MULTI-ROBOT SYSTEM IN COVERAGE CONTROL: DEPLOYMENT, COVERAGE, AND RENDEZVOUS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Shaocheng Luo

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Byung-Cheol Min, Chair

Department of Computer and Information Technology

Dr. James Eric Dietz

Department of Computer and Information Technology

Dr. Shreyas Sundaram

School of Electrical and Computer Engineering

Dr. Xiumin Diao

School of Engineering Technology

Approved by:

Dr. Kathryne Newton

Associate Dean for Graduate Programs and Faculty Success

This dissertation is dedicated to my parents, for their endless love and support.

ACKNOWLEDGMENTS

I would like to take this special moment to express my sincere gratitude to my advisor, Professor Byung-Cheol Min. Without all his selfless support and encouragement, I would never have finished my degree. Because of his vision, I was able to get exposure to my current research area and conduct research in multi-robot systems. Because of his open mind, I have been able to participate in many lab projects and generate meaningful research questions based on the realistic challenges in robotic applications. Because of his support, I have been able to interact and work with scholars from all over the world. He has taught me a lot about conducting scientific research as well as living as an independent researcher. Undoubtedly, Professor Min is the best role model for me when taking an academic position.

I wish to gratefully acknowledge my committee members for their insightful comments and guidance. Thanks to Professors Eric Dietz, Shreyas Sundaram, and Xiumin Diao for their valuable comments and advice throughout my Ph.D. journey as well as this dissertation process. Their suggestions for writing scientific papers, searching for positions in academia, and preparing for interviews have been valuable.

My sincere thanks to my former advisor, Professor Shiqiang Zhu from Zhejiang University, for introducing me to the fascinating field of robotics. I also want to thank Professor Guity Ravai for her firm support while I was her teaching assistant. I will never forget the time that Professor Guity and I planned the new course for our department and made it a reality. She set a great example for me and demonstrated what a dedicated and passionate lecturer is like.

I will forever be thankful to all of the former and current members in the SMART Lab for willingly discussing my research topics and collaborating with me in conducting experiments. Without their expertise, my research might not have

come to fruition. Special thanks to my old friends and long-term lab mates, Jun Han Bae and Manoj Penmetcha, for their encouragement and help. I wish Jun Han and Manoj, along with all my lab members, great success in the future.

Last but not least, I would like to express my gratitude to my parents for their selfless support as I pursue my own dream. Without their understanding, I would not have traveled to the US and completed my Ph.D. program. Here, I also want to mention my girlfriend, Huifang Ma. I would never be looking forward to my future life like this before meeting her.

To those people who love me and people whom I love, you are the reason why I strive forward.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABBREVIATIONS

ABSC	Asymptotic Boundary Shrink Control
AP	Access Point
APF	Artificial Potential Field
API	Application Programming Interface
AUV	Autonomous Underwater Vehicle
BFS	Breadth-first Search
CCW	Counterclockwise
CPS	Cyber-Physical Systems
CVT	Centroidal Voronoi Tessellation
DNN	Deep Neural Network
DOA	Direction of Arrival
FOT	Field of Tracking
FOV	Field of View
GPS	Global Positioning System
GT	Geometric Tessellation
ILP	Integer Linear Programming
IMU	Inertial Measurement Unit
LOS	Line-of-Sight
ML	Machine Learning
MRS	Multi-Robot Systems
NMS	Soft Non-maximum Suppression
P3AT	Pioneer3-AT Robot
PRBC	Pivot-robot Based Coverage
R-CNN	Region-based Convolutional Neural Network

R-FCN	Region-based Fully Convolutional Network
RL	Reinforcement Learning
ROS	Robot Operating System
RSSI	Radio Signal Strength Index
RVO	Reciprocal Velocity Obstacle
SOM	Self-Organizing Map
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USAR	Urban Search and Rescue
USV	Unmanned (Water) Surface Vehicle
WCA	Weighted Centroid Algorithm
WSN	Wireless Sensor Network

ABSTRACT

Luo, Shaocheng Ph.D., Purdue University, May 2020. Multi-robot System in Coverage Control: Deployment, Coverage, and Rendezvous. Major Professor: Byung-Cheol Min.

Multi-robot systems have demonstrated strong capability in handling environmental operations. In this study, We examine how a team of robots can be utilized in covering and removing spill patches in a dynamic environment by executing three consecutive stages: deployment, coverage, and rendezvous.

For the deployment problem, we aim for robot allocation based on the discreteness of the patches that need to be covered. With the deep neural network (DNN) based spill detector and remote sensing facilities such as drones with vision sensors and satellites, we are able to obtain the spill distribution in the workspace. Then, we formulate the allocation problem in a general optimization form and provide solutions using an integer linear programming (ILP) solver under several realistic constraints. After the allocation process is completed and the robot team is divided according to the number of spills, we deploy robots to their computed optimal goal positions. In the robot deployment part, control laws based on artificial potential field (APF) method are proposed and practiced on robots with a common unicycle model.

For the coverage control problem, we show two strategies that are tailored for a wirelessly networked robot team. We propose strategies for coverage with and without path planning, depending on the availability of global information. Specifically, in terms of coverage with path planning, we partition the workspace from the aerial image into pieces and let each robot take care of one of the pieces. However, path-planning-based coverage relies on GPS signals or other external

positioning systems, which are not applicable for indoor or GPS-denied circumstances. Therefore, we propose an asymptotic boundary shrink control that enables a collective coverage operation with the robot team. Such a strategy does not require a planned path, and because of its distributedness, it shows many advantages, including system scalability, dynamic spill adaptability, and collision avoidance. In case of a large-scale patch that poses challenges to robot connectivity maintenance during the operation, we propose a pivot-robot coverage strategy by mean of an a priori geometric tessellation (GT). In the pivot-robot-based coverage strategy, a team of robots is sent to perform complete coverage to every packing area of GT in sequence. Ultimately, the entire spill in the workspace can be covered and removed.

For the rendezvous problem, we investigate the use of graph theory and propose control strategies based on network topology to motivate robots to meet at a designated or the optimal location. The rendezvous control strategies show a strong robustness to some common failures, such as mobility failure and communication failure. To expedite the rendezvous process and enable herding control in a distributed way, we propose a multi-robot multi-point rendezvous control strategy.

To verify the validity of the proposed strategies, we carry out simulations in the Robotarium MATLAB platform, which is an open source swarm robotics experiment testbed, and conduct real experiments involving multiple mobile robots.

CHAPTER 1. INTRODUCTION

Multi-robot systems (MRS) have significant potential in applications such as cooperative exploration, urban search and rescue (USAR), and monitoring and securing of large complex environments (Burgard, Moors, Fox, Simmons, & Thrun, 2000; Dunbabin & Marques, 2012; Liu & Nejat, 2013). Recent research advances in ad hoc networks have led to the development of networked robots and facilitated multi-robot utilization for a variety of coordination, e.g., (Min, Lewis, Matson, & Smith, 2013; Min, Matson, & Jung, 2016; Min, Parasuraman, Lee, Jung, & Matson, 2018).

Among all the active fields of research, the coverage control problem is of great interest and significance. Robot coverage control means that robots spread out over an environment while aggregating in areas of high sensory interest (Schwager, Rus, & Slotine, 2009), or that robots move and visit every region of the area to be covered (Rekleitis, New, Rankin, & Choset, 2008). Recently, MRS have been widely introduced as a solution to coverage problems that are characterized by being large-scale; urgent; labor-intensive; and prohibitive to human operators due to toxic vapors, smoke, and chemicals. For example, a multi-robot team is used in cleaning up hazardous spills such as oil (Jin & Ray, 2014b) and in marine surveys that are crucial for assessing the effects of maritime disasters (Mukhopadhyay, Wang, Patterson, Malisoff, & Zhang, 2014). In the U.S., more than a hundred oil and chemical spills are detected in aquatic areas every year, which severely threatens natural resources and public health. Moreover, these spills substantially disrupt water transportation and cause tremendous ecological, economic, and social impacts. For instance, the most recent incident was BP's Deepwater Horizon rig oil spill in the Gulf of Mexico, which spilled more than 210 million gallons and affected an area of 180,000 km^2; 11 people and hundreds of birds and marine organisms were

killed in this disaster (Smith, Smith, & Ashcroft, 2011). Development of advanced technologies for oil spill cleaning, especially a collective *coverage* control using multi-robot systems, is necessary to build a timely and fiscally prudent fast response system (N. M. P. Kakalis & Ventikos, 2008).

Furthermore, MRS can help fire surveillance operations in natural environments (Casbeer, Kingston, Beard, & McLain, 2006; X. Zheng, Koenig, Kempe, & Jain, 2010), and even fire extinguishing missions (Viguria, Maza, & Ollero, 2010). MRS coverage control is also popular in household scenarios, such as lawn mowing and room cleaning (Kong, Peng, & Rekleitis, 2006), and can be used for military purposes such as automated humanitarian de-mining (Rekleitis, New, & Choset, 2005).

Before performing coverage control, we need to decide the optimal locations that the robots can be deployed and start working from. This step is called robot *deployment* (Schwager, Rus, & Slotine, 2011) and is especially significant when the areas to be covered in the workspace are discrete. We consider this issue as a robot deployment problem, which consists of optical allocation and robot navigation to the goal positions.

When the coverage operation reaches an end, we want to recall the robots to a station for further deployment or maintenance (Dimarogonas & Kyriakopoulos, 2007). This behavior, in which all or many of the robots meet at a location, is called *rendezvous*. Rendezvous also happens when robots are docking or meet a charging robot for recharging (Mathew, Smith, & Waslander, 2015).

In this dissertation, we propose a systematic coverage control solution with MRS, focusing on the aforementioned three stages, and present validation processes.

1.1 Motivation and Goals

The motivation for this study on the multi-robot coverage control problem is to propose an efficient strategy that coordinates the networked robots to cover the

patches[1] in the workspace. For instance, where a number of spills are distributed in an aquatic environment and waiting to be removed, a team of robots will be deployed to take care of every spill and perform the cleanup operation. After the coverage operation is completed, the robots will be gathered at a place or will meet at the designated location.

The deployment problem is motivated by robot allocation based on the tasks and motion controllers that drive each robot to the goal. The deployment problem is essential for centralized control cases. Once a global map is available for every robot, an optimal allocation of robots according to the distribution of tasks is desired. The allocation result should consider an evenly distributed workload for each robot and try to minimize the overall traveling distance during the deployment. At the same time, the challenges of collision avoidance and deadlock in robot maneuvering must be circumvented in the deployment control.

The study on the collective coverage control is motivated by the aforementioned oil spill coverage operation, in which many robots are utilized to clean up the spills. There are two situations in coverage control: First, if a global map of the workspace is available for all the robots or the coverage task is persistent, we can plan an efficient path for the robot to travel and perform coverage. We can name this solution the centralized coverage control. Second, in order to deal with a dynamic spill, or if a global map is not always available, distributed coverage control is needed. In the distributed coverage control, the robots have to cope with moving spills resulting from external disturbances. Moreover, practical constraints such as limited wireless communication range and vision sensing range have to be fully considered in devising the control strategy. Finally, the robots have to avoid inter-robot collisions in moving. We research both centralized and distributed coverage using a multi-robot team.

The motivation for the rendezvous control is that when robots have completed their tasks or have to recharge themselves at a certain time, they need to

[1]Throughout this dissertation, the terms “patch,” “spill patch,” and “spill” are interchangeable and refer to the spill patch that needs to be covered.

maneuver to the rendezvous point distributedly by means of network connection topology. An appropriate rendezvous control law will enable robot rendezvous efficiently while avoiding obstacles and circumventing a variety of failures. Since the distribution of the robots may not be known at the time that rendezvous control initiates, and one robot can hardly connect to all the other robots in the workspace, a distributed rendezvous control is indispensable and will be discussed thoroughly in this dissertation.

The overall goal of this study is threefold. First, we will use optimization to determine the optimal allocation for the robots in the workspace depending on the distribution of the spills, and provide motion control laws to navigate the robots to their goal locations free of collision or deadlock. Second, we propose two coordination control strategies, centralized and distributed, to enable robots to smoothly cover and remove the spills. Last, we develop hierarchical rendezvous algorithms that realize rendezvous for single or multiple rendezvous points using network connection topology.

1.2 Problem Statements

In this dissertation, we seek to address the problems associated with multi-robot coverage: deployment, coverage control, and rendezvous. Each of these problems is described as follows.

1. **Multi-robot deployment, presented in Chapter 3**
   - **Input:** One or multiple raw aerial images from remote vision sensing that show the workspace containing spills as well as the initial distribution of robots.
   - **Output:** The spills differentiated from the background such as water, allocation of the robots based on the distribution of the spills, and deployment control that drives the robots to their goals.

- **Goals:** i) The spill/cluster has to be identified using image processing techniques from the provided aerial images captured by remote sensing facilities. The spill has to be differentiated from the background such as water by showing a boundary. ii) Assuming that the number of robots is greater than the number of patches, the robot team has to be allocated optimally to each spill/patch. iii) A robot motion control scheme for deployment should be developed.
- **Interpretation:** In this multi-robot deployment problem, in addition to spill identification using image processing, two fundamental problems need to be solved - *Allocation* and *Motion Control*. In the allocation problem, every spill in the workspace has to be taken care of by at least one robot. Moreover, we want to achieve a balanced workload for every deployed robot. This can be realized by letting the number of robots associated with a specific spill be in proportion to the area of the spill. Therefore, a uniform density of robot is realized among all spills. As each robot has the same cleaning capability, a uniform density of robots implies a uniform completion time for spill cleaning. This makes the completion time predictable, thus facilitating robot recall after the work is completed. Once the associated spill is determined for a robot, the robot has to maneuver to the goal position on the boundary of the spill using the proposed motion controller.

2. **Multi-robot coverage, presented in Chapter 4**

   - **Input:** Multiple discrete spills along with mobile robots that are allocated to them.
   - **Output:** Coordination control strategy that enables robots to perform complete coverage of the spills.
   - **Goals:** i) Robots maneuver without collision. ii) Robots can be controlled to search and track the boundary of patches. iii) The

coordinate control strategy devised has to be tailored for the specific problem setting.

- **Interpretation:** The designed coverage control strategy has to consider the environment setting where the strategy will be applied. If the workspace information can be shared by all the robots involved and a positioning system is accessible, a centralized coverage control strategy can be realized with an optimal path planned for the robot. In contrast, if only local information is accessible for each robot, a distributed coverage control strategy is indispensable. The distributed control needs to show features including scalability, collision avoidance, dynamic spill adaptability, and final convergence. When a robot executes coverage operation, it will follow the coverage manifold that we proposed based on the robot prototype. The coverage can be done either with or without a planned path. Coverage control in a distributed manner implies that the robot team will not rely on path planning and external localization systems. The system scalability of distributed control allows many robots employed in the task to increase efficiency. Collision can interrupt robot motion and halt the coverage operation. Thus, in a collective cleanup operation, the motion controller has to motivate robots to maneuver but prevent collisions. We develop a strategy, based on spill boundary searching and tracking, that can even deal with a moving spill. The final convergence means that the robot team gathers at a common location after the coverage operation.

3. **Multi-robot rendezvous, presented in Chapter 5**

   - **Input:** The network connectivity of robots represented by graph theory.
   - **Output:** The robot team gathers at one or a certain number of points.
   - **Goals:** i) The robots utilize the limited connection to gather at the rendezvous point. ii) Robots complete rendezvous in the shortest path,

avoid obstacles, and circumvent failures such as loss of mobility and/or wireless connection. iii) The coordinate control strategy is distributed and allows scalability of the system.

- **Interpretation:** Assuming that the networked robots are initially connected, we will utilize such connections as well as radio signal strength of wireless signals to realize rendezvous. Considering limitations in reality, such as unknown and cluttered environments, limited communication ranges, and failures in mobility and communication, our proposed single-point and multi-point rendezvous control strategies should be deliberately designed and adaptive to those challenges.

1.3 Contributions

We expect the contributions of this study to cover the following aspects.

1. **Multi-robot Coverage Control.** The major contribution of this work is the three well-structured problem statements that constitute a systematic multi-robot coverage solution.

2. **Theoretical Analysis to the Solution** The research problems are formulated mathematically, and the stability and convergence of the proposed solutions are also strictly proved.

3. **Simulation and Real Experiment Validation.** The last, but not least, contribution of this work is that the control strategies are validated through intensive experiments in both simulations and field tests. A multi-robot platform with networked robots is utilized to conduct real experiments.

1.4 Dissertation Outline

This dissertation consists of six chapters, including this introductory chapter. Each chapter is self-contained and can be read independently. The rest of the dissertation is organized as follows.

Chapter 2 presents the literature review of the related work and elaborates the state-of-the-art progress related to the three research questions.

Chapter 3 presents the problem of multi-robot deployment control. Assuming that a DNN-based spill detector along with vision sensors can be used to localize spills such as algae blooms, we propose an optimization-based robot allocation strategy for the bounding boxes generated from the detector with real images. Moreover, more complicated shapes are tested to validate the allocation strategy. We also present methods to extract the spills from their background using image segmentation techniques. This helps the execution of coverage control in the following chapter. The robot trajectory in the deployment process is also presented.

Chapter 4 elaborates first the coverage manifold model and then the coverage control strategy after deployment. Based on the situation of the workspace, we propose different strategies for coverage, i.e., planning-based centralized control and distributed control. We propose a distributed coverage control strategy called asymptotic boundary shrink control (ABSC) without path planning. Eventually, the robot controllers for every strategy are validated through a set of simulations.

Chapter 5 describes multi-point rendezvous control after introducing the networking topology represented by a connected graph. The control strategy is verified via both simulation experiments and field tests with networked unmanned ground vehicles (UGVs).

Finally, Chapter 6 summarizes the current work and presents some thoughts for the future.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

This chapter provides a review of the literature relevant to the proposed three research problems, i.e., multi-robot deployment, multi-robot coverage, and multi-robot rendezvous. We will have three sections to review these three problems, and have the last section for the research at their intersection.

2.1 Multi-robot Deployment

In this section, we do a literature review of multi-robot deployment control. We discuss the deployment problem from three perspectives: workspace remote sensing, robot allocation (team partition), and motion control with obstacle avoidance.

2.1.1 Sensing Over Workspace and Spill Detection

Vision Sensing and *In-situ* Image Processing

The overall observation of the workspace and spill identification is a significant part of this dissertation. We assume that an aerial image can be obtained by using computer vision-based remote sensing techniques. Examples are UAVs with vision sensors (Morgenthal & Hallermann, 2014), human-operated helicopters with imaging devices (Pyo et al., 2018), satellite-based remote sensing (Câmara et al., 1996), etc.

Computer vision-based algae monitoring systems have been developed that exploit the distinctive green or greenish-blue color characteristic of spills such as algae blooms. However, such traditional computer vision pipelines do not have high repeatability; they depend significantly on the effectiveness of their feature detectors

or their segmentation procedures. These can be rendered ineffective by environmental conditions such as fluctuating illumination, occlusion, or the presence of comparable objects in the background.

Admittedly, outdoor vision sensing is tricky and subject to light conditions. But a variety of sensors and sensing technologies have been developed in order to circumvent the previously mentioned challenges in vision sensing (Honegger, Meier, Tanskanen, & Pollefeys, 2013; Narasimhan & Nayar, 2003; Um, Ryu, & Kal, 2011). Furthermore, hyperspectral imagers, which provide a full spectrum for each pixel in an image, recording data in dozens or hundreds of different narrow wavelength bands, are used to identify algae bloom spills (Pyo et al., 2018). We believe such vision-sensing technology can definitely be applied to identify many other spills.

Based on these raw aerial images, we propose a DNN-based detector for spills in the workspace (Penmetcha et al., 2019), which is depicted in Figure 2.1.

We choose algae bloom as the target scenario to build a DNN-based spill detector. We first collected a dataset of images taken from ground and aerial vehicles depicting algae in pools, lakes, ponds, etc. These images were collected using a variety of resources, and priority was given to generating a diverse dataset. We collected some of the images ourselves, obtained some from online resources, and generated some through artificial simulations. We then utilized the Tensorflow Object Detection API to train our chosen models to detect algae (Abadi et al., 2016). The Tensorflow Object Detection API is an open source framework based on the Tensorflow library that provides a well-structured environment for developing, training, testing, and deploying deep learning models.

