제약 없이 확장 가능한 다중 로봇 시스템을 구축할 수 있다. 시뮬레이션을 통해 제안된 시스템의 성능과 확장성을 보여주며, 또한 하드웨어 실험으로 보다 현실적인 시나리오에서 알고리즘의 유용성을 검증한다.

마지막으로, 앞서 설명한 환경 학습의 결과를 다중 목표물의 탐색 및 추적 문제에 적용할 수 있다. 목표물의 탐색 및 추적을 위해 여러 대의 로봇을 배치하는 것은 많은 연구에서 다뤄져 왔지만, 표적의 위치가 사전에 알려지지 않았거나 부분적으로 알려진 경우에서의 로봇 경로 계획 문제는 여전히 해결하기 어려운 문제이다. 그러나 최근에 떠오르고 있는 딥러닝과 강화 학습 같은 지능형 제어 기술을 이용하여, 에이전트는 사전 지식 없이 오직 환경과의 상호 작용을 통해 자율적으로 목표물을 탐색 및 추적할 수 있다. 이러한 방법은 데이터 기반 기법으로 동작함으로써, 경로 계획 문제에서의 탐색-활용 트레이드오프를 다룰 수 있으며, 기존 접근 방식의 전형적이고 실험적인 기법을 사용하지 않고도, 종단 간 학습을 통해 의사 결정 과정을 간소화할 수 있다. 이 논문에서는 분산 가우시안 프로세스를 기반으로 목표물 위치 지도를 구축하고, 이를 다중 에이전트 강화 학습에 적용하는 기법을 제안한다. 분산 가우시안 프로세스를 활용하여 대상 위치에 대한 신뢰 지도를 생성하고, 위치를 알 수 없는 목표물에 대해 효율적으로 탐색 및 추적을 할 수 있는 경로 계획법을 고안한다. 훈련된 정책의 성능과 새로운 환경으로의 전이 가능성을 시뮬레이션으로 평가하고, 다중 UAV를 활용한 하드웨어 실험을 통해 검증한다.

주요어**:** 다중 로봇 시스템, 환경 학습, 정보량 기반 경로 계획, 심층 강화 학습

학　번**:** 2019-35974