One issue that we observed in the spill detection results is that the algorithms Region-based Convolutional Neural Network (R-CNN) and Region-based Fully Convolutional Network (R-FCN) produced multiple detections on the same algae spill. This is a consequence of the underlying workings of neural networks. However, in order to cope with our proposed resource allocation algorithm, an algae

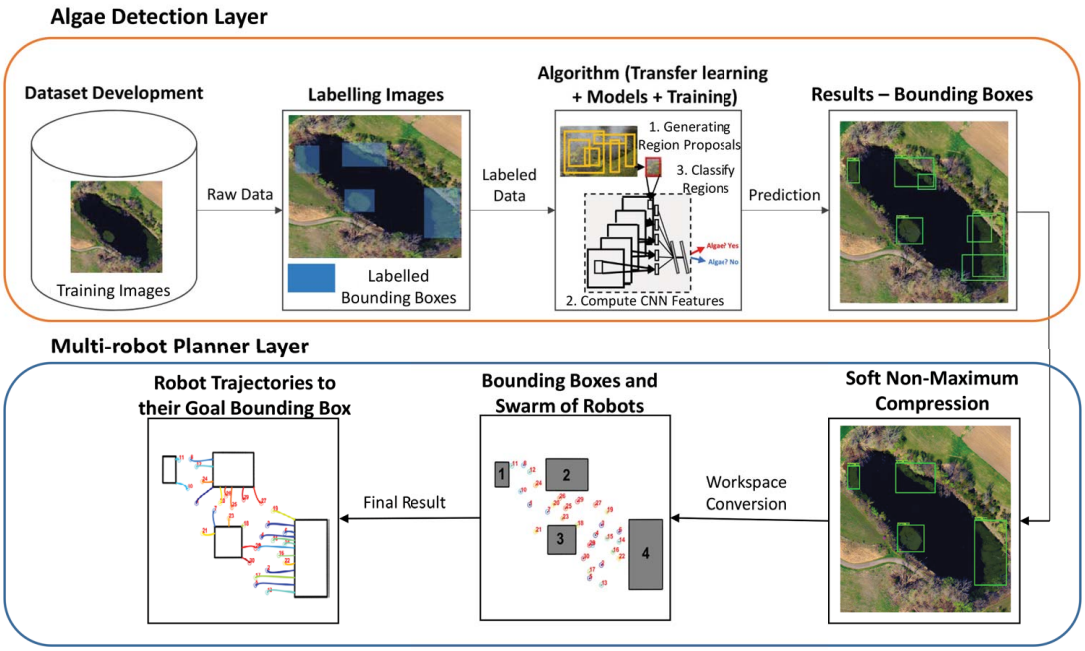


Figure 2.1. Workflow of the proposed Multi-robot Algae Removal Planner. The components of the planner can broadly be categorized into the *Algae Detection Layer* and the *Multi-robot Planner Layer* (Penmetcha et al., 2019).

patch must be detected singly. To achieve this, we employed a soft non-maximum suppression (NMS) algorithm (Bodla, Singh, Chellappa, & Davis, 2017). NMS makes use of scores on each bounding box to remove overlapping bounding boxes. This spill detector is implemented as the first layer in the Multi-robot Algae Removal Planner shown in Figure 2.1 (Penmetcha et al., 2019). With those bounding boxes calculated from the spill detector, we are able to plan robot team allocation.

Even if a spills can be identified in the image with bounding boxes, their boundary is more essential for a robot team to dedicate to coverage over the spill while avoiding empty areas. For instance, the efficiency for algae removal can be improved if a team of removal robots only performs coverage of the algae spills and avoids clean areas. Therefore, we propose as well a boundary detector that uses

image segmentation techniques to estimate the boundary (Luo, Singh, et al., 2019). The proposed method can even deal with coarse boundary situations, which are seen in many practical cases.

2.1.2 Robot Team Allocation (Team Partition)

In terms of robot team partition and allocation, plenty of partition techniques can be used to carry out such tasks, such as the *Voronoi diagram* (Breitenmoser, Schwager, Metzger, Siegwart, & Rus, 2010; Schwager et al., 2011) as well as clustering methods including *k-means* (Elango, Nachiappan, & Tiwari, 2011) and *spectral clustering* (Brunskill, Kollar, & Roy, 2007). Generally, the Voronoi diagram requires seeds, which are a specific subset of the plane specified. One could simply assume that those points are the centroids of every algae cluster. This solution works, but it does not take into account the sizes of algae clusters. K-means is sensitive to initial centroids and may not converge to the global optimum, and spectral clustering requires general connectivity within the robot team. Moreover, neither of them considers the location or geometry of algae clusters. This present study considers the partitioning of robots based on the area of the clusters, in order to accomplish a balanced workload for each robot. Moreover, achieving a minimum deployment displacement for the robot team is essential because it saves energy consumed in maneuvering.

2.1.3 Robot Team Deployment

The deployment objective is defined as optimally placing a group of robots in an environment of interest (Bullo, Cortés, & Martinez, 2009). Deployment problems play a relevant role in coordination tasks such as surveillance, search and rescue, data collections, and exploration for mapping in unknown environments. These up-to-date deployment algorithms and solutions stem from the interplay between spatial distribution and the limited sensing and network communication

coverage. Moreover, there are no universal notions or evaluation criteria for deployment quality. Depending on whether there exist designated deployment locations, the current multi-robot deployment can be classified into two categories: specified location deployment and unspecified location deployment. All those proposed strategies apply in multi-robot systems, with some special consideration for the characteristics of various robots.

Specified Location Deployment

Under certain circumstances, the users have to specify the deployment locations and a general strategy for the deployment. The task for the deployment control is to determine which robot should go where based on its availability. The scenario could be deploying a group of robots to a building to provide line-of-sight communication or monitoring after the map has been made by previous exploration (Simmons et al., 2000). Assuming that location information has been obtained, (Simmons et al., 2000) provided three deployment strategies.

1. *Group.* The group deployment moves all the robots concurrently from the closest locations to the furthest as a flood. Every single location witnesses a series of robots and is eventually occupied by the robot that visits last.

2. *Wave.* Wave deployment works similar to a stream, where the robot team moves in a line formation. The location to visit for a robot is the one visited by the previous robot. When the first robot reaches the destination in this wave-like manner, the last robot arrives at the first location visited by the group.

3. *Leap-frog.* Leap-frog deployment works like a Chinese checkers game: the first robot travels to the closest location and stays there, the second robot travels past the first robot and stays in its place, and so on.

It is clear that *wave* deployment has less efficiency than *group* deployment but is equal to *leap-frog* deployment. In order to avoid disconnection in communication or sensing, the next location to be visited cannot lie beyond the sensing range.

Unspecified Location Deployment

For deployment in a bounded environment without location information a priori, geometric optimization or probabilistic deployment strategies have to be utilized to achieve optimal deployment performance. (Bullo et al., 2009) utilized expected-value multicenter functions to define deployment tasks and used a Centroidal Voronoi Tessellation (CVT) approach to evaluate the strategy performance.

Given a bounded environment of interest $S \subset \mathbb{R}^d$ defined by a set of points $P_1, ..., P_n$, and a density function $\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$. Only the value of ϕ restricted to S is considered. Density function can be understood as the possibility that an event takes place in q over the environment; the larger the value of $\phi(q)$, the more important the location q is. It also defines a performance function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$, which is non-increasing and piecewise continuously differentiable. Performance functions describe the utility of placing a node to a designated location away from its original one. The smaller the distance, the larger the value of f, i.e., the better the performance. The value of f can be understood as the travel time or energy expenditure required to travel to a new place, or the signal-to-noise ratio (SNR) between a source with unknown location and a sensing robot trying to locate it. The *expected-value multicenter* function $\mathsf{H}_{\exp} : S \to \mathbb{R}$ can be defined as

$$\mathsf{H}_{\exp}(p_1, ..., p_n) = \int_S \max_{i \in \{1,...n\}} f(\|q - P_i\|)\phi(q)dq \tag{2.1}$$

The definition of $\mathsf{H}_{\exp}$ can be interpreted as follows: for each location $q \in S$, find the best coverage of q with appropriate point set $p_1, ..., p_n$ corresponding to the

value $\max_{i\in\{1,...n\}} f(\|q - P_i\|)$. Then the performance by the importance function $\phi(q)$ of the arbitrary location $q \in S$ is evaluated. Eventually, the performance of all those locations is summed over the environment S, thus obtaining $\mathsf{H}_{\exp}(p_1, ..., p_n)$.

Given the mathematic meaning of $\mathsf{H}_{\exp}$, it seeks to solve the following geometric optimization problem:

$$\text{maximize } \mathsf{H}_{\exp}(p_1, ..., p_n) \tag{2.2}$$

The deployment problem can be formulated based on the proposed expected-value multicenter function, assuming that S is a uniform robotic network, where the robots' working space is a polytope $Q \subset \mathbb{R}^d$ defined by their physical locations $P_1, ..., P_n$. Due to this, finding the best deployment strategy converts to a geometric optimization problem as Eq.(2.2).

To boost the performance of deployment, (Bullo et al., 2009) introduced the CVT method and achieved larger network coverage by continually driving robots toward the centroids of their Voronoi cells. To describe such deployment under CVT, it defined ϵ-*deployment task* $\mathsf{T}_{\epsilon-\text{dply}} : Q_n \to \{\mathsf{true}, \mathsf{false}\}$ as follows.

$$\mathsf{T}_{\epsilon-\text{dply}}(P) = \begin{cases} \mathsf{true}, & \text{if } \|p_i - CM_\phi(V^{[i]}(P))\| \leq \epsilon,\ i \in \{1, ..., n\} \\ \mathsf{false}, & \text{otherwise.} \end{cases} \tag{2.3}$$

where $V^{[i]}(P)$ denotes the Voronoi cell of robot i and $CM_\phi(V^{[i]}(P)$ denotes the Voronoi centroid computed with ϕ.

When $\mathsf{T}_{\epsilon-\text{dply}}$ is true, it means the robot moves sufficiently close to the Voronoi centroid to finish its deployment. The simulated deployment evolution is shown in Figure 2.2.

The aforementioned deployment strategies all apply to multi-robot systems. It is important to choose a proper one based on specific scenarios. Other physical constraints such as energy and time availability should be given enough consideration in building a heterogeneous MRS. For instance, aerial vehicles have a

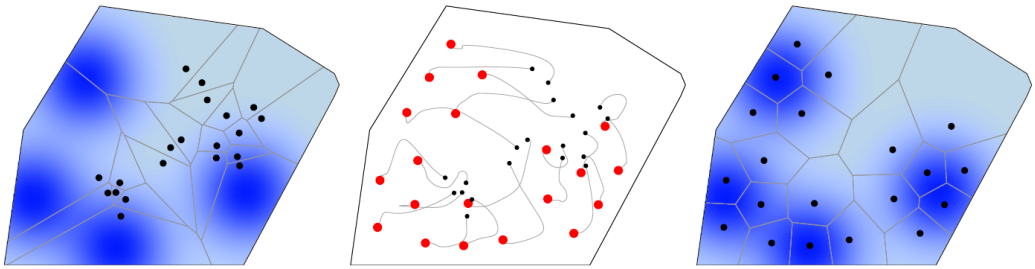


Figure 2.2. The deployment evolution with $N = 20$ robots. The left figure illustrates the initial locations and Voronoi partition, while the right figure illustrates the final ones. The evolution of the robots is indicated in the central figure and $\mathsf{H}_{\text{exp}} = -0.515$ (Bullo et al., 2009).

better view and faster mobility than ground robots; a heterogeneous exploration robotic team with them will have a better capacity to cover the working space in a short time. However, aerial robots cannot last long due to battery volume issues, so appropriate arrangements such as sequential and discrete deployment should be considered. In (Simmons et al., 2000), deployment locations for a heterogeneous robot team are specified by users after unknown environment mapping is achieved. A simple visibility deployment in a nonconvex environment without synchronization and under limited communication is presented in (Ganguli, Cortés, & Bullo, 2008).

2.2 Multi-robot Coverage

Coverage control based on the Voronoi diagram and Lloyd's algorithm proposed in (Bhattacharya, Ghrist, & Kumar, 2014; Schwager et al., 2011) allows a designated robot to collect a specific algae cluster in a static manner once the robot is deployed to the goal position. However, once again, this method does not consider whether the workload is within the capabilities of a robot, and this is a problem, given that collection relies on sucking with pumps and does not guarantee that an area of algae can be fully harvested within a certain period of time.

In the past, moving robot coverage methods included trapezoidal decompositions (H. M. Choset, 2005) and Morse decompositions (Acar, Choset, Rizzi, Atkar, & Hull, 2002), where a robot from the team was designated to take care of a specific small area. Both methods provide centralized controllers and, thus, are not scalable in computation or communication or robust in the case of a faulty robot. Coverage in water area with a multi-robot team was presented in (Li, Moridian, & Mahmoudian, 2016), but this experiment involves accurate trajectory planning and is less robust in unknown environments.

The work in (An, Qu, & Roberts, 2017) proposes motion planning based on triangulation for coverage of a target region. The studies in (Choi, Lee, Baek, & Oh, 2009; Hsu, Lin, & Yang, 2014) propose online and time-energy minimized path planning strategies for complete coverage, and, similar to our own research, both result in a spiral trajectory for the robot. However, these studies focus only on a single robot and do not show system scalability.

The work in (Y. Zhou, HU HS, et al., 2017) demonstrates a planning-based motion control strategy for multi-robot systems. However, a distributed multi-robot control strategy without any motion planning is necessary, meaning that a solution without pre-planning is more adaptive to dynamic and unknown environments. Additionally, it is important to achieve complete coverage of the patches while avoiding collision between robots, which is not yet reflected in (Y. Zhou et al., 2017). The paper (An, Qu, & Roberts, 2018) studies coverage control but mainly focuses on a single robot patrolling. Nevertheless, a coverage strategy performed by multiple robots is more necessary, as it leads to promising scalability. These limitations will be solved in this dissertation.

2.3 Multi-robot Rendezvous

Multi-robot rendezvous, which is also known as consensus (Olfati-Saber & Murray, 2004) and convergence (Ando, Oasa, Suzuki, & Yamashita, 1999), is to

achieve agreement over the physical location of as many robots as possible and steer the robots to a common location (Bullo et al., 2009). To this end, researchers begin their work primarily from three different aspects: 1) The utilization of a robotic network and the connectivity to achieve rendezvous, 2) the determination of an efficient rendezvous control law in order to achieve multi-robot rendezvous, and 3) dealing with potential constraints such as wireless communication availability, obstacle avoidance, executive time, robotic heterogeneity, etc., in robot rendezvous.

Typically, the prerequisite for multi-robot rendezvous is a connected communication graph, where every robot is a vertex and the connection between two robots is an edge. If one robot is not connected with all the others, it cannot realize the rendezvous task. Due to this, it is safe to assume that the network is connected initially, and one significant property of coordination algorithms for rendezvous is the maintenance of network connectivity among those robots. In a real circumstance, there would be obstacles or communication disturbances that prevent the robot group from building or maintaining a complete connection. In such cases, a distributed coordination control strategy that is superior for maintaining or rebuilding connectivity should be prioritized as the candidate for rendezvous control. Otherwise, the communication connection might be broken during maneuvering and prevent the rendezvous process.

Besides the aforementioned communication constraints, modern rendezvous control strategy should yield an optimal solution with requirements or other constraints such as shortest time consumption or trajectory, designated rendezvous time window for each robot, more efficient computing, dynamic environments, etc.

We will discuss the state of the art in multi-robot rendezvous research in this section and point out the major open questions that are significant but remain unresolved or are being solved.

2.3.1 Rendezvous Scenarios

Multi-robot rendezvous has long been used in a variety of scenarios such as multi-robot consensus while exploring an unknown environment (Cortés, Martínez, & Bullo, 2006), robot recharging (Keshmiri, 2011; Mathew, Smith, & Waslander, 2013; Mathew et al., 2015), and collaborative tasks such as interception and surveillance (Kunwar & Benhabib, 2006; Li et al., 2016). In the following sections, we review the literature of rendezvous control from three main categories.

Multi-Robot Consensus

In the case of multi-robot consensus, we consider the rendezvous problem to consist of robots exploring an unknown environment with minimum communication and arriving at the selected rendezvous location. The problem of rendezvous is ubiquitous in nature. Animals in migration are able to share information about food and water by rendezvous, thus allowing the whole group to know those locations. Humans also have this issue, as we need to meet specific people in specific places, which also applies in multi-agent robotic systems. With emerging technologies such as localization, ubiquitous wireless communication, and advanced computation capability, enhanced rendezvous control will bring wider application scenarios like intelligent warehouses and urban search and rescue (De Hoog, Cameron, Visser, et al., 2010; Dudek & Roy, 1997; Roy & Dudek, 2001).

An individual robot that explores part of the unknown bounded topological environment starting at an unknown location has to meet other robots somewhere to share knowledge and build a joint map (X. S. Zhou & Roumeliotis, 2006). Since the robots do not have a persistent rendezvous point but depend on stochastic possibility, and inter-robot communication is constrained or eliminated, the rendezvous research problem becomes how to find the landmarks that are the potential rendezvous points and enable robots to select a rendezvous location and decide the order of visits to the location based on the ranking criteria. The

rendezvous points should be highly stable and mutually agreed-upon extrema among robots (Roy & Dudek, 2001). The proposed method by Meghjani and Dudek depends on a set of conditions including world size, number of robots, starting location of each robot, and the presence of sensor noise (Meghjani & Dudek, 2011, 2012, 2014). The performance of their proposed rendezvous strategy is validated by two agents traveling the shortest path in various cities and selecting waypoints to realize rendezvous. In the presence of nonconvex environments, (Ganguli, Cortés, & Bullo, 2009) presented a coordination algorithm for mobile autonomous robots to rendezvous based on distributed visibility sensing.

Robot Recharging

Multi-robot collaboration involves continuous powering and energy maintenance. The robot recharging paradigm can be loosely divided into two main categories, namely stationary recharging docks and dynamic recharging station (Keshmiri, 2011). A stationary recharging dock means introducing a fixed charging station in the robot working environment; the robots rendezvous autonomously to the charging station from their current working locations (Silverman, Nies, Jung, & Sukhatme, 2002). The shortcomings include: 1) The robot needs to expend power to rendezvous to the recharging station. 2) Every individual robot has to monitor the energy threshold and guarantee sufficient energy to rendezvous and recharge when spreading over the working environment. 3) Chaos and collisions may happen when traffic around the docking station increases. Some of those issues may be tackled by exchanging energy cells such as portable batteries rather than local recharging (Ngo, Raposo, & Schiøler, 2008). However, sufficient space to store multiple batteries for the station becomes another challenge. In this research, finding an optimal rendezvous location, which is the recharging station location, and generating the shortest path while avoiding jamming are key challenges.

A dynamic recharging station provides a novel solution for multi-robot recharging. (Zebrowsk & Vaughan, 2005) provided a solution of a tanker robot to find and deliver energy to other robots, but communication and searching were bottlenecks for this approach, as the robot team was spread over a large area. To address this issue, (Litus, Zebrowski, & Vaughan, 2009) proposed a distributed heuristic for a group of mobile robots such that a single service robot can rendezvous with every other member of a team in a prescribed order and with minimum travel cost. Assuming a persistent task executed by a ground robot team in a non-obstacle environment, with a certain number of dynamic recharging robots carrying unlimited energy, a rendezvous strategy based on mixed integer linear programming can be deployed for a small number of recharging robots to find a set of recharging points at which rendezvous with working robots can occur (Mathew et al., 2013). (Mathew et al., 2015) expanded the rendezvous law to a team of unmanned aerial vehicles (UAVs) in a 3-D space; a heuristic and approximation algorithm applies, and the optimal solution is not computationally feasible when the population of robots grows significantly. In (Mathew et al., 2015), two recharging robots rendezvous with the four working robots during a persistent task. The shortcoming of this strategy is that it may not be compatible with changing environmental conditions, non-persistent tasks, or dynamic obstacles. The energy capacity of recharging robots is also not considered. The research challenge of this approach is to find a computationally efficient algorithm, algorithmic or heuristic, which partitions working robots corresponding to a specific recharging robot, and calculates optimal rendezvous points and their order for recharging robots to meet with every individual working robot.

Robot Surveillance

Robotic rendezvous can also accompany other collaborative tasks such as surveillance. (Li et al., 2016) proposed a scenario for searching for missing airplane

MH370 in the ocean with autonomous underwater vehicles (AUVs). As indicated in (Li et al., 2016), three AUVs are searching with a comb pattern, and one charging robot is maneuvering to rendezvous with different AUVs and recharge them when the working robot is running out of power. In this paper, the feasibility of having static recharging stations is also validated by simulation. It concludes that to persistently cover the whole area in surveillance and exploration, both static and dynamic mobile chargers are needed.

CHAPTER 3. MULTI-ROBOT DEPLOYMENT PROBLEM

This section discusses the optimal robot deployment strategy, which includes three research questions: i) How to identify the spills in the workspace in practice; ii) based on the identified spills, how to allocate the robots optimally considering the distribution of spills and initial locations of robots; iii) how the robots can be driven to the goal positions without colliding with each other. Necessary validation of the proposed allocation planner is conducted, and the results are presented to show the efficacy and efficiency of the proposed deployment strategy.

3.1 Spill Detection and Boundary Extraction

3.1.1 Boundary Extraction

For a detected spill in the bounding box, we can still increase the efficiency by eliminating empty areas in the case of a bounding box that is not fully filled with spill. As the solution, we perform an image segmentation strategy before planning the path for coverage operation. Thus, the USV only travels within the area of spill. Meanwhile, the segmentation method shows strong capability in handling a coarse boundary case that has long been a challenge for the research community.

We first process the image by applying a median filter and BGR2GRAY methods (Kapur & Thakkar, 2015) to convert the color image into a grayscale version, as indicated by *Step 1* in Figure 3.1. Based on the grayscale image, we then use the Canny operator to extract the boundary of the spill (Canny, 1986). For some spill with a coarse and obscure boundary, we propose to apply polynomial fitting and approximate the boundary with a spline. These are summarized as *Step 2*. In *Step 3*, since the spline plus image border forms a closed shape, we then

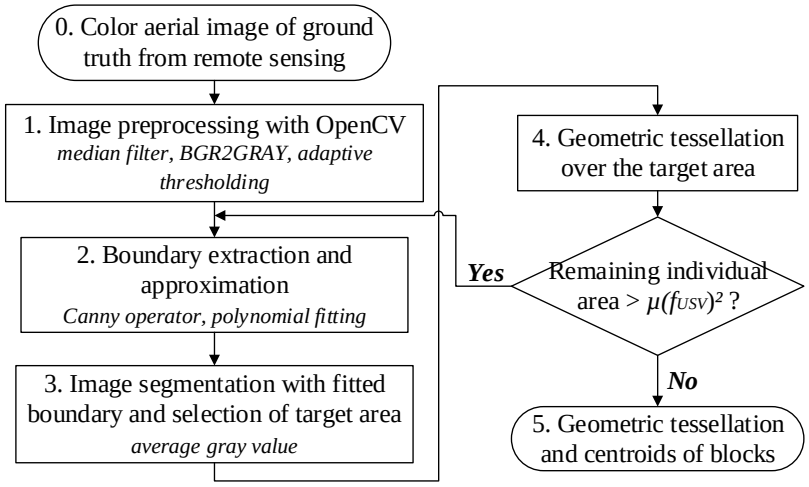
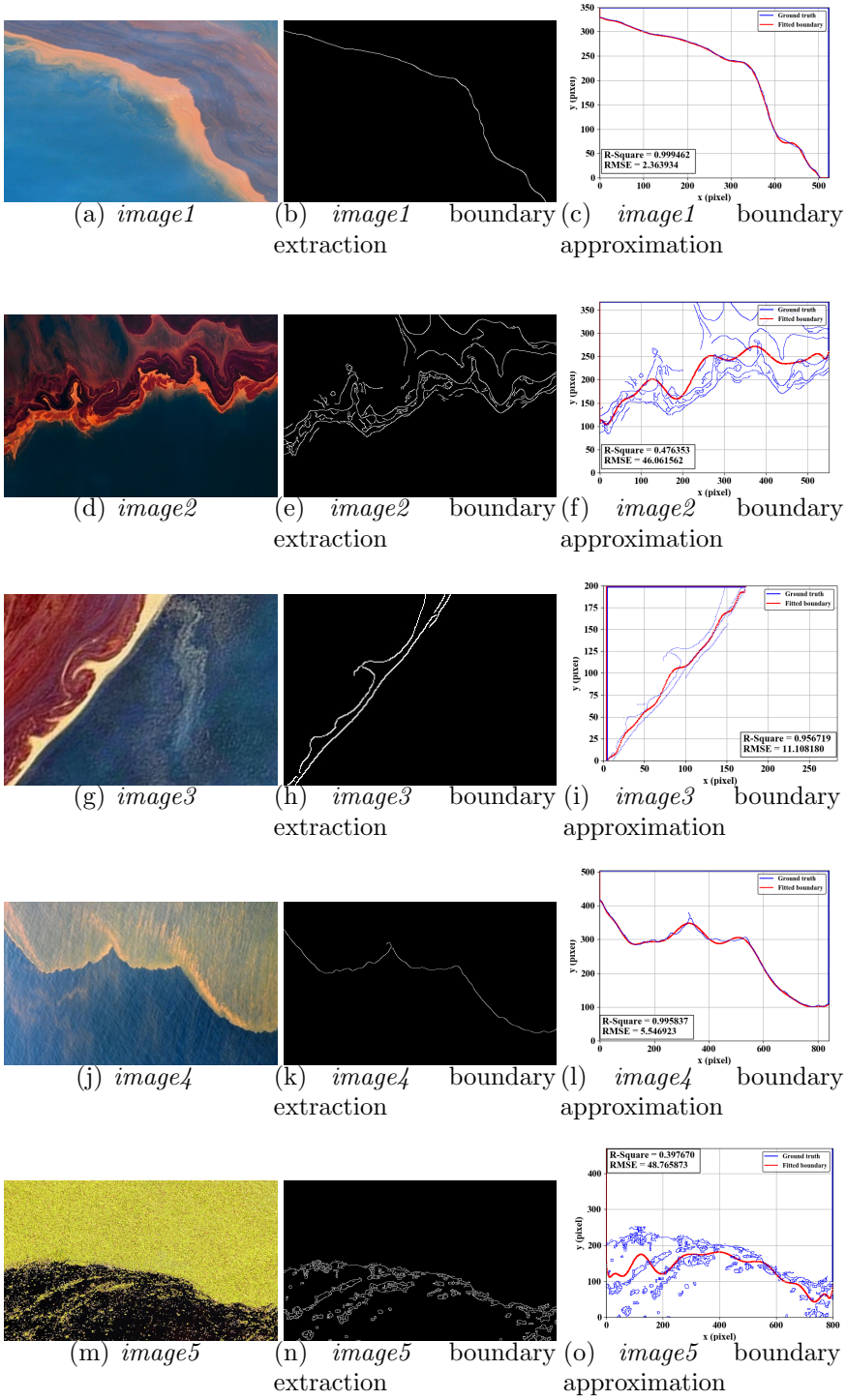


Figure 3.1. The flowchart of image segmentation and workspace tessellation, along with the techniques used.

differentiate the target area, which has much of the spill, from the background by referring to the average gray value. Usually, the background (e.g., water) has much higher grayscale (intensity) than the spill (e.g., oil or algae). Eventually, we perform a geometric tessellation to the target area with squares in *Step 4*. Since there may be small patches outside the target area or not covered by tessellation blocks, we examine the remaining area of the original image after subtracting the tessellated area. If some large spill patches remain, we can perform boundary extraction again until no large spill patch can be detected in the image. The coefficient μ refers to the ratio of the area of one remaining spill patch to the area of a tessellation block $(h_{USV})^2$. Ultimately, the tessellation outcome is produced in *Step 5*.

Noticeably, from *Step 2* to *Step 5* we confirm that our strategy can deal with spills with a coarse boundary. We first approximate the spill boundary with image processing. Thus, we remove a large portion of spill. By adjusting the coefficient μ and redoing *Step 4*, we are able to cover the remaining spill patches and achieve the coverage with high completeness.



(a) *image1* (b) *image1* boundary extraction (c) *image1* boundary approximation

(d) *image2* (e) *image2* boundary extraction (f) *image2* boundary approximation

(g) *image3* (h) *image3* boundary extraction (i) *image3* boundary approximation

(j) *image4* (k) *image4* boundary extraction (l) *image4* boundary approximation

(m) *image5* (n) *image5* boundary extraction (o) *image5* boundary approximation

Figure 3.2. The procedure of the image segmentation using five different real aerial images. The real images are presented in the first column. The outcomes after boundary extraction are shown in the second column, while the approximated boundary is shown in the third.

We evaluate the goodness of boundary approximation with both the coefficient of determination R^2 and the Root mean square error (RMSE), defined as below. In our test, the R^2 and RMSE values are indicated in every specific figure in the third column of Figure 3.2.

$$R^2 \triangleq 1 - \frac{SS_{res}}{SS_{tot}}, \quad RMSE \triangleq \sqrt{\frac{\sum_{i=1}^{T}(y_i - f_i)^2}{T}}, \tag{3.1}$$

where $SS_{res} = \sum_{i=1}^{T}(y_i - f_i)^2$, $SS_{tot} = \sum_{i=1}^{T}(y_i - \bar{y})^2$. Here, y_i refers to the y coordinate of a point on the extracted spill boundary, f_i refers to the y_i values after fitting, $\bar{y}$ means the mean of y_i, and T refers to the total number of pixels along the x-axis.

We have tested multiple images with spills using the proposed algorithm shown in Figure 3.1. The results are shown in Figure 3.2.

3.2 Robot Allocation

The robot allocation here means how to partition the robot team optimally according to the number and area of patches, as well as the distance from each robot to its assigned patch. The optimal allocation should yield the minimal sum of traveling displacement of every agent to reach its goal position, in order to prevent excessive energy consumption in robot deployment. More specifically, robot allocation should meet two goals: i) The allocation should guarantee that at least one robot is associated with each patch, since it is assumed that the number of robots is greater than the number of patches. ii) The number of robots associated with a specific patch should be in proportion to the patch area. In other words, the first goal is to make sure every patch in the workspace can be taken care of, and the second goal is to achieve a balanced workload for every deployed robot by assigning a uniform "density" of robots to every patch. As each robot has the same coverage capability, a uniform density of robots implies a uniform completion time for patch

coverage. That makes the completion time predictable and unique, thus facilitating robot recall after the work is completed. The initial global map can be provided by a unmanned aerial vehicle (UAV). When the robot allocation is done, the UAV can be removed from the system. Robot deployment happens after allocation.

Robot allocation is carried out after a global map of the workspace $\mathcal{W}$ including all the robots and patches is produced by a vision-sensor-equipped UAV. The shortest distance $\|d_{ij}\|_2$ between q_{si}, the start position of robot $R_i, i \in 1, 2, ..., N$, and q_{gi}, the goal position that lies on the edge of patch $\mathcal{C}_j, j \in 1, 2, ..., M$, can be estimated after those objects are identified by image processing techniques(Konolige et al., 2008). To achieve the first goal previously mentioned, a density value ρ_d is calculated as $\rho_d = \frac{N}{\sum_{i=1}^{M} A_i}$, where $A_i = \int \mathcal{C}_i$ is the area of patch $\mathcal{C}_i$. The number of robots preferred for each patch is therefore $n_i = \rho_d \cdot A_i$. Special consideration should be given to the case when calculated $n_i < 1$, meaning no robot is allocated to patch $\mathcal{C}_i$. As a solution, our strategy prioritizes a patch that is relatively small in area when determining the number of robots associated. Working from the assumption that the number of available robots is no less than the number of patches, we could select $n_i = 1$ in the case that $n_i < 1$, and thus, every patch can be taken care of.

The shortest distance set $\{\|d_{ij}\|_2\}$ achieving the second goal and minimizing the total traveling displacement is obtained from the UAV. The best solution can be attained by exhaustive search when converting this location-allocation problem into a combinatorial problem. However, when the size of the problem grows, an exhaustive search method becomes impractical due to its computation complexity of $O(M^N)$, which can be confirmed by quantitative experiments and the complexity theory. Even if we set the two goals as constraints and reduce the search space, the quantity of potential solutions is

$$\prod_{i=1}^{M} \binom{N - \sum n_{i-1}}{n_i} = \prod_{i=1}^{M} \frac{(N - \sum n_{i-1})!}{(N - \sum n_{i-1} - n_i)! \cdot (n_i)!}, \tag{3.2}$$

where $n_0 = 0$. Nevertheless, the result is still huge.

In order to cope with a dynamic environment or to achieve greater scalability, algorithms with less computation cost are required. Chen (2010) proposed the ant colony clustering algorithm that reduces the computation time for robots assembling at predefined positions to be linear and independence from number of objects (Chen, 2010). Alternatively, the generalized Voronoi diagram method can be applied, resulting in a fair partition over the robot group with a complexity of $O(N \log N)$, when a sufficient number of robots are distributed evenly in the workspace (Breitenmoser et al., 2010; H. Choset & Burdick, 2000). Generic heuristic methods such as Genetic Algorithm (Min et al., 2013) and other evolutionary algorithms are applicable, but the global optimality is not guaranteed.

3.2.1 Optimization Method

We formulate the allocation problem into a binary integer linear programming (ILP) problem as below in (3.3)-(3.7), and an exact optimal solution is obtained by using a generic MILP solver in MATLAB.

$$\min \sum_{j=1}^{M} \sum_{i=1}^{N} z_{ij} \cdot \|d_{ij}\|_2 \tag{3.3}$$

subject to

$$\sum_{i=1}^{N} z_{ij} \geq 1, \quad \forall j \in \{1, 2, ..., M\} \tag{3.4}$$

$$\sum_{j=1}^{M} \Big(\frac{A_j}{\sum_{i=1}^{N} z_{ij}} - \rho_d \Big)^2 \leq \delta^2 \rho_d^2 \tag{3.5}$$

$$\sum_{j=1}^{M} z_{ij} = 1, \quad \forall i \in \{1, 2, ..., N\} \tag{3.6}$$

$$z_{ij} = \begin{cases} 1, & \text{if robot } R_i \text{ is allocated to patch } \mathcal{C}_j, \\ 0, & \text{otherwise,} \end{cases} \tag{3.7}$$
$$\forall i \in \{1, 2, ..., N\}, \forall j \in \{1, 2, ..., M\}$$

The objective function (3.3) seeks to minimize the total traveling displacements of every robot in the workspace. Constraint (3.4) represents the first goal proposed at the beginning of this section and guarantees every patch is taken care of by at least one robot. Constraint (3.5) is to satisfy the second goal proposed and let the number of robots allocated be proportional to the area of the patch. Here a coefficient δ is introduced to control the variance of the "density" of robots among different patches. Constraint (3.6) is to guarantee that one specific robot can be allocated to only one patch, while the last constraint (3.7) specifies a binary characteristic for the decision variables z_{ij}.

Since (3.5) is a nonlinear constraint, faster convergence speed can be obtained by converting it to be a linear constraint with simple arithmetics. The relaxed constraint is shown as follows:

$$\left| \frac{A_j}{\sum_{i=1}^{N} z_{ij}} - \rho_d \right| \leq \frac{\delta \rho_d}{\sqrt{M}}, \quad \forall j \in \{1, 2, ..., M\}, \tag{3.8}$$

which can be further specified successively with two linear constraints, (3.9) and (3.10), shown below,

$$(1 + \delta^*)\rho_d \cdot \sum_{i=1}^{N} z_{ij} \geq A_j, \quad \forall j \in \{1, 2, ..., M\}, \tag{3.9}$$

$$(1 - \delta^*)\rho_d \cdot \sum_{i=1}^{N} z_{ij} \leq A_j, \quad \forall j \in \{1, 2, ..., M\}, \tag{3.10}$$

where $\delta^* = \delta / \sqrt{M}$.

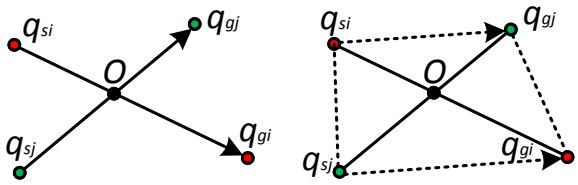


Figure 3.3. A representative figure showing an intersection between trajectories of two robots R_i and R_j.

Remark 3.2.1 *With the allocation configuration obtained from (3.3), no intersection exists between the trajectories of any two robots working in the deployment state.*

Proof This remark is proved by contradiction. Without loss of generality, we assume that there is one intersection between the trajectories of robots R_i and R_j, starting from their current positions q_{si} and q_{sj}, and ending in their goal position q_{gi} and q_{gj}, respectively. The intersection is denoted as O, shown in Figure 3.3 (left). If connected with line segments, the four points form a quadrilateral. We can show that $\overline{Oq_{si}} + \overline{Oq_{gj}} > \overline{q_{si}q_{gj}}$ and $\overline{Oq_{sj}} + \overline{Oq_{gi}} > \overline{q_{sj}q_{gi}}$, thus we show $\overline{q_{si}q_{gi}} + \overline{q_{sj}q_{gj}} > \overline{q_{si}q_{gj}} + \overline{q_{sj}q_{gi}}$. Obviously, those two robots can have less total displacement if their goal positions are swapped, which clearly violates the optimization goal. Conclusively, there is no intersection of the trajectories of any two robots.

■

Remark 3.2.2 *The replacement of the nonlinear constraint (3.5) with the linear constraint (3.8) may lead to a larger marginal variance of optimization outcomes.*

Proof The constraints (3.5) and (3.8) are to satisfy the second allocation goal by determining the threshold of variance. We then start with analyzing how much variance can be allowed for the allocation result of a specific patch $\mathcal{C}_x$ under those constraints. Let $\sigma^* = \delta^2\rho_d^2$, number of allocated robots of $\mathcal{C}_x$ being $E = \sum_{i=1,j=x}^{N} z_{ij}$,

then the variance can be defined as $\sigma_E = (\frac{A_x}{E} - \rho_d)^2$ and (3.8) is equivalently converted to

$$\sigma_E \leq \sigma^*/M. \tag{3.11}$$

Denoting the disturbance $\zeta \in \mathbb{Z}$ the additional number of robots allocated to $\mathcal{C}_x$, we show the updated variance with disturbance as $\sigma_{E+\zeta} = (\frac{A_x}{E+\zeta} - \rho_d)^2$. The difference in variance before and after introducing a disturbance is:

$$\begin{aligned} \Delta_\sigma = \sigma_{E+\zeta} - \sigma_E &= (\frac{A_x}{E+\zeta} - \rho_d)^2 - (\frac{A_x}{E} - \rho_d)^2 \\ &= \frac{2\rho_d A_x \zeta}{E(E+\zeta)} - \frac{2EA_x^2\zeta + A_x^2\zeta^2}{E^2(E+\zeta)^2}. \end{aligned} \tag{3.12}$$

Considering that, $A_x = \rho_d E$, (3.12) becomes as follows:

$$\Delta_\sigma = \frac{2\rho_d^2 E\zeta}{E(E+\zeta)} - \frac{2E^3\rho_d^2\zeta + \rho_d^2E^2\zeta^2}{E^2(E+\zeta)^2} = \frac{\rho_d^2\zeta^2}{(E+\zeta)^2}. \tag{3.13}$$

By selecting various disturbances $\zeta = \{-2, -1, 1, 2\}$ and a constant σ^*/M, the relationship between Δ_σ and E is illustrated in Figure 3.4. We can see from Figure 3.4 that a smaller patch contributes greater variance due to the offset of its allocated number of robots. For instance, a patch that deserves *two* allocated robots can produce a maximal variance of ρ_d if having *one* more or less robot; however, a patch that deserves *ten* allocated robots can only produce a maximal variance of $0.01\rho_d$ with the same disturbance. Furthermore, with constraint (3.5), σ^*/M has to follow the below condition to guarantee a solution for the objective function (3.3):

$$\sigma^*/M \geq \max \min_E \Delta_\sigma(E, \zeta). \tag{3.14}$$

Since larger patches produce less variance even under greater disturbance, a loose threshold as (3.14) leads to a large variance of optimization outcomes, which ends the proof.

■

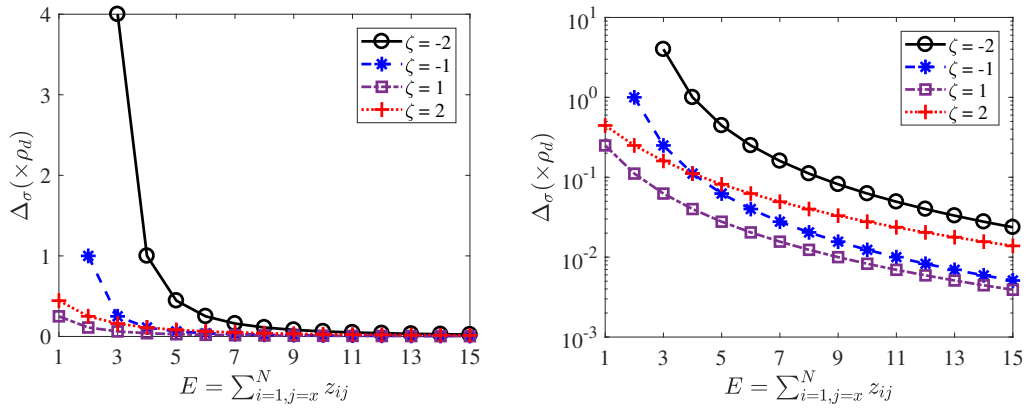


Figure 3.4. The change in variance Δ_σ in linear (left) and logistic (right) by injecting a disturbance ζ into the number of allocated robots E for a patch.

Constraint (3.5) limits the collective variances of allocated numbers of robots from all patches, in which greater variances for small patches will lead to less variance for large patches, and vice versa. While (3.5) with this feature theoretically prevails over linear constraint (3.8) according to Remark 3.2.2, it may also face other issues, such as existence of solution. The applicability of linear constraint (3.8) in robot allocation is validated through simulation experiments, with the results shown in Table 3.3.

3.3 Robot Deployment

After determining the goal position for each robot with allocation, we build a motion controller based on the artificial potential field (APF) method (Valbuena & Tanner, 2012; G. Zhang, Fricke, & Garg, 2013) to realize deployment for non-holonomic robots.

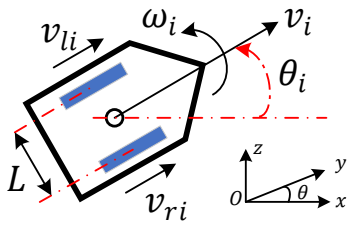


Figure 3.5. The mobility chassis of the robot driven by two wheels. The robot has three degrees of freedom, i.e., $\dot{x}_i$, $\dot{y}_i$, and ω_i.

3.3.1 Vehicle Dynamics

Let the generalized coordinate of a robot be $q_i = (x_i, y_i, \theta_i)$. A unicycle model for such a non-holonomic robot (Kim & Kim, 2003; G. Zhang et al., 2013) is introduced as below and illustrated in Figure 3.5.

$$\dot{q}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = J(\theta_i) Q_i \tag{3.15}$$

$$\mathbf{Q}_i = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(v_{ri} + v_{li}) \\ \frac{1}{L}(v_{ri} - v_{li}) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} v_{ri} \\ v_{li} \end{bmatrix} \tag{3.16}$$

where $\mathbf{Q}_i$ is the velocity vector and control objective of robot R_i consisting of linear velocity v_i and angular velocity ω_i with respect to the centroid. L in (3.16) denotes the tread width of the robot. The angular velocity ω_i is bounded by ω_{max}, which will be determined later in equation (4.10) of Section 4.2.3. Furthermore, the angular controller for the robot is discrete in time and its resolution is $\phi_{ci} = \omega_i T_c$, where T_c is the system cycle. Since the robot is driven by the two motors, the control objective space of Q_i has to be transformed into the space of v_{li} and v_{ri}. Additionally, we formulate control input as $[u_i, w_i]$ to distinguish it from the robot velocity $[v_i, \omega_i]$.

3.3.2 Velocity Control

Given the coordinates of goal positions for robots in the workspace, which can be transmitted from a UAV to every individual robot, our study applies the APF method to implement navigation for deployment. The navigation can be achieved with a robot onboard GPS sensor that measures the current location. An attractive potential field is introduced for navigation, and a repulsive potential field is introduced for collision avoidance. The robots working in the searching and coverage states are assumed to have a higher avoidance priority than those maneuvering for deployment.

The attractive potential for robots working in the deployment state is formulated as

$$U_d(\mathbf{q}) = \frac{1}{2}\xi_1 \|d(\mathbf{q}, \mathbf{q}_g)\|_2^2 \tag{3.17}$$

where ξ_1 is a scaling parameter and $\|d(\mathbf{q}, \mathbf{q}_g)\|_2$ the distance between the current positions of robots and their goals, which coordinates can be obtained from GPS sensors. Meanwhile, there is repulsive potential exerted on the robots working in deployment state by the robots in either searching or tracking state, in order to avoid collision. This repulsive potential field is formulated as

$$U_i(\mathbf{q}) = \begin{cases} \frac{\xi_2}{2}\left(\frac{1}{\|d(\mathbf{q},q_i)\|_2} - \frac{1}{d_0}\right)^2, & \text{if } \|d(\mathbf{q}, q_i)\|_2 \leq d_0, \\ 0, & \text{if } \|d(\mathbf{q}, q_i)\|_2 > d_0, \end{cases} \tag{3.18}$$
$$i \in \mathcal{N}_s \cup \mathcal{N}_t$$

where ξ_2 is a positive scaling factor, and $\|d(\mathbf{q}, q_i)\|_2$ denotes the distance between robot R_i that works in searching or tracking state and any other robots within its sensing scope. $\mathcal{N}_s$ and $\mathcal{N}_s$ denote the robots working in searching and tracking state, respectively. Here d_0 is the scope range, and we further assume that all the robots have equal $d_0 = r_{\mathcal{A}}$.

Due to this, for any robot, the final constructed potential field for deployment navigation is:

$$U^*(\mathbf{q}) = U_d(\mathbf{q}) + \sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} U_i(\mathbf{q}) \tag{3.19}$$

From the potential field function (3.19), the control input for the robots is obtained:

$$\mathbf{u} = k_u \tanh(\|\mathbf{q}_g - \mathbf{q}\|^2), \tag{3.20a}$$

$$\mathbf{w} = -k_w(\boldsymbol{\theta} - \boldsymbol{\varphi}) + \dot{\boldsymbol{\varphi}}. \tag{3.20b}$$

Here, $k_u > 0$ and $k_w > 0$ are control gains, $\mathbf{u} = [u_1, u_2, ..., u_j]$ and $\mathbf{w} = [w_1, w_2, ..., w_j]$ are the input linear and angular velocity vectors of all the robots in deployment state $\mathcal{N}_d$, respectively. $\varphi_i \triangleq \arctan\left(\frac{U_y^*}{U_x^*}\right)$ is the direction of the collective potential field at a point (x, y), while $\dot{\varphi}_i$, the time derivative of φ_i, is shown as below considering the nonlinear model (3.15):

$$\dot{\varphi}_i = \frac{1}{U_x^{*2} + U_y^{*2}} \Bigg(\left(\frac{\partial U_y^*}{\partial x} \cos\theta_i + \frac{\partial U_y^*}{\partial y} \sin\theta_i \right) U_x^* - \left(\frac{\partial U_x^*}{\partial x} \cos\theta_i + \frac{\partial U_x^*}{\partial y} \sin\theta_i \right) U_y^* \Bigg) u_i$$

where u_i is provided in (3.20a). Here, the singularity happens when $U_x^{*2} + U_y^{*2} = 0 \Leftrightarrow U_x^* = 0 \wedge U_y^* = 0$. The singularity can be reached only if robot R_i is located at the local minimum of the potential field, and R_i will be stuck. However, we conclude in Remark 3.2.1 that such local minimum does not exist if applying our proposed hybrid control scheme.

Noticeably, the input velocity $\|u_j\|$ for the searching state does not need to be bounded by the maximum speed, because the robot has not yet started coverage operation. But in analysis, the velocity is made to be bounded by v_{max} just to

simplify the analysis process. Meanwhile, the robot acceleration is bounded mechanically for the searching state.

The linear control input (3.20a) is to motivate the robot R_i to approach the goal q_{gi}, while the angular control (3.20b) is to align the orientation of the robot to the direction of the collective potential field U^*, which is φ_i. The control law (3.20) makes the robot with a unicycle model converge to the goal of deployment asymptotically from almost all initial conditions; its stability is proved in *Proposition 1* of (Valbuena & Tanner, 2012).

One may wonder why the repulsive potential component (3.18) does not include those robots in deployment state $\mathcal{N}_d$. According to Remark 3.2.1, the reason is that one robot in the deployment state cannot collide with any other robots while moving under the same state, since there is no intersection between their paths. Furthermore, due to the existence of avoidance priority, a saddle point caused by the local minimum in an APF does not exist. This fact guarantees that every allocated robot ultimately will reach its goal position.

Deadlock equilibrium is a common issue that hinders robot movement in a multi-robot situation. Although robots working in deployment state can avoid collision by eliminating a saddle point, as discussed above, they still face a deadlock issue for unicycle models. Specifically, a robot with a unicycle model has to steer from the initial orientation while approaching to the goal position. This steering may exert conflicting forces upon adjacent robots in opposite directions and hence cause deadlock equilibria. To eliminate such deadlock, we can increase the scope range d_0 in (3.18) to allow more room for steering, or stop robots before entering a computed collision zone and resume motion once the zone is clear (Soltero, Smith, & Rus, 2011). Alternatively, strategies based on robot trajectory re-planning such as (Qutub, Alami, & Ingrand, 1997) or reciprocal velocity obstacle method (RVO) (Van den Berg, Lin, & Manocha, 2008) can also be utilized. As the validation, the trajectories of robots in coverage using the unicycle model can be seen in Figure 4.14 from Chapter 3.

3.4 Robot Allocation Validation

3.4.1 Robot Allocation for Rectangular Areas Defined by Bounding Boxes

In this section, we validate the proposed optimization-based robot allocation strategy in (3.2) in a scenario of algae bloom cleaning using vision sensing techniques presented in Section 2.1.1. The results are shown in Figure 3.6.

The proposed DNN-based detector depicted in Figure 2.1 can yield rectangles, namely bounding boxes, to represent the spills. Based on the relative locations and areas of those bounding boxes denoted as $\mathcal{B}_i$, we tested the performance of the proposed allocation planner in (3.3), and the allocation result is shown in Table 3.1. From the results, we see that the number of robots allocated for a bounding box was in proportion to the area of the box. Meanwhile, compromising with the previous goal, each robot moved the shortest distance to its associated box. The allocation results demonstrate consistency in terms of different numbers of bounding boxes and locations. The robot trajectories in deployment under the proposed controller (3.20) are also shown in Figure 3.6.

Table 3.1.

Number of robots allocated per bounding box under three different scenarios

Scenario 1 (N=10)	**Bounding boxes (2)**	$\mathcal{B}_1$	$\mathcal{B}_2$	-	-
	Areas (m^2)	0.541	0.478	N/A	N/A
	Allocated robots (#)	5	5	N/A	N/A
Scenario 2 (N=20)	**Bounding boxes (3)**	$\mathcal{B}_1$	$\mathcal{B}_2$	$\mathcal{B}_3$	-
	Areas (m^2)	0.475	0.508	0.327	N/A
	Allocated robots (#)	8	7	5	N/A
Scenario 3 (N=30)	**Bounding boxes (4)**	$\mathcal{B}_1$	$\mathcal{B}_2$	$\mathcal{B}_3$	$\mathcal{B}_4$
	Areas (m^2)	0.079	0.325	0.193	0.572
	Allocated robots (#)	2	9	6	13

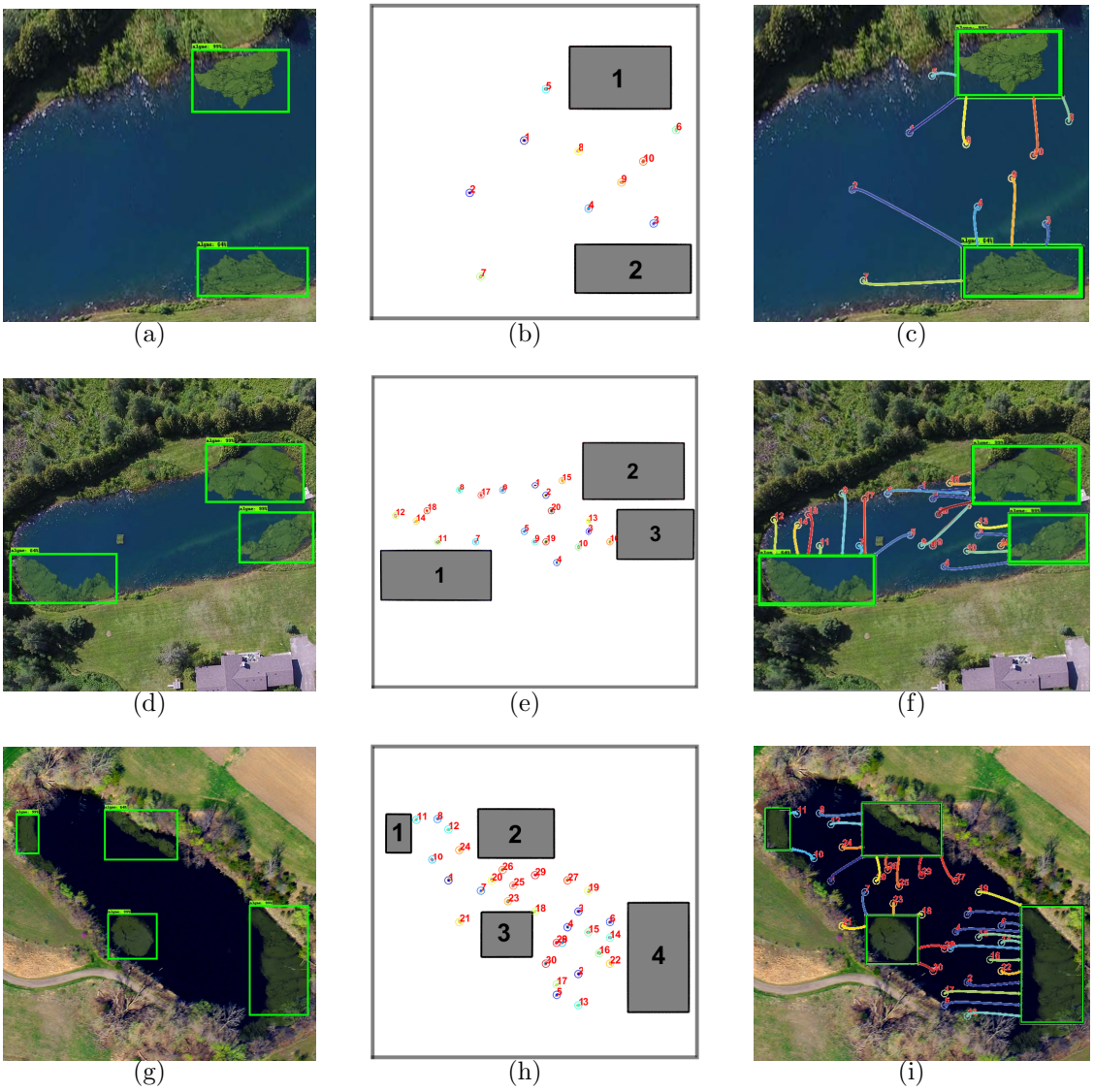


Figure 3.6. (a), (d), and (g) are the ground truth of workspace that contains algae patches, labeled with *Scenario 1, 2, and 3* with the number of bounding boxes being $M = 2, 3,$ and 4. (b), (e), and (h) show the initial robot distributions in the workspace. (c), (f), and (i) indicate the trajectories of the deployed robots during allocation for the three scenarios, respectively.

3.4.2 Robot Allocation for Areas with Different Shapes

To further demonstrate the performance of the allocation planner, we conduct validation for robot allocation and deployment in the Robotarium MATLAB platform (Pickem et al., 2017), which is an open source swarm robotics experiment testbed. Robotarium also supports hardware experiments with GRITSbot robots. Robotarium has a 2D arena of size 3m × 3m, which can be used to simulate a scaled-down aquatic workspace, such as a lake or ocean, where spills are patched. In the experiments, four spills with different geometric features are presented, including two circles with different areas, one ellipse and one quadrilateral, as shown in Figure 3.7. The detailed geometric parameters of this setting are summarized in Table 3.2.

Table 3.2.

Geometry of the spills in the workspace $\mathcal{W}$.

Geometry	$spill1$	$spill2$	$spill3$	$spill4$
$Area(m^2)$	0.4301	0.4046	0.4200	0.0314
$Perimeter(m)$	2.3247	2.3926	2.8415	0.6283

Specifically, we designed three scenarios for the workspace $\mathcal{W}$ including circle, eclipse, and polygon; they are detailed in Figure 3.8 with the number of robots being $N = \{5,\ 10,\ \text{and}\ 40\}$. The allocation result of robots under this setting is detailed in Table 3.3.

Case 1 (N = 5)

In this case, a minimum number of *five* robots was tested. Each spill in the workspace should be allocated with at least one robot. As documented in the top field of Table 3.3, every spill was allocated with one robot, except for spill 3. This result confirmed the efficacy of the proposed allocation planner when the number of robots is small.

Table 3.3.

Performance of having $N = \{5,\ 10,\ \text{and}\ 40\}$ robots in coverage, with corresponding maximum iterations k_{max}.

Case	**Metrics**	***Spill*1**	***Spill*2**	***Spill*3**	***Spill*4**
$\boldsymbol{N = 5}$ $k_{max} =$ $25,000$	*Residual area at* $k_{max}(m^2)$	2.85×10^{-4}	7.94×10^{-5}	7.85×10^{-5}	7.85×10^{-5}
	Completeness at $k_{max}(\%)$	99.93	99.98	99.98	99.75
	Allocated robots # (indices)	1 (#5)	1 (#4)	2 (#1, #2)	1 (#3)
	k_{stop} *at* A_{min} (#)	23485	20464	14677	1518
	$D_{sum}(m)$	68.8	59.3	296.3	23.1
$\boldsymbol{N = 10}$ $k_{max} =$ $11,000$	*Residual area at* $k_{max}(m^2)$	7.85×10^{-5}	3.22×10^{-4}	2.54×10^{-3}	7.85×10^{-5}
	Completeness at $k_{max}(\%)$	99.98	99.92	99.39	99.75
	Allocated robots # (indices)	2 (#1, #8)	4 (#5, #6, #9, #10)	3 (#2, #4, #7)	1 (#3)
	k_{stop} *at* A_{min} (#)	9907	4815	10240	1196
	$D_{sum}(m)$	75.1	861.5	959.3	14.1
$\boldsymbol{N = 40}$ $k_{max} =$ $3,000$	*Residual area at* $k_{max}(m^2)$	7.85×10^{-5}	7.97×10^{-5}	7.19×10^{-4}	7.85×10^{-5}
	Completeness at $k_{max}(\%)$	99.96	99.98	99.83	99.75
	Allocated robots # (indices)	12 (#1, #9, #11, #14, #17, #18, #21, #22, #30, #33, #34, #39)	12 (#2, #3, #10, #13, #15, #16, #19, #20, #27, #35, #36, #37)	15 (#4, #5, #6, #7, #8, #12, #23, #24, #25, #26, #28, #29, #31, #32, #40)	1 (#38)
	k_{stop} *at* A_{min} (#)	2161	1850	2065	1085
	$D_{sum}(m)$	2576.8	3317.9	4815.0	30.9

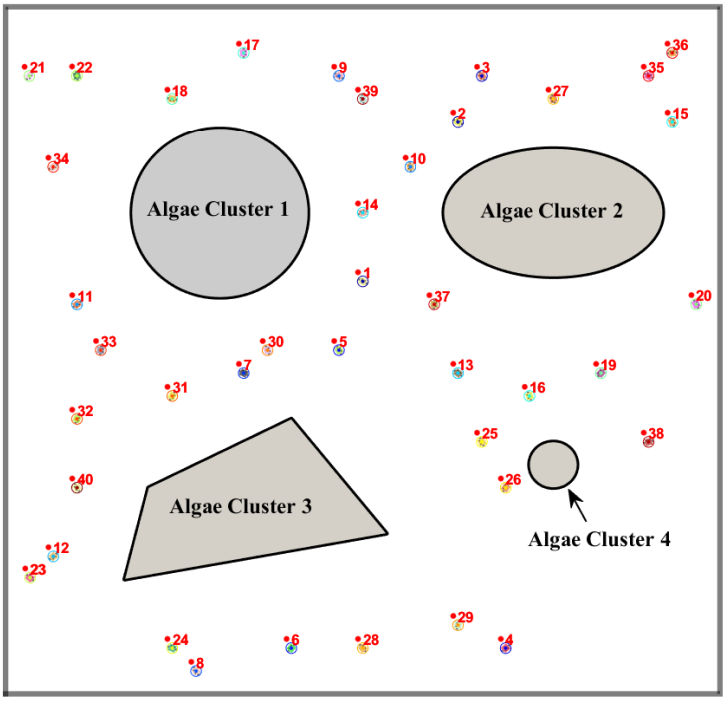


Figure 3.7. Setting of simulation experiment featured with 4 spills of different shapes and areas, and robots randomly distributed in the workspace.

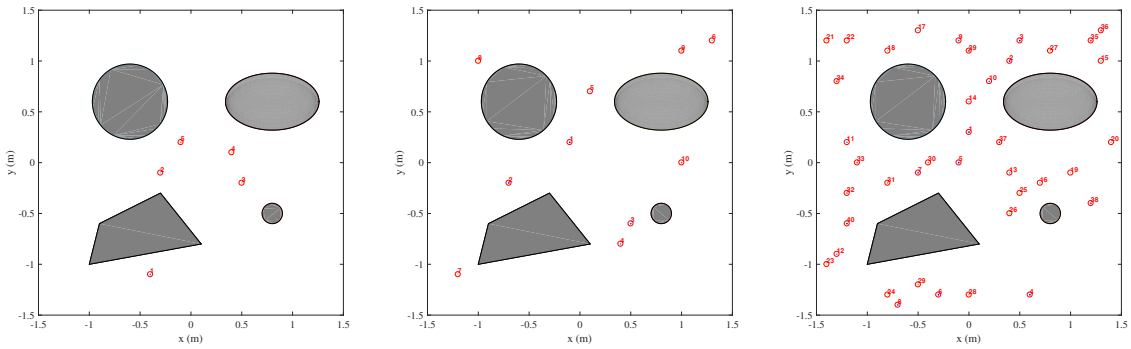


Figure 3.8. This row of images shows the initial random distributions of robots for three cases: Case 1, 2, and 3 from left to right, each with a different number of robots. The association of robots to these spills after allocation is shown in Table 3.3.

Case 2 (N = 10)

In this case, we want to show whether spills can be removed in a shorter time than the first case by having more robots. We adopted $\delta^* = 0.9$ in (3.9) and (3.10). From the result shown in Table 3.3, the allocation planner achieved a good balance between the density of robots allocated to each spill and the overall deployment distance.

Case 3 (N = 40)

In this case, we introduce the highest number of robots. The result shows that robots can be allocated to each spill based on the related distances and be in proportion to the area of four spills. With more robots, the parameter δ^* can be even lower, such as $\delta^* = 0.5$, to prioritize the constraints in (3.9) and (3.10), among others.

3.5 Conclusion

In this section, we discussed the multi-robot deployment problem. We first identified spills in the workspace and extracted spill boundaries for further operations. In terms of robot allocation to the discrete spills, we applied an optimization method in determining the optimal partition and deployment goals. When the goal position for each robot is decided, we proposed an APF-based motion controller to navigate the robots to the goals. By prioritizing the collision avoidance hierarchy to avoid local minima and deadlock, the robot team will not have any collision or stalemate during the deployment.

Acknowledgment

In Section 3.1, the subsection Section 3.2 is published in (Penmetcha et al., 2019), while the subsection Section 3.1.1 is published in (Luo, Singh, et al., 2019). Section 3.3 is included in (Luo, Kim, & Min, (Under review)).

CHAPTER 4. MULTI-ROBOT COVERAGE PROBLEM

In this section, we present coverage control strategies that a team of robots can use to remove or explore every individual patch in the workspace. Considering a workspace with multiple spills that can be detected using bounding boxes or image segmentation techniques devised in Chapter 3, we can develop a coverage control strategy based on a planned path. Since this strategy relies on global information of the workspace, we call it centralized path-planning-based coverage control. However, the coverage capacity of a robot has to be modeled beforehand, and the planned path has to be optimized, taking into account the efficiency. Since path-planning-based coverage may rely on GPS signals and external positioning systems, this makes the solution not suitable for indoor environments and other GPS-denied circumstances. Therefore, we further propose distributed coverage control strategies, where each robot relies on its local information. Specifically, we discuss two typical situations for the distributed non-path-planning coverage control: multiple discrete patches in the workspace, and the same problem in the presence of very large patches. What is worth mentioning is that since we are using networked robots, their connection maintenance is essential during the operation. The large-scale patches pose a significant challenge to the inter-robot connection, such that an novel solution without breaking network connection has to be researched.

4.1 Centralized Path Planning-based Coverage Control

4.1.1 Coverage Problem Setting

Water surface robots such as USVs have been studied extensively recently. With aerial images provided by drones or satellites, the USV can be used in a wide

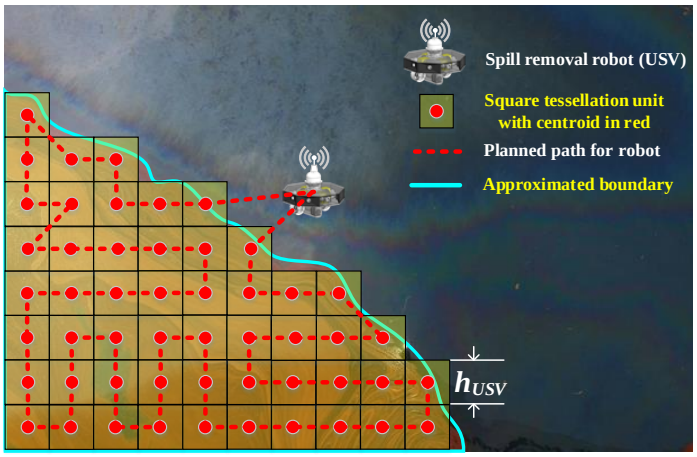


Figure 4.1. A concept of the proposed image processing and model-based spill coverage path planning for a USV. Provided that an aerial image is obtained with remote sensing techniques, an effective path is planned using our proposed image processing and geometric tessellation strategies for the USV to travel and remove the spill. The size of square tessellation block h_{USV} can vary and is decided by the spill removal rate of the USV.

variety of applications, including riverine environmental monitoring (Pinto et al., 2014), flooded open-pit mine 3D reconstruction (Almeida et al., 2016), ocean cleaning (Bella, Belbachir, & Belalem, 2018), marine incident response (Xiao, Dufek, Woodbury, & Murphy, 2017), and water and sediment sampling (Bae et al., 2019).

Nevertheless, a spill cleaning solution based on remote sensing and USV technologies has not yet been fully explored, mostly because of the lack of a functional spill removal model and an adequate planner that enables practical and efficient USV path planning with aerial images. For instance, (N. M. Kakalis & Ventikos, 2008) and (Jung et al., 2017) implemented a monocat and catamaran type of water surface vehicles, respectively, to confront oil spills. However, their traditional conveyor-belt-based appliance can hardly separate oil spills from water, which hinders further increases in efficiency. Moreover, most of the vehicles used in

large-scale removal operations, such as proliferated plankton spill processing and oceanic oil leakage restoration, are designed to be human-operated. No global information is utilized to enable autonomy. In practice, the boundary of the spill can be extremely coarse due to its own molecular characteristics and interaction with water, which presents a substantial challenge for the realization of a complete coverage operation. A practical solution for coverage control in the presence of a coarse boundary is rarely researched.

We propose an effective path planner working in a centralized way that can generate an efficient path for a USV to cover a spill with aerial images provided from remote sensing. To localize the spill area to be covered from an image, spill boundary extraction and approximation are implemented via image processing. The processing capacity of the USV in moving has to be considered, and the total length of the planned path should be minimized for the purpose of saving energy. The concept of the proposed path planner is depicted in Figure 4.1, where an efficient path is generated from the aerial spill image.

4.1.2 Spill Processing Model with a USV Prototype

In order to show our contribution in determining the moving speed and operation range of the USV, the prototype of a generic USV needs to be demonstrated. We adopt a common two-propeller-driven USV as the end effector for our developed planner, and we conceptually develop a polygonal shape USV platform equipped with eight evenly distributed vacuum nozzles along its edge, as shown in Figure 4.2. This conceptual platform is inspired by our previous work (Jo, Hoashi, et al., 2019; Jo, Park, Hoashi, & Min, 2019). When the water pump runs and causes greater fluid pressure at the bottom of the robot than on the surface of the water, the spill on the surface can be sucked into the separator inside the robot for segregation.

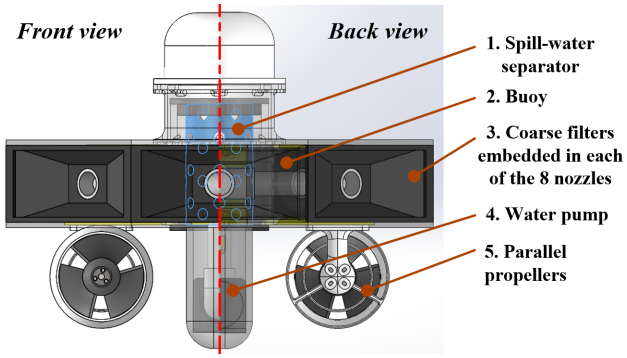


Figure 4.2. Conceptual design of the spill removal USV. The USV uses a water pump to create suction and absorb spills coming from the eight nozzles with both coarse filters and a separator.

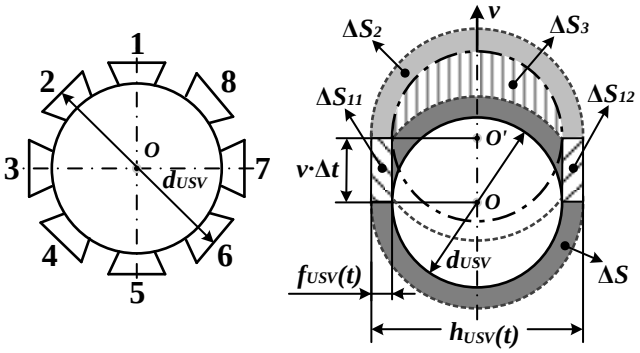


Figure 4.3. The nozzle layout of a USV. The left figure shows the indices of nozzles, while the right one shows the spill collected (shadowed areas) when this robot is maneuvering with velocity v.

A proper processing model is indispensable in determining the maximum speed for a USV, according to capacity. As shown in the prototype of USV in Figure 4.2 and Figure 4.3 (left), the diameter designed for USV is $d_{USV} = 350$ mm, and eight nozzles are arranged along the robot body to enable the spill removal operation.

When a USV is cleaning spills, it should not exceed the maximum power of the water pump or travel too fast so that it remains able to create steady vacuum

pressure and separate spill from water in a timely manner. Obviously, a larger USV has a higher capability for spill removal. However, the size of the USV depends on many factors, including waterproofing and security of other parts, and is therefore restricted. Because of the size restriction, too much spill absorbed at one time may block the nozzles or damage the separator. Therefore, the maximum linear velocity will be calculated under the maximum capacity of the robot, which is denoted as $\mathcal{V}$ in this paper.

In practice, not every nozzle has a chance to collect spill in processing. Hence, we introduce a coefficient γ_{nz} that represents the ratio of operating nozzles n_{nz} to the total number, i.e., $\gamma_{nz} = n_{nz}/8$. It is evident that the spill collected by an individual USV can be formulated as $\gamma_{nz}\mathcal{V}$ in unit time. When a USV is static and has velocity $v = 0$, the width $f_{USV}(t)$ of the coverage area has to follow a static model shown in Theorem 4.1.1, which is depicted with a dark shadowed ring ΔS in Figure 4.3 (right).

Theorem 4.1.1 (Spill Processing Model in Static) *If the USV stays still in place ($v = 0$) and is fully surrounded by the spill, the $f_{USV}(t)$ follows the conclusion below:*

$$f_{USV}(t) = \sqrt{\frac{\mathcal{V}}{\pi}t + c_1} - c_0, \forall t \geq 0 \tag{4.1}$$

where $c_0 = \frac{d_{USV}}{2}$ and $c_1 = c_0^2$. t refers to the elapsed time.

However, since the USV is moving dynamically with a velocity of v, an additional spill can spontaneously rush into the robot, in addition to the spill collected with the vacuum water pump, creating an extra burden for the separator. Thus, we need a dynamic spill processing model to determine the optimal combination of spill absorbing rate and moving velocity v. We assume that when the robot is moving forward, it cannot capture spill located behind it, as spill travels slowly. Thus, nozzles {1,2,3,7,8}, shown in Figure 4.3 (left), are doing most of the collection work, meaning $\gamma_{nz} = 0.625$. Considering those facts along with Theorem 4.1.1, we build the dynamic processing model as below.

Theorem 4.1.2 (Spill Processing Model in Moving) *If the USV moves with a velocity $v \neq 0$ in a fully spill-surrounded area, the spill collected with respect to the capacity of a USV has to satisfy the following equation set:*

$$\sum_i \Delta S_i = \Delta S_{11} + \Delta S_{12} + \Delta S_2 + \Delta S_3 = \gamma_{nz}\mathcal{V} \cdot t = 0.625\mathcal{V} \cdot t, \tag{4.2}$$

where

$$\Delta S_{11} = \Delta S_{12} = v(t) \cdot t \cdot f_{USV}(t),$$
$$\Delta S_2 = \frac{\pi}{2}\left[(f_{USV}(t) + c_0)^2 - c_0^2\right],$$

$$\Delta S_3 = \begin{cases} c_0 v(t) \cdot t \sin c_2 + (\pi - c_2)c_0^2 - c_3(f_{USV}(t) + c_0)^2, \\ \qquad\qquad\qquad\qquad if v(t)t \leq 2c_0 + f_{USV}(t), \\ \approx 2c_0 v(t)t, \qquad\qquad otherwise. \end{cases} \tag{4.3}$$

$$c_0 = \frac{d_{USV}}{2},$$
$$c_2 = \arccos\left(\frac{c_0^2 + (t \cdot v(t))^2 - (f_{USV}(t) + c_0)^2}{2c_0 t \cdot v(t)}\right),$$
$$c_3 = \arccos\left(\frac{(f_{USV}(t) + c_0)^2 + (t \cdot v(t))^2 - c_0^2}{2(f_{USV}(t) + c_0)^2 t \cdot v(t)}\right).$$

From Theorem 4.1.2, we can see the relationship between robot moving velocity v and spill coverage radius f_{USV} that leads to the maximum amount of spill cleaned. In order to reduce the computation cost and simplify further analysis, we seek for an approximate dynamic spill processing model that shows this relationship in a more straightforward way.

Lemma 4.1.3 *The relationship between robot moving velocity v and spill area coverage radius f_{USV} can be approximated as*

$$0.625\mathcal{V} = \begin{cases} 2vf_{USV} + c_4 v + c_5\mathcal{V}, \ if v(t)t \leq 2c_0 + f_{USV}, \\ 2vf_{USV} + 2c_0 v + 0.5\mathcal{V}, \ otherwise, \end{cases} \tag{4.4}$$

where $c_4 = c_0 \sin c_2$, and $c_5 = \frac{1}{2} - \frac{c_3}{\pi}$.

Proof We examine the Theorem 4.1.2 and can show the following conclusions with ease:

$$0 \leq c_2, c_3 \leq \pi;$$
$$\Delta S_{11}, \Delta S_{12} = \mathcal{O}(t^{1.5}); \ \Delta S_2 = \mathcal{O}(t); \ \Delta S_3 = \mathcal{O}(t),$$

where $\mathcal{O}(\cdot)$ means asymptotic upper bound.

As time elapses ($t \to \infty$), we only retain those terms ΔS_x or part of a term ΔS_x with a higher order of t, thus (4.2) becomes

$$\begin{aligned} \sum_i \Delta S_i &= 0.625\mathcal{V} \cdot t \\ &= \begin{cases} 2vtf_{USV} + (\frac{\pi}{2} - c_3)f_{USV}^2 + c_0 vt \sin c_2, \\ \qquad\qquad\qquad\qquad \text{if } v(t)t \leq 2c_0 + f_{USV}(t), \\ 2vtf_{USV} + \frac{\pi}{2}f_{USV}^2 + 2c_0 vt, \qquad \text{otherwise.} \end{cases} \end{aligned}$$

Eliminate the factor t from both sides of the second equal sign and let $c_4 = c_0 \sin c_2$, $c_5 = \frac{1}{2} - \frac{c_3}{\pi}$, we show (4.4). This concludes Lemma 4.1.3. ■

From (4.2) and (4.4), we can see that the faster the USV moves forward, the more spill is collected passively, meaning more spill is fed into the robot while less spill is collected with suction. By contrast, if the robot moves slowly, less spill is flooded into the robot, but more is collected actively by the pumps. As evidence,

the spill-absorbing border f_{USV} grows farther apart during a period of time when the robot moves slowly.

As we concluded earlier, the robot cannot exceed a maximum speed of v_{max} because excessive spill collected may damage the robot and cause the task to fail. Such speed limitation v_{max} can be determined from the result of Lemma 4.1.3. It is preferable for the USV to move at as high a speed as possible when processing because this results in a shorter operation time. As the speed is bounded by v_{max}, we want the robot to maneuver at the maximum speed v_{max} without detected risks.

Lemma 4.1.4 *The maximum speed v_{max} allowed for USV processing within a non-hollow and symmetrical spill is*

$$v_{max} = \frac{1}{16c_0}\mathcal{V}. \tag{4.5}$$

Proof Considering (4.4), since the f_{USV} represents the spill actively collected by a USV, the faster USV moves, the lower the f_{USV} value is. Then we can simply let $f_{USV} \to 0$, and thus we obtain $\widetilde{v}_{max}$ as below:

$$\widetilde{v}_{max} = \begin{cases} \frac{5-5c_5}{8c_4}\mathcal{V}, & \text{if } \widetilde{v}(t)t \leq 2c_0, \\ \frac{1}{16c_0}\mathcal{V}, & \text{otherwise.} \end{cases}$$

We then decide $\frac{1}{16c_0}\mathcal{V}$ is the candidate of v_{max} because it is derived on the basis that a robot moves with a higher speed, according to (4.3) and (4.4). This concludes the proof. ■

With a USV traveling at the maximum speed v_{max}, the corresponding f^*_{USV} can be obtained from (4.4), which will be used to determine the maximum size of a packing shape in workspace tessellation. However, if the real maximum speed of the USV is bounded mechanically and less than the computed v_{max}, the actual f_{USV}

value may be greater than the computed f^*_{USV}. Ultimately, the width that the USV covers in maneuvering is $h_{USV} = 2f_{USV} + d_{USV}$. Our solution shows strong adaptation to diverse h_{USV} value, as demonstrated in Figure 4.6.

4.1.3 Geometric Tessellation and Path Planning of the USV

Geometric Tessellation

We perform geometric tessellation with squares to the areas of interest, as shown in Figure 3.2, as squares can realize a higher coverage rate than circles (Luo, Bae, & Min, 2018). The squares are of uniform dimension, and the length of side equals h_{USV}, which is proposed in Section 4.1.2. The tessellated areas can be covered by the USV if it visits all the centroids of the tessellation blocks, namely waypoints. Therefore, the spill coverage problem is formulated as a TSP. The shortest path for such travel can be obtained by solving this TSP, and an efficient solver for the problem will be elaborated in Section 4.1.3.

Multi-Goal Path Planning Problem and SOM Approach

In this section, we discuss the generation of an efficient path for a USV visiting all the tessellation blocks and performing spill removal operations. Since we want the USV to return to the original location after the operation for the purpose of replacing the filters and recharging the robot, a closed path is desired. Provided that the centroids of tessellation blocks are known, this becomes a multi-goal path planning problem. The concept of multi-goal path planning is defined in terms of finding the shortest closed path for a given set of goals in the workspace. This problem is inspired by the planning of robotic manipulators, in which multiple goals have to be attained by the robot in an effective time period (Danner & Kavraki, 2000).

The problem of multi-goal path planning in spill-cleaning USV can be conceived as a TSP where each tessellation square represents a domain or goal point. As the TSP is modeled as a uni-directed and weighted graph leading to higher computational complexity, a property of the self-organizing map (SOM) method is utilized to obtain a near-optimal solution to it.

SOM is an artificial neural network-based solution that can realize dimensionality reduction using competitive learning (Villmann & Bauer, 1998). The update function for neuron v with the weight vector of Euclidean distance W_v in SOM is shown below:

$$W_v(s+1) = W_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (D(t) - W_v(s)), \tag{4.6}$$

where s refers to the step index, t refers to the training sample index, u represents the index of the best matching unit for the input vector $D(t)$, $\alpha(s)$ is the learning coefficient, and $\theta(u, v, s)$ is the neighborhood function reflecting the distance between neurons u and v in the step indexed by s.

The SOM approach is widely used in meteorology and oceanography (Liu & Weisberg, 2011) and other geological analyses. In this work, the SOM approach is used in planning the path for spill coverage, which results are shown in Section 4.2.5. The current study makes a first attempt towards understanding the effectiveness of the proposed SOM path planner for the concept of spill cleaning.

Benchmarking

In order to benchmark the proposed SOM-based path planner, it is compared against the greedy algorithm (Bednorz, 2008) and 2-Opt algorithm (Englert, Röglin, & Vöcking, 2007). Both the greedy algorithm and the 2-Opt algorithm are well-studied and widely applied in TSP-related cases. 2-Opt is one of the most basic and diffusely used local search heuristics for the TSP. Additionally, 2-Opt shows its

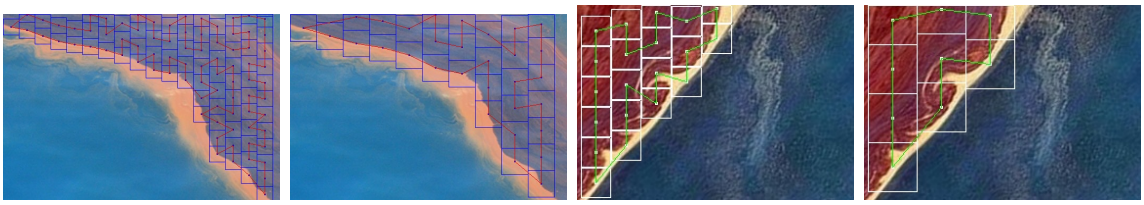
Table 4.1.

Computation costs and outcomes of path planning with figures from Figure 3.2 using SOM approach ($h_{USV} = 40$ pixels).

Metrics	***image1***	***image2***	***image3***	***image4***	***image5***
Computation time (s)	76.18	83.36	62.51	98.38	105.97
Neurons created #	392	504	104	1112	1416
Iterations #	19901	20738	15478	23376	24181
Total length of path $(pixels)$	2080.779	2643.399	560.871	5824.980	7330.534

significance, particularly by achieving good results of TSP in terms of running time and length of the path.

4.1.4 Simulation Experiments for Path-Planning-based Coverage Control



(a) *image1* ($h_{USV} = 30$ pixels) (b) *image1* ($h_{USV} = 50$ pixels) (c) *image3* ($h_{USV} = 30$ pixels) (d) *image3* ($h_{USV} = 50$ pixels)

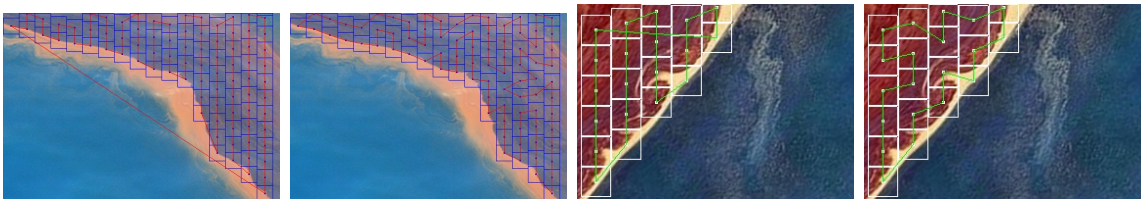
Figure 4.4. Planned paths for *image1* and *image3* using SOM but with different h_{USV} values.

The generated paths for each image in Figure 3.2 are shown in Figure 4.6 using SOM approach with $h_{USV} = 40$ pixels. The computation cost and other facts of path planning after running on Jetson Nano are summarized in Table 4.1. For an image with more pixels, it deserves more tessellation blocks as the image represents a larger physical area, and hence higher computation cost. For instance, it took 105.97 seconds for *image5* to generate a sub-optimal path and meanwhile, more neurons and iteration times are needed. However, *image3* took 105.97 seconds to get

Table 4.2.

*Computation costs and outcomes of path planning with image*1 *and image*3 *from Figure 3.2 with* $h_{USV} = \{30, 40, \text{ and } 50\}$ *pixels by using SOM, greedy algorithm, and 2-Opt algorithm.*

Images	h_{USV} **value**	**SOM**				**Greedy**		**2-Opt**	
		$\boldsymbol{C'time\,(s)}$	$\boldsymbol{N's\#}$	$\boldsymbol{Itr's\,\#}$	$\boldsymbol{P'\,length}$	$\boldsymbol{C'time\,(s)}$	$\boldsymbol{P'\,length}$	$\boldsymbol{C'time\,(s)}$	$\boldsymbol{P'\,length}$
***image*1**	30	89.09	672	21697	2666.976	2.39	3147.935	90.05	2660.207
	40	76.18	392	19901	2080.779	0.46	2502.363	18.78	2102.835
	50	72.98	248	18375	1681.480	0.11	2030.552	4.44	1667.099
***image*3**	30	66.83	160	16914	654.213	0.03	618.911	1.09	655.774
	40	62.51	104	15478	560.871	0.01	506.634	0.3	560.472
	50	58.26	72	14235	480.786	0.01	417.799	0.09	480.786



(a) *image1* using greedy algorithm (b) *image1* using 2-Opt algorithm (c) *image3* using greedy algorithm (d) *image3* using 2-Opt algorithm

Figure 4.5. Planned paths for *image1* and *image3* using greedy algorithm and 2-Opt algorithm, with a uniform $h_{USV} = 30$ pixels.

the path while using only 8% number of neurons of *image5*. We did not specify the starting point of the USV in Figure 4.6, as the USV may start from any tessellation block. Even if the USV is outside the spill, its traveling distance to the closest block is negligible comparing to the overall path length.

A wider operation range h_{USV} helps reduce the number of tessellation blocks as well as computation costs. The outcomes of having $h_{USV} = \{30, 40, \text{and } 50\}$ are shown in Table 4.2, after testing with *image1* and *image3*. Beside less computation time, denoted as $C'\,time$ in Table 4.2, a greater h_{USV} value also leads to a shorter path length, denoted as $P'\,length$ in Table 4.2. The planned paths of USV by having different h_{USV} values are depicted in Figure 4.4.

To evaluate the performance of SOM in generating the sub-optimal path, we also present the results by applying the greedy algorithm and 2-Opt algorithm. We still test with *image1* and *image3*. The quantitative results are presented in the rightmost part of Table 4.2, while the planned paths are presented in Figure 4.5. By comparing Figure 4.5 with Figure 4.4(a) and Figure 4.4(c), we can see that the SOM approach generates a smoother path than the other two, although it takes much more computation time. The greedy algorithm typically yields the longest path length, yet the shortest computation time. But for the cases with very few tessellation blocks, the greedy algorithm may yield a better solution, as indicated by *image*3 in Table 4.2. The 2-Opt algorithm maintains a good balance between

computation time and total path length, but it becomes slower than SOM if more tessellation squares are involved. We would suggest using the greedy algorithm for computation-time-sensitive tasks, and the 2-Opt algorithm for situations with fewer tessellation blocks or waypoints. The SOM approach can usually yield a better solution than many other solutions, and this further strengthens its wide application in oceanography and geographic analysis.

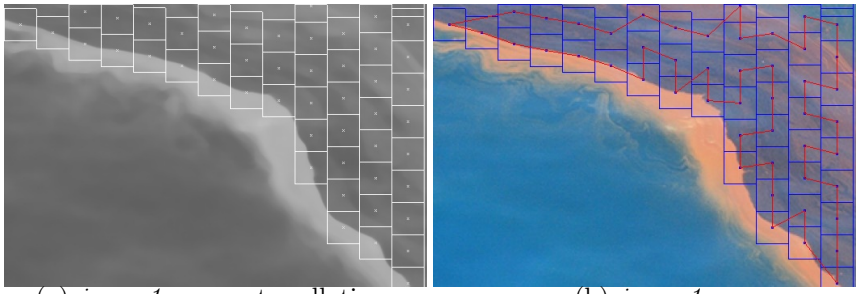
The generated paths for the images shown in Figure 3.2 are shown below in Figure 4.6.

4.2 Distributed Coverage Control Without Path Planning

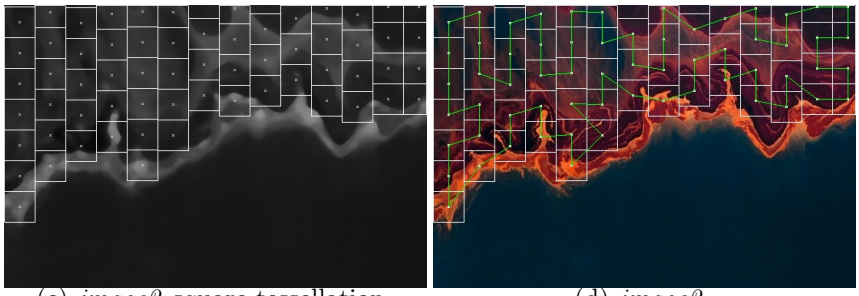
We first present a coverage manifold model and then elaborate the collective coverage control law. For the first aforementioned situation, we propose an Asymptotic Boundary Shrink Control (ABSC) algorithm in (Luo et al., (Under review)), while for the second situation, we propose a pivot-robot-based coverage (PRBC) algorithm. Their stability and convergence proof, as well as simulation experiment validation, are presented immediately after elaboration of each algorithm.

4.2.1 Coverage Problem Setting

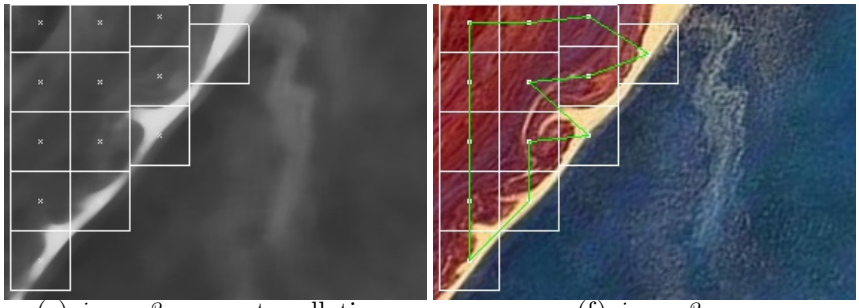
The goal of multi-robot coverage of a patch is for the robots to collectively traverse and remove the patch, as shown in Figure 4.7. In the case of multiple patches in the workspace, we apply allocation as described in Section 3.2.



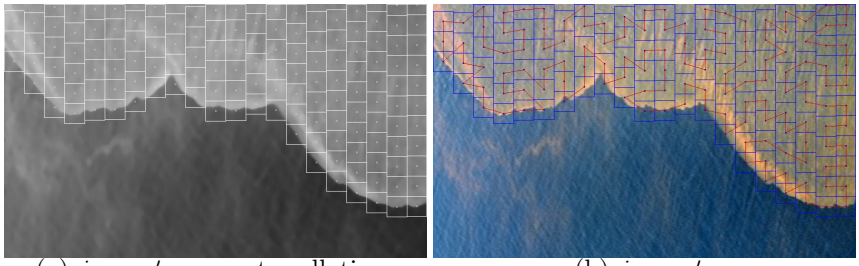
(a) *image1* square tessellation (b) *image1*



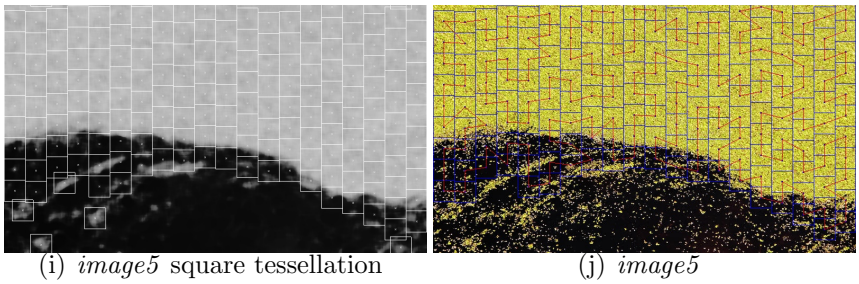
(c) *image2* square tessellation (d) *image2*



(e) *image3* square tessellation (f) *image3*



(g) *image4* square tessellation (h) *image4*



(i) *image5* square tessellation (j) *image5*

Figure 4.6. The procedure for image segmentation using the initial five images. The last column demonstrates the results after applying square tessellation with $h_{USV} =$ 40 pixels.

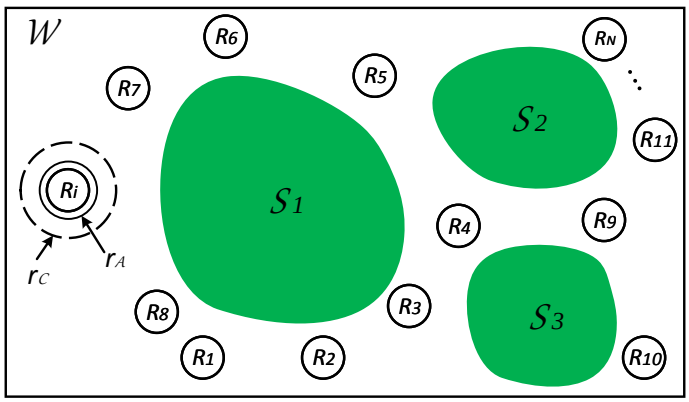


Figure 4.7. A demonstrative figure showing the workspace $\mathcal{W}$ of coverage, where three patches, $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$, are surrounded by N number of distributed robots. For each robot R_i, $r_{\mathcal{A}}$ and $r_{\mathcal{S}}$ represent its vision sensor range and the wireless communication range. Note that the ranges and sizes of robot are not in real-scale.

4.2.2 Coverage Manifold Model

We show the coverage model here. When a robot is maneuvering in $\mathcal{S}_i$ and traveled points $\mathrm{A}(x_1, y_1)$ and $\mathrm{B}(x_2, y_2)$ in sequence, the coverage manifold can be formulated as:

$$\Delta\mathcal{C} = \{\mathrm{M}(x,y) : (0 < \mathbf{AM}\cdot\mathbf{AB} < \mathbf{AB}\cdot\mathbf{AB}) \cap (0 < \mathbf{AM}\cdot\mathbf{AD} < \mathbf{AD}\cdot\mathbf{AD})\}, \quad (4.7)$$

$$\mathrm{D}(x_4, y_4) = \Big(-\sqrt{\|\mathbf{AD}\|^2 - y_4^2}\,,\ \frac{\|\mathbf{AB}\|^2 + \|\mathbf{AD}\|^2 - \|\mathbf{BD}\|^2}{2\|\mathbf{AD}\|} \Big) \quad (4.8)$$

where M is a point in $\Delta\mathcal{C}$, and $\|\mathbf{AD}\| = d$ is the horizontal coverage distance. The coverage manifold is demonstrated in Figure 4.8. Furthermore, assuming a maximal processing capacity $\mathcal{V}$ of a robot in removing the spill, such as the centrifugal volume

to algae spill harvesting and the filtering system capacity to oil spill cleaning, by having $\Delta\mathcal{C} = vd\Delta t = \mathcal{V}\Delta t$, the maximum speed of the robot is bounded by

$$\dot{q}_{max} = u_i = \frac{\mathcal{V}}{d}. \tag{4.9}$$

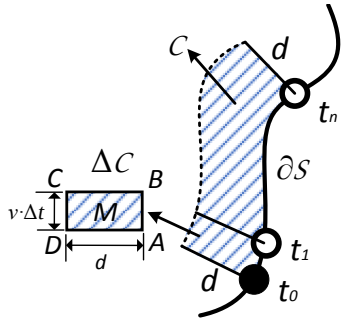


Figure 4.8. Coverage manifold when a robot maneuvers along $\mathcal{S}_i$. An approximated fractional coverage area after Δt time with speed v is shown as $\Delta\mathcal{C}$.

The coverage model (4.7) can be interpreted as the robot cleaning the spill d distance to its left during maneuvering. This model can be applied in existing robots such as (MIT Senseable City Lab, n.d.). Note, d value relies on the curvature of the trajectory and can be approximated based on the contour of the spill. We chose left-side coverage for the reason that we want all the robots to travel uniformly counterclockwise to avoid collisions and realize distributed coverage behavior. The details will be elaborated later.

4.2.3 Asymptotic Boundary Shrink Control (ABSC)

Assuming the diameter of each patch in Figure 4.8 does not exceed robot communication range $r_{\mathcal{S}}$, which means the robots can maintain the connection after allocation and deployment, we propose the ABSC method as collective coverage control strategy. The motion of each robot and the relationship between coverage and other states are represented as a finite-state machine shown in Figure 4.9.

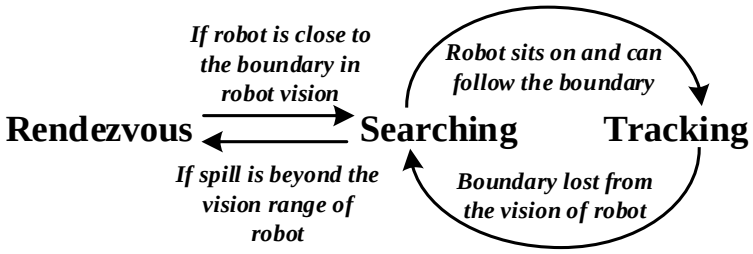


Figure 4.9. The state diagram of the proposed multi-robot ABSC method, and the state transition of robots in workspace $\mathcal{W}$. The collision avoidance hierarchy is prioritized as *Tracking* > *Searching* > *Rendezvous*.

The ABSC takes patch boundary as the input and motivates robots to perform coverage while tracking the boundary. We designed two states, boundary searching and boundary tracking, for ABSC, and they are detailed below.

When each robot is in place, it will then move along the patch boundary with the interior of the patch on the left-hand side. In other words, all the robots move counterclockwise around the perimeter, as demonstrated in Figure 4.10.

For an agent R_i with a field of view (FOV) of span $(-\phi, \phi)$, an angular velocity control law needs to be determined such that the perimeter is detected by an agent. If the perimeter does not initially reside in the FOV, this angular velocity control law is applied, and the robot starts searching the perimeter. The searching state continues until the boundary is maintained within the field of tracking (FOT), a small view angle denoted $\pm\varepsilon_\phi$ and located within the FOV of the robot, as shown in Figure 4.11(a). The FOT is an axial dead zone designed for angular control that avoids frequent oscillation when a robot is advancing. Algorithm 2 presents the control algorithm of perimeter searching in detail.

If a robot R_i finds itself no longer on the boundary $\partial\mathcal{S}$ in searching state, possibly due to an overshoot or large disturbance, i.e., $\mathbf{q}_i - \mathbf{q}_i^{nt} > \epsilon$ where $\mathbf{q}_i^{nt}$ is the nearest point on $\partial\mathcal{S}$ for R_i, it needs to switch back to the deployment state, as shown in Figure 4.9. When a robot re-enters the deployment state, control law

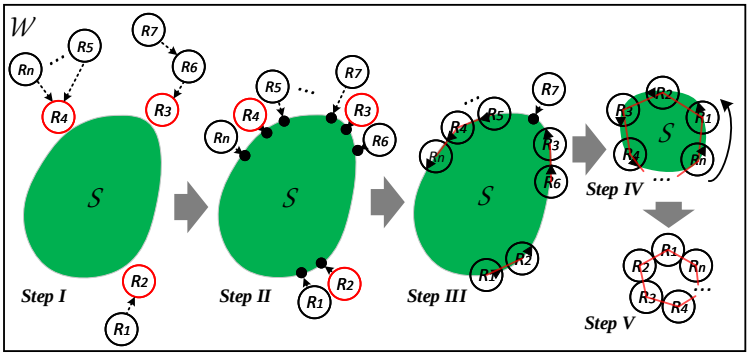
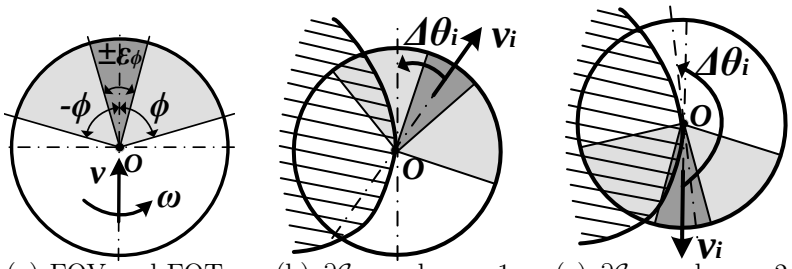


Figure 4.10. A representative figure showing our proposed collective boundary shrink control strategy. The multi-robot hierarchical rendezvous is demonstrated in *Step I* where robots serving as rendezvous points D_m are highlighted with a red circle and the rendezvous hierarchy is indicated with dashed arrows. The robots deployed shall start boundary searching and navigate to the goals when they are in the vicinity of the spill and have no child robot associated, as shown in *Step II*. *Step III* demonstrates boundary tracking state on the completion of boundary searching, and robots maneuver along the boundary to shrink the spill. When robots are moving sequentially and getting closer, the proposed strategy prevents collision, as depicted in *Step IV*. The task is finished in *Step V*, where boundary shrinks to zero and no spill remains. The black dots denote the nearest deployment positions that robots are aiming for in the searching state. The triangles indicate robot moving direction, while the red lines mean the leading and trailing robots reside within APF effective range.

(3.20) can be applied by letting the goal position $\mathbf{q}_g$ be $\mathbf{q}_i^{nt}$. The whole procedure is reflected in Algorithm 2 with Line 2-4.

Algorithm 2 presents the control algorithm of boundary searching in detail. Before boundary searching starts, a few system internal states that may affect Algorithm 2 are decided beforehand as initialization in Algorithm 1.



(a) FOV and FOT (b) $\partial\mathcal{C}$ search case 1 (c) $\partial\mathcal{C}$ search case 2

Figure 4.11. Agent's field of view (FOV) and field of tracking (FOT) definition, and two common patch boundary $\partial\mathcal{C}$ searching cases with angular changes produced by control laws and algorithms.

Algorithm 1: Boundary $\partial\mathcal{S}$ Searching Initialization

```
Initialization /*All flags clear to 0 */
repeat
    if R_i detects itself sitting on the spill boundary ∂S then
        Set on_the_boundary_flag ← 1
        if boundary ∂S resides in FOT then
            Set tracking_enable_flag ← 1
    if ∂S can be detected and is within the FOV then
        Set boundary_within_FOV_flag ← 1
    if no boundary ∂S is detected then
        Set no_boundary_detected_flag ← 1
        Set u_i = 0, w_i = 0
        /*Terminate this program, as no spill is detected within vision
        range r_A */
until robot R_i rotates in the same place for 2π
```

Persistent tracking of spill boundary $\partial\mathcal{S}$ is accomplished by Algorithm 3 and lasts until the coverage work is done. If a robot R_i worked to track the boundary but suddenly misses the boundary in its vision, possibly due to an overshoot or large disturbance, it needs to switch back to the searching state and detect the boundary again, as shown in Figure 4.9. Note that for a drastically changing environment where the spill boundary may escape from the vicinity of the robot and no longer be

Algorithm 2: Boundary $\partial\mathcal{S}$ Searching Algorithm

```
repeat
    if on_the_boundary_flag = 0 then
        if boundary_within_FOV_flag = 1 then
            repeat
                Identify the nearest point q_{g,i} on ∂S to the robot;
                Apply control law (3.20) to drive robot to q_{g,i};
            until on_the_boundary_flag = 1;
    else if on_the_boundary_flag = 1 then
        The robot linear velocity control input u_i = 0;
        if boundary_within_FOV_flag = 1 then
            if spill is on the right hand side of R_i then
                R_i rotates counterclockwise with respect to its centroid for
                 a degree of φ_temp = ω_max · T_c such that ∂S is outside the
                 FOV;
                Clear boundary_within_FOV_flag ← 0;
            else
                repeat
                    Apply control laws (4.12) and (4.13) for direction
                     alignment /* Since ∂S is within FOV and spill is on
                     the left-hand side of R_i */;
                until ∂S is within the FOT;
                Set tracking_enable_flag ← 1
        else if boundary_within_FOV_flag = 0 then
            repeat
                R_i begins rotating counterclockwise with respect to its
                 centroid for a degree of φ_temp ;
            until ∂S is within the FOV;
            Set boundary_within_FOV_flag ← 1;
until tracking_enable_flag = 1;
return Switch to the Tracking state /* End the program, and switch to
 the Tracking state in Algorithm 3 */;
```

detected by vision sensors, the robot team needs to enter rendezvous state and move to the boundary again. Under certain circumstances, the robot can remain in tracking state despite a dynamic spill. More details can be found in Section 4.2.4.

Algorithm 3: Boundary $\partial\mathcal{S}$ Tracking Algorithm

```
1  The robot R_i is located on the boundary of an associated spill
2  The velocity control input of R_i follows constraints of bounded velocity
     and acceleration; repeat
3    if ∂S is within the FOV then
4      if ∂S is within the FOT then
5        Apply control law (4.20) and (4.18)
6        if ‖q_i − q_{D_j}‖ ≤ α · r_C, D_j ∈ D then
7          Enable idle mode
8        else
9          Enable normal mode
10     else
11       Apply control laws (4.12) and (4.13)
12   else
13     return Switch to the Searching state    /* End the program,
         tracking failed, switch to the Searching state in Algorithm 2*/
14 until forever
```

A special case exists that the initial boundary may be shrunk by robots rooted at rendezvous D_i before it can be sensed by robots from $D_j, (j \neq i)$; thus, rendezvous robot D_j loses its vicinity of the boundary. To circumvent this issue, we design two modes for a robot in tracking, namely *normal mode* and *idle mode*. The robot performs boundary shrink control only under normal mode, while tracking without shrinking the boundary in idle mode. One robot R_i switches to idle mode when it resides within an effective range of a static rendezvous robot D_j (with child robots associated), i.e., $\|\mathbf{q}_i - \mathbf{q}_{D_j}\| \leq \alpha \cdot r_{\mathcal{C}}, \alpha \in (0, 1)$. Therefore, the boundary detected by D_j will remain until rendezvous completes. This fact is also reflected in Line 7∼10 of Algorithm 3.

Special consideration should be given in determining the ϕ_{cm} in Algorithm 2. On one hand, it should be as large as possible in order to enlarge the step size, making rotation complete in fewer operation cycles. But on the other hand, it should not be so large that the robot misses the FOV in one rotation. Evidence for

this is found in Algorithm 2, where the robot has to stop moving and indicate "search fails" if failing to detect the perimeter in one full rotation. Therefore, we conclude that $\phi_{temp} = \omega_{max} \cdot T_c \leq 2\phi$. Thus, ω_i is bounded by:

$$\omega_{max} = \frac{2\phi}{T_c} \tag{4.10}$$

If the boundary $\partial\mathcal{C}$ is in the FOV but not the FOT, with the heading angle of agent being θ_i, the angle error with respect to the desired angle $\theta_d(= 0)$ is defined as $\theta_{ei} = \theta_d - \theta_i$. Due to this, one can apply the following proportional-derivative (PD) control law to align the forward direction of the robot with the desired angle θ_d:

$$w_i = \begin{cases} -K_p \cdot \theta_{ei} - K_d \cdot \dot{\theta}_{ei}, \\ \qquad\qquad \text{if } \partial\mathcal{C} \text{ appears in sector } [-\phi, -\varepsilon_\phi) \cup (\varepsilon_\phi, \phi], \\ 0, \qquad \text{if } \partial\mathcal{C} \text{ appears in sector } [-\varepsilon_\phi, \varepsilon_\phi], \end{cases} \tag{4.11}$$

where K_p and K_d are proportional and derivative gains, respectively. Considering the fact that the robot control is discrete in time, the following discrete control law is adopted instead of (4.11):

$$w_i = \begin{cases} -K_p \cdot \theta_{ei}(T) - \frac{K_d}{T_c}\big(\theta_{ei}(T) - \theta_{ei}(T - T_c)\big), \\ \qquad\qquad \text{if } \partial\mathcal{C} \text{ appears in sector } [-\phi, -\varepsilon_\phi) \cup (\varepsilon_\phi, \phi], \\ 0, \qquad \text{if } \partial\mathcal{C} \text{ appears in sector } [-\varepsilon_\phi, \varepsilon_\phi]. \end{cases} \tag{4.12}$$

Finally, by considering the upper limit of angular speed shown in (4.10), we update the controller to be:

$$\|w_i\| = \begin{cases} \|K_p \cdot \theta_{ei}(T) + \frac{K_d}{T_c}\big(\theta_{ei}(T) - \theta_{ei}(T - T_c)\big)\|, \\ \qquad\qquad\qquad\qquad \text{if } \|w_i\| \leq \omega_{max}, \\ \omega_{max}, \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{4.13}$$

Specifically, if the boundary $\partial\mathcal{C}$ is outside the FOV, the proposed control laws are able to rotate the robot until $\partial\mathcal{C}$ falls into the FOV. However, if the $\partial\mathcal{C}$ resides in FOV but the patch is on the right-hand side of the robot, then the robot is heading clockwise. Hence, the robot has to rotate counterclockwise until it catches sight of $\partial\mathcal{C}$ once again, as illustrated in Figure 4.11(c). The whole process is open-loop-based and aims for no specific goal. The proposed ω_{max} guarantees the minimum possible steps to finish searching. When the $\partial\mathcal{C}$ is once again within the FOV, control laws (4.12) and (4.13) can be applied. Both (4.12) and (4.13) are based on a vision sensor feedback available at low cost in terms of implementation. If more accurate and consistent control is needed for more complex environments, one can refer to the visual servoing methods proposed in (Lots, Lane, Trucco, & Chaumette, 2001; Sattar, Giguere, Dudek, & Prahacs, 2005).

When a group of robots is cleaning a patch along the perimeter, a proper control method needs to be proposed to avoid collision between robots but still motivate them to move forward. Such a controller based on the APF method is provided and discussed in this section.

Since workspace $\mathcal{W}$ is assumed to be obstacle-free, attractive potential U_{att} exerted on robot R_i by its leading robot R_{i+1} is sufficient to drive R_i. When R_i is tracking the boundary $\partial\mathcal{C}$, it can measure the length in perimeter rather than line distance between itself and other agents within its FOV. It uses length in perimeter because a closer line distance between two robots does not mean that they are adjacent if tracking a serpentine perimeter. Practically, this measurement can be performed with stereo vision sensors, LIDAR, or high-resolution laser rangefinders, along with the techniques proposed in (Gowal, Prorok, & Martinoli, 2011; Mitiche & Aggarwal, 2014). Suppose there is a function $s = f(\mathbf{q})$ to represent the patch boundary $\partial\mathcal{C}$, where $s \in [0, \|\partial\mathcal{C}\|]$ indicates the length between a reference point and an agent of position $\mathbf{q}$ in the counterclockwise direction. We can show that once a

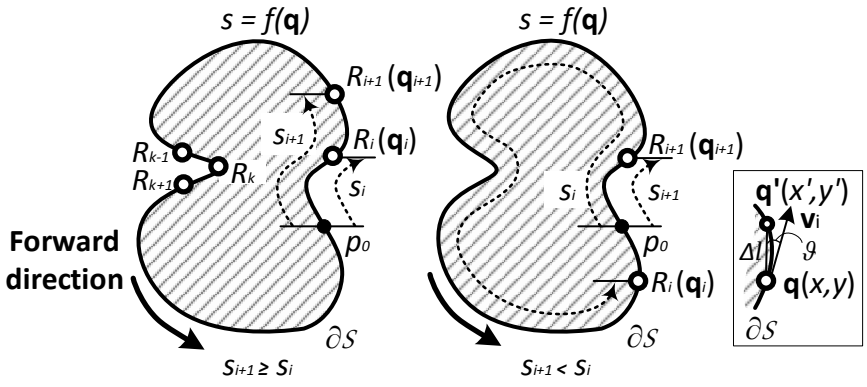


Figure 4.12. The left figure shows two subgroups of adjacent robots, one subgroup consisting of R_{k-1}, R_k, and R_{k+1}, and the other consisting of R_i and R_{i+1}. Both the left and middle figures illustrate the definition of s_i and s_{i+1} for the trailing robot R_i and its leading robot R_{i+1}, respectively. The right figure in the box shows the symbols and their geometrical relationship when a robot is maneuvering along the boundary $\partial\mathcal{S}$.

robot R_{i+1} falls within the FOV of its trailing robot R_i, the distance between these two adjacent robots, namely $l_i = \|q_i, q_{i+1}\|_{\partial\mathcal{C}}$, can be computed by

$$l_i = \begin{cases} s_{i+1} - s_i, & \text{if } s_{i+1} \geq s_i, \\ s_{i+1} - s_i + \|\partial\mathcal{C}\|, & \text{if } s_{i+1} < s_i. \end{cases} \tag{4.14}$$

$s_{i+1} \leq s_i$ can happen depending on where the reference point is, which is shown in Figure 4.12. If the leading robot of R_i is beyond the radius $r_{\mathcal{A}}$ of its FOV, (4.14) does not apply. Therefore it defines a virtual distance l_i^* that includes both situations as below:

$$l_i^* = \begin{cases} l_i, & \text{if } l_i \leq r_{\mathcal{A}}, \\ r_{\mathcal{A}}, & \text{if } l_i > r_{\mathcal{A}} \text{ or } l_i \text{ is unknown.} \end{cases} \tag{4.15}$$

The attractive potential function is defined as

$$U_{att} = \frac{1}{2}\xi_3 l_i^{*2} \tag{4.16}$$

where ξ_3 is a positive scaling factor.

The robot velocity v_i should be in the direction of a negative gradient of U_{att} with respect to s_i, such that

$$v_i = -\nabla_{s_i} U_{att} = \xi_3 l_i^*. \tag{4.17}$$

4.2.4 Theoretical Analysis of the ABSC

Extra factors need to be considered when applying (4.17) as robot velocity control input.

Maximum Linear Speed

It is worth noticing that the robot cannot exceed the maximum linear speed v_{max}. Thus, we should have

$$\mathbf{v}_i = \begin{cases} \xi_3 l_i^*, & \text{if } \|\mathbf{v}_i\| \leq v_{max}, \\ \frac{\xi_3 l_i^* v_{max}}{\|\xi_3 l_i^*\|}, & \text{otherwise.} \end{cases} \tag{4.18}$$

Control Design

Controllers similar to (4.17) are developed in works such as (Kumar, Garg, & Kumar, 2010; G. Zhang et al., 2013); however, they do not consider or fully discuss the control design under unicycle dynamics (3.15). Here, we provide a strategy inspired by (Pickem, Lee, & Egerstedt, 2015) to determine the velocity control input tailored for a unicycle model. Note that the robot acceleration is bounded mechanically for boundary searching and tracking states.

Let (x, y) present the current position of R_i. Assuming a point (x', y') lying on $\partial\mathcal{S}$ and in front of the robot R_i with a small length offset of Δl, we have its global coordinate as $x' = x + \Delta l \cos(\vartheta + \theta_i)$ and $y' = y + \Delta l \sin(\vartheta + \theta_i)$, having ϑ as the angle between orientations of the robot and the potential field $U_{att}(x, y)$. Recall that θ_i is the orientation direction of the robot R_i. The aforementioned symbols and their geometrical relationship can be found in Figure 4.12 (in the box), where the orientation of the potential field is represented as $\mathbf{v}_i$. Practically, (x', y') can be sensed by local vision sensors of R_i. Let $\mathbf{Q} = [u_i, w_i]^T$ be the velocity control input and $\mathbf{v}_i = [v_x^i, v_y^i]^T$ in (4.18). The velocity becomes as follows:

$$\begin{bmatrix} \dot{x'} \\ \dot{y'} \end{bmatrix} = \begin{bmatrix} v_x^i \\ v_y^i \end{bmatrix} = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \Delta l \end{bmatrix} \begin{bmatrix} u_i \\ w_i \end{bmatrix} = \mathbf{G}(\vartheta, \Delta l)\mathbf{Q}. \tag{4.19}$$

Then the control input can be obtained by $\mathbf{Q} = \mathbf{G}^{-1}\mathbf{v}_i$, i.e.,

$$u_i = v_x^i \cos\vartheta + v_y^i \sin\vartheta, \tag{4.20a}$$

$$w_i = \frac{1}{\Delta l}(-v_x \sin\vartheta + v_y \cos\vartheta). \tag{4.20b}$$

Note that the controller (4.20) is bounded using (4.18), where the velocity set $\mathbf{v}_i = [v_x^i, v_y^i]^T$ in (4.20) is derived from (4.18). In the case of coarse measurement or irregular spill shapes such that (4.14) cannot be obtained with precision, cooperative Kalman filters as presented in (F. Zhang & Leonard, 2010) can be utilized to estimate the boundary and determine the control law, when a noisy scalar field is built for the spill.

Speed Convergence

When $l_i^* \to 0$, we need to show the convergence of linear speed. By differentiating $U_{att}(q_i)$ in (4.16) with respect to time t and considering (4.17), we have

$$\dot{U}_{att} = -\xi_3 \dot{l}_i^* = -\xi_3^2 l_i^* = -2\xi_3 U_{att}. \tag{4.21}$$

From (4.21), it shows

$$U_{att} = U_{att}(0) e^{-2\xi_3^2 t}. \tag{4.22}$$

Obviously, the maximum potential $U_{att} = \frac{1}{2}\xi_3 r_{\mathcal{A}}^2$ exists if the distance of perimeter between robot R_i and its leader R_{i+1} is far enough. Therefore, it is preferable if R_{i+1} travels beyond the FOV of R_i. Moreover, the greater ξ_3 is, the faster U_{att} and l_i^* converge to 0. One last significant remark is that from control laws (4.17) and (4.13), we can see that one trailing robot R_i cannot wrap up its leading robot R_{i+1}, because along the perimeter, the attractive potential decreases to *zero* as R_i is approaching R_{i+1}. Thus, linear velocity input also decreases to *zero*.

Robot Convergence after Operation

When cleaning is coming to an end, all the robots associated with a specific patch should then form a loop and converge around a place, as long as the patch remains convex during cleaning. One might ask whether the robots will reduce their speed while moving to the endpoint and eventually stop when they have arrived. A positive conclusion is given in the discussion in Section 4.2.4, which shows that the linear speed $\|u_i\|$ for every agent converges to *zero* when l_i^* approaches *zero*. The whole process stops when the robot cannot find any more patch, or, more specifically, when robot FOV still misses boundary $\partial\mathcal{C}$ after a full rotation in searching state. This implies that the patch has been fully cleaned. More information about the convex patches is provided in Section 4.2.4.

The convergence result may vary based on the dimensions of the robot. When many robots are allocated to a small patch, the convergence is not obvious before and after cleaning as a point, since the robots' dimensions prevent them from getting closer to each other. The stability of multi-robot coordination control in coverage is detailed in Section 4.2.4.

Stability Analysis

We investigate the stability of robot coordination in coverage control. At a certain time $t = T_+$ after long operation, we assume that all robots deployed are in the tracking state and each of them has $l_i \leq r_{\mathcal{A}}$, meaning that it has a leading robot within its view and is within the view of a trailing robot. Defining the n_j number of robots allocated for patch $\mathcal{C}_j$ as $R_1^j, R_2^j, ..., R_{n_j}^j$, the perimeters for patches $[\|\partial\mathcal{C}_1\|, \|\partial\mathcal{C}_2\|, ..., \|\partial\mathcal{C}_M\|]_{t=T_+}$, and $\mathbf{l}^* = [l_1^*, l_2^*, ..., l_N^*]_{t=T_+}^T$ at the current time, the system collective potential energy is provided by

$$V(\mathbf{s}) = V(\mathbf{l}^*) = \sum_{j=1}^{M} \left(\sum_{i=1}^{n_j} U_{att}(i) \right). \tag{4.23}$$

Here, $\mathbf{s} = [s_1, s_2, ..., s_N]^T$, and $U_{att}(i)$ is the attractive potential shown in (4.16).

We define a sparse matrix $Z = [z_{ij}]_{M \times N}$ based on the optimization results of (3.3) to represent the association of each robot to a specific patch. The matrix Z also satisfies constraints (3.4) and (3.6). Obviously, the collective potential (4.23) has to satisfy a boundary constraint as shown below:

$$Z \cdot \mathbf{l}^* = [\|\partial\mathcal{C}_1\|, \|\partial\mathcal{C}_2\|, ..., \|\partial\mathcal{C}_M\|]_{t=T_+}^T. \tag{4.24}$$

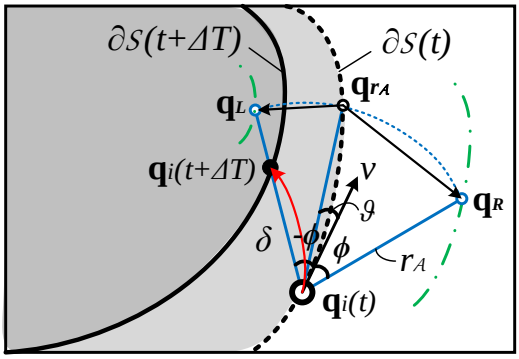


Figure 4.13. The trajectory (red line) of robot R_i working in the boundary tracking state when the boundary is changing. The dashed black curve is the boundary at time t while the solid black curve is the boundary in $t + \Delta T$. The blue lines with a span $(-\phi, \phi)$ indicate the FOV of the robot. The dash-dotted lines in green represent the critical location of boundary escaping the robot vision range from the left and right side, respectively.

Using (4.23) and the result of Section 4.2.4, we have the time derivative of V(**s**) as

$$\begin{aligned}\dot{V}(\mathbf{s}) &= \frac{dV_1}{ds_1} \cdot \frac{ds_1}{dt} + \frac{dV_2}{ds_2} \cdot \frac{ds_2}{dt} + ... + \frac{dV_N}{ds_N} \cdot \frac{ds_N}{dt} \\ &= -\xi_3 \mathbf{l}^* \cdot \mathbf{v} \leq 0\end{aligned}$$

where $\mathbf{v} = [v_1, v_2, ..., v_N]^T$. Thus, we show that the Lyapunov function $V(\mathbf{s})$ is negative semi-definite, and $\dot{V} = 0 \; iff \; \mathbf{l}^* = 0$, i.e., $s_i = s_{i+1}$ for any two adjacent robots tracking the same patch and $\|\partial \mathcal{C}_i\| = 0$, according to (4.14), (4.15), and (4.24). Meanwhile, the robot velocity v_i is bounded by (4.9) and decays with l_i^*. Therefore, according to LaSalle's invariance principle (LaSalle, 1976), those robots deployed to a specific patch will asymptotically converge to a point when the patch is removed.

Dynamic Spill Applicability

In real cases, the boundary of a spill cannot always be static during the operation. In this section, we prove that our proposed boundary shrink control strategy can also handle a case when the spill boundary is moving.

The multi-robot rendezvous algorithm deals with a moving boundary by motivating the rendezvous robot D_m to move according to the boundary. Therefore, the child robots will eventually gather in the vicinity of the changing boundary. Such behavior that has a dynamic rendezvous point is named robotic *herding* and validated in (Parasuraman, Kim, Luo, & Min, 2018).

In boundary searching state, our solution can handle a moving boundary case by simply remaining in this state and rerunning Algorithm 2, provided that the boundary does not move faster than robot maximal speed v_{max} in the case that the boundary escapes from robot vision range $r_{\mathcal{A}}$ when the robot is chasing the goal position on the boundary.

If boundary moving happens in the boundary tracking state, then our Algorithm 3 can tackle this issue, even without switching to boundary searching state if the changing boundary appears inside the FOV, as shown in Figure 4.13. Since the changing boundary appears inside robot FOV, the trajectory of a robot working in the boundary tracking state and a dynamic boundary is shown in the red line in Figure 4.13. In the same figure, the robot's current position is $\mathbf{q}_i(t)$, and the goal position on the dynamic boundary for the robot to track denotes $\mathbf{q}_i(t + \Delta T)$, whose coordinate can be calculated as

$$\mathbf{q}_i(t + \Delta T) = \begin{bmatrix} x(t + \Delta T) \\ y(t + \Delta T) \end{bmatrix} = \begin{bmatrix} \delta \cos(\phi + \theta_i) + x(t) \\ \delta \sin(\phi + \theta_i) + y(t) \end{bmatrix}, \tag{4.25}$$

where $[x(t), y(t)]^T$ is the coordinate of $\mathbf{q}_i(t)$, and δ is the distance between $\mathbf{q}_i(t)$ and $\mathbf{q}_i(t + \Delta T)$, which can be determined by image processing techniques such as

(Saxena, Chung, & Ng, 2006). Using control law (3.20) by replacing $\mathbf{q}_g$ in it with $\mathbf{q}_i(t + \Delta T)$, the robot can track the dynamic boundary.

For a robot working under boundary tracking state, as shown in Figure 4.13, the spill boundary can escape the robot vision range from either the left or right side if $\mathbf{q}_{r_\mathcal{A}}$, the farthest point on the boundary that is within robot vision range, moves over point $\mathbf{q}_L$ or $\mathbf{q}_R$. Given this fact, we can decide the maximum speed for the moving spill, $v_{max}^{\partial\mathcal{S}}$, such that the boundary always resides in the robot's FOV. $v_{max}^{\partial\mathcal{S}}$ can be determined as $v_{max}^{\partial\mathcal{S}} = \min\{\frac{\|\mathbf{q}_L - \mathbf{q}_{r_\mathcal{A}}\|}{T_c}, \frac{\|\mathbf{q}_R - \mathbf{q}_{r_\mathcal{A}}\|}{T_c}\} = \min\{\frac{(\phi-\vartheta)r_\mathcal{A}}{T_c}, \frac{(\phi+\vartheta)r_\mathcal{A}}{T_c}\}$. Here, recall that T_c is the robot control system cycle time.

Non-convex Patch

The patch mentioned above is assumed to be convex during operation. However, as can be seen from the experiment videos, patch 3 failed to remain convex while robots were cleaning the patch. Fortunately, the task was still completed because the patch was not completely broken. It is possible to overlook this situation and apply our solution to non-convex situations. However, the challenge is how to make sure every separate piece of patch has at least one robot associated with it when an original patch is broken into several pieces, since the narrowest place is eroded with cleaning. UAV can again be applied to reallocate the robots or create allocation methods that promote robot dispersion and prevent many robots from gathering at one piece (Batalin & Sukhatme, 2002). Enabling ad hoc networking among robots may also help to achieve convergence while the robots are assembling for cleaning.

4.2.5 Simulation Experiments for ABSC

To evaluate the proposed coverage operating robot control methods, extensive scaled-down simulation experiments were conducted in Robotarium

MATLAB platform (Pickem et al., 2017), which also supports hardware experiments with GRITSbot robots.

In the 2D arena of size 3m × 3m of Robotarium, we designed four spills with different geometric features, as shown in Figure 3.7. The diverse geometry of presented spills will help to validate the capability of the proposed method. In addition, Table 3.2 shows the index and area of the spill patches, and v_{max} is made equal to 0.01 m/s. Since the robot allocation is already elaborated in Section 3.4.2, we focus on ABSC in this section.

Evaluation Metrics

The following three metrics were used to evaluate the performance of the proposed coverage control strategy in our experiments.

1. Lyapunov candidate function (Convergence):

$$L_x = \sum_{R_i, R_j \in \mathcal{S}_x, i \neq j} \|q_i - q_j\|. \tag{4.26}$$

   If there is only one robot allocated for a spill $\mathcal{S}_x$, the Lyapunov candidate function is revised as

$$L_x = \sum_{R_i \in \mathcal{S}_x} \|q_i - c_x\|, \tag{4.27}$$

   where c_x is the computational centroid of the patch.

2. Total distance traveled by all robots after covering each spill (Distance):

$$D_{sum} = \sum_{\mathcal{S}_x \in \mathcal{W}} \sum_{k=1}^{k_{max}} \|q_i(k+1) - q_i(k)\|. \tag{4.28}$$

3. Number of iterations ($k_{stop} \leq k_{max}$) to reach the following stop condition:

$$\text{Stop at } k_{stop} \text{ if the current area } A_x \leq A_{min}, \forall \mathcal{S}_x \in \mathcal{W}. \tag{4.29}$$

Here, A_{min} is defined to be 1% of the original area of each spill.

Experiment Cases and Results with Relatively Large Vision Sensing Range $r_{\mathcal{A}}$

We validate our proposed solution in terms of efficacy, efficiency, scalability, and convergence. First, we select a case where a relatively large vision sensing range $r_{\mathcal{A}} = 1$m is adopted such that every robot can detect the boundary within its vision range. Therefore, every robot itself becomes a rendezvous point $D_i, i \leq N$ and is designated to a specific spill by following a nearest-neighbor method. This case is to show that a larger vision range helps reduce rendezvous hierarchy and hence the total operation time. Since $N = 5$ is the minimum reasonable number for the experiment environment, and we want to scale up N with the same interval, six scenarios with $N = \{5, 10, 15, 20, 25, \text{ and } 40\}$ are chosen. We first present the detailed experiment results of three sample cases that are most representative, $N = \{5, 10, \text{ and } 40\}$, as shown in Figure 4.14 and Table 3.3.

Case 1 (N = 5)

In this case, a minimum number of *five* robots was tested, and their allocation is shown in Table 3.3. After removing the spill, the robots ultimately should converge at a point near the centroid of the spill. As documented in the top field of Table 3.3, every spill was allocated with one robot, except for spill 3. The area evolution chart in the leftmost column of Figure 4.14 shows that the completion progress of coverage was steadily declining. Since spills 1, 2, and 4 were allocated with only one robot each, their Lyapunov candidate function was defined according to (4.27). The convergence figure in Figure 4.14 (third row) also shows the sum of the Lyapunov candidate values of all the spills, i.e., $\sum_{\mathcal{S}_x \in \mathcal{W}} L_x$, and therefore describes the overall performance of robot convergence in the workspace. According to the convergence figure, all the robots converged to their endpoints while covering, which was confirmed by the Lyapunov candidate function L_x

approaching to *zero*. However, even though it is obvious that the all the robots were approaching an end asymptotically, as shown in the convergence figure of $N = 5$, the convergence of robots associated with spill 3 was oscillating, as indicated by the dotted-dash line. The reason was that the distance between the two allocated robots varied when they were moving along the perimeter of the quadrilateral.

Case 2 (N = 10)

In this case, we want to show whether spill can be removed in a shorter time than the first case, and whether robots still converge at some points. The outcome can be seen from the trajectory figure in the second column of Figure 4.14. The coverage to A_{min} ended at an iteration $\max\{k_{stop}\} = 10240$. Given a higher ratio of robots in the workspace area, the completion time was shortened by over 50%, compared with Case 1. The robots still converged after a certain operation time, as indicated in the middle field of Table 3.3. Additionally, the convergence of this case was much smoother than the previous case because every robot had a shorter trajectory before final convergence.

Case 3 (N = 40)

In this case, we want to show the scalability of our proposed solution and how deploying a greater number of robots affects the theoretical properties. The convergence curve was smoother in this scenario than in any of the others, and reduced to *zero* faster than it did in any other cases. Given a higher ratio of robots in the workspace area, the completion time was shortened by over 85%, compared with Case 1. Since the paths traveled by the robots became much shorter than in previous cases, the robots were converging in a more straightforward way. This fact can be observed from the convergence chart in the rightmost column of Figure 4.14. In addition, because of the physical dimensional restraint of robots, the sum of convergence in this case did not amount to *zero* when the task approached its end.

The numerical results of allocation and convergence are shown at the bottom of Table 3.3. From the trajectory figure in Figure 4.14, we can see that twisted trajectories developed in R_{21} and R_{36} during the deployment period. This happened because R_{21} and R_{36} were trying to avoid colliding with approaching robots R_{22} and R_{15}, respectively. Robots R_{22} and R_{15} were working in the tracking state, which was prioritized over the deployment state that R_{21} and R_{36} were in. This validated the efficacy of our proposed APF-based collision avoidance controllers in Chapter 3.

Through these experiments, we validated the efficacy, efficiency, and scalability of our proposed solution. As long as the initial view of the workspace can be obtained, our solution can be applied in workspaces of any scale.

Further Discussion

Our analysis of six cases, $N = \{5, 10, 15, 20, 25, \text{ and } 40\}$, reveals some significant outcomes. First, as the total number of robots increased, the Lyapunov candidate function value at $max\{D_{sum}\}$ also increased due to the physical restraint of robot dimensions. Secondly, the timing of spill coverage was tested with different numbers of robots. Figure 4.15(a) shows that the spill was removed steadily as time elapsed. The completion time was reduced significantly if more robots were deployed to the task. However, when the number of robots became greater than 15, completion time was not significantly reduced, as can be seen from $max\{k_{stop}\}$ in Figure 4.15(a).

A greater number of robots N yields a shorter completion time. Nevertheless, this may not be a concern in many cases. Since the number of robots is usually limited by necessity, more robots deployed involves greater financial expense, physical restraint, and time lost with avoiding each other. Moreover, an excessive number of robots results in unnecessary energy consumption, which is reflected in $\sum_{i=1}^{M} D_{sum}(i)$, the overall distance traveled by all the deployed robots. Our experiment results show the changes of both k_{stop} and $\sum_{i=1}^{M} D_{sum}(i)$ while robot

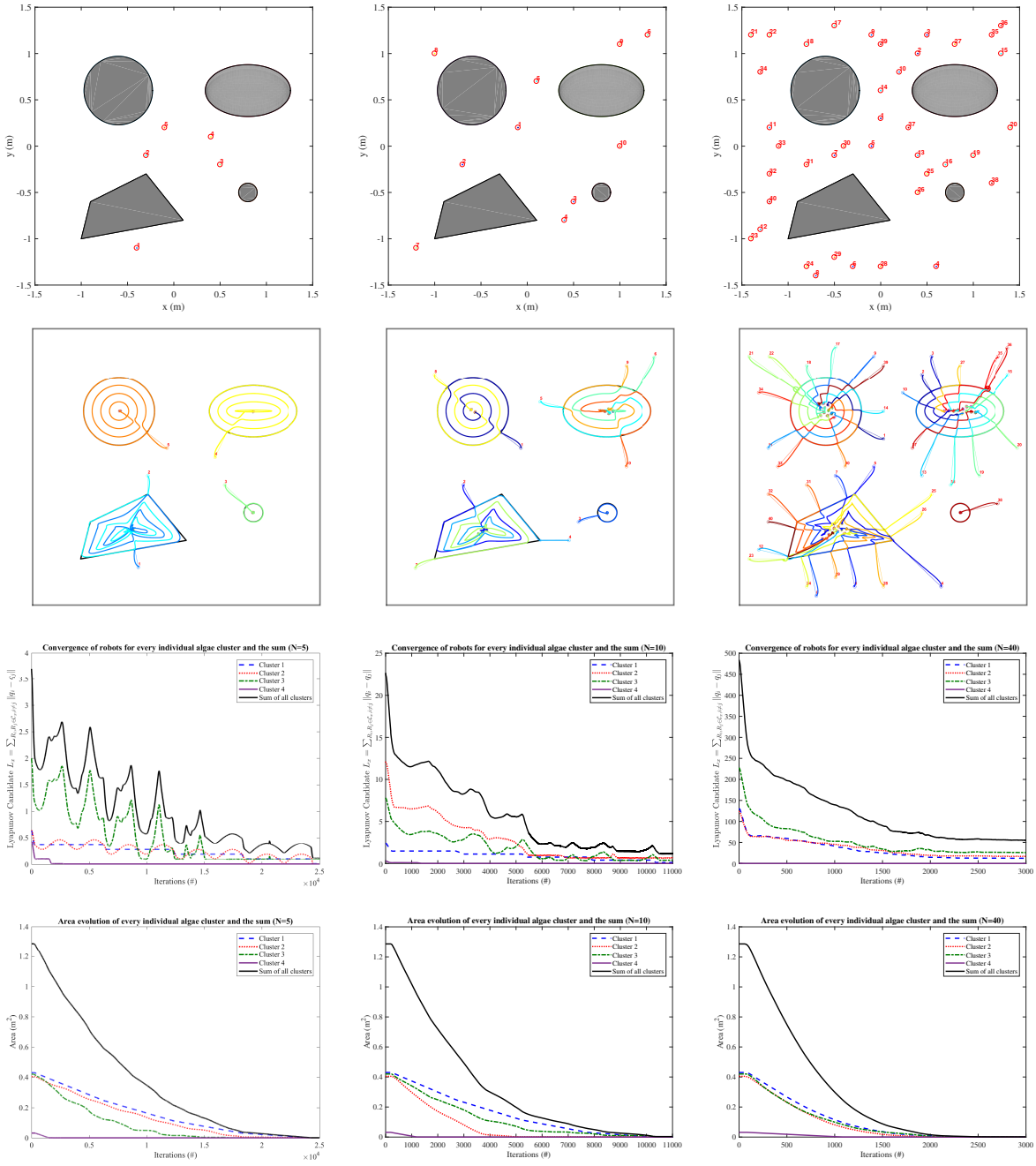
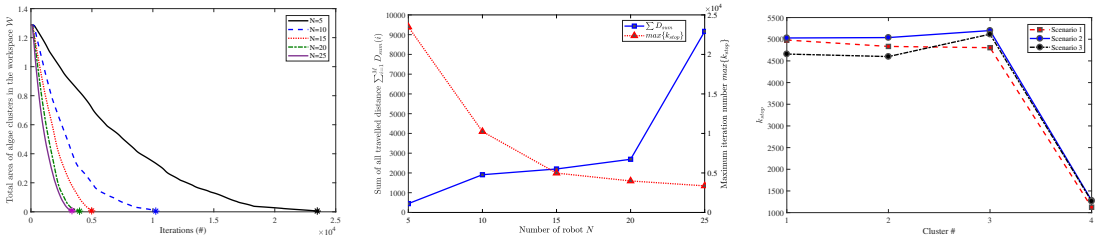


Figure 4.14. The first row shows the initial random distributions of robots, and the second row shows the coverage trajectories, with the number of robots being 5, 10, and 40. The convergence and area evolutions with respect to time are shown in the third and fourth rows, respectively.



(a) Evolution of areas with different N values.

(b) $\sum D_{sum}$ versus $\max\{k_{stop}\}$ for different N.

(c) The k_{stop} of four spills while $N = 15$.

Figure 4.15. (a) shows the change of residual spill areas in the workspace with respect to the iteration number, while the number of coverage robots differs. The star marker at the end of every plotted line indicates its corresponding $\max\{k_{stop}\}$; (b) shows the sum of traveled distance of all four spills and the maximum k_{stop} among the spills, while the number of robots $N = \{5, 10, 15, 20, \text{ and } 25\}$; (c) shows k_{stop} values of four individual spills while $N = 15$; three scenarios with different initial distributions were tested.

number $N = \{5, 10, 15, 20, \text{ and } 25\}$ with a random initial distribution. As shown in Figure 4.15(b), we can see that the optimal number of robots that fits the spill setting in Figure 3.7 is $N = 15$.

One might wonder how much the distribution of robots affects the k_{stop} of every individual spill coverage. Thus, three more scenarios with $N = 15$ but different initial distributions were tested. The results are depicted in Figure 4.15(c).

From Figure 4.15(c) we can see that k_{stop} value for each spill showed good invariance, thus facilitating a prediction of stop time when the distribution of robots differed. Meanwhile, the k_{stop} values of spill 1, 2, and 3 for each scenario were in the same time frame, which suggested that the coverage of these spills can be finished within almost the same time. Such uniformity in completion time further suggested that the workload was evenly distributed among deployed robots. Admittedly, the k_{stop} value for spill 4 seemed significantly lower than the others in Figure 4.15(c). It

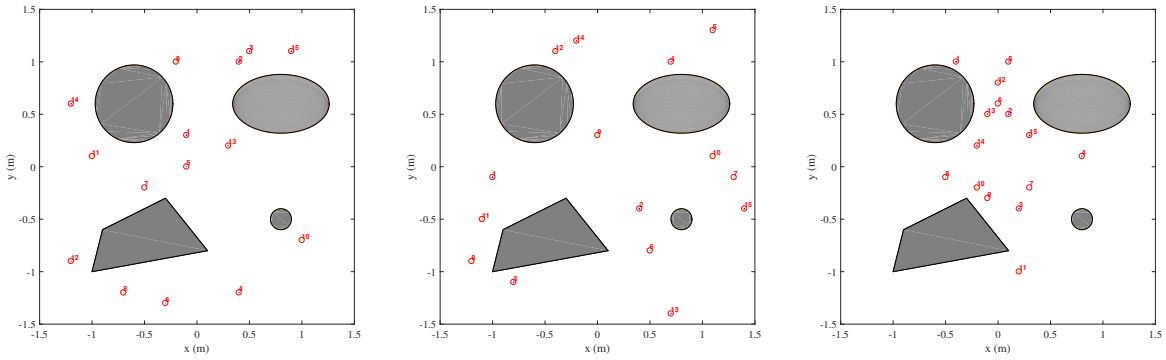


Figure 4.16. Three scenarios of having $N = 15$ robots deployed in coverage with different random initial positions.

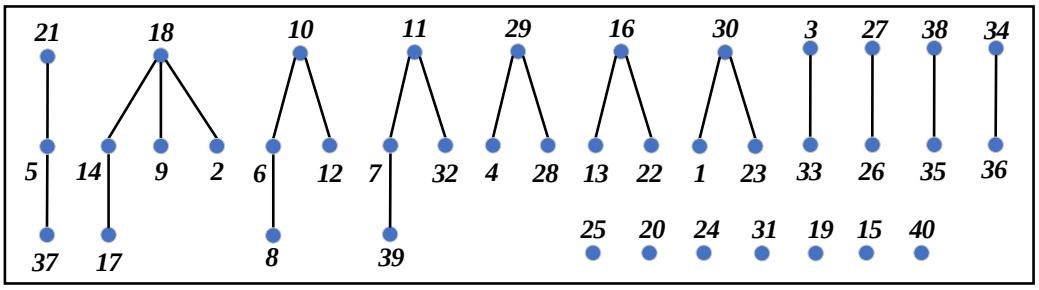


Figure 4.17. The shortest path trees that depict the hierarchical rendezvous, where the root node for each tree is a rendezvous point.

was because spill 4 had the highest robot density according to constraint (3.4), which necessitates that every spill has at least one robot. Thus, the experiment results conclusively confirm the efficacy of the proposed optimal allocation strategy in Section 3.2 and satisfy the two goals specified at the beginning of the same section.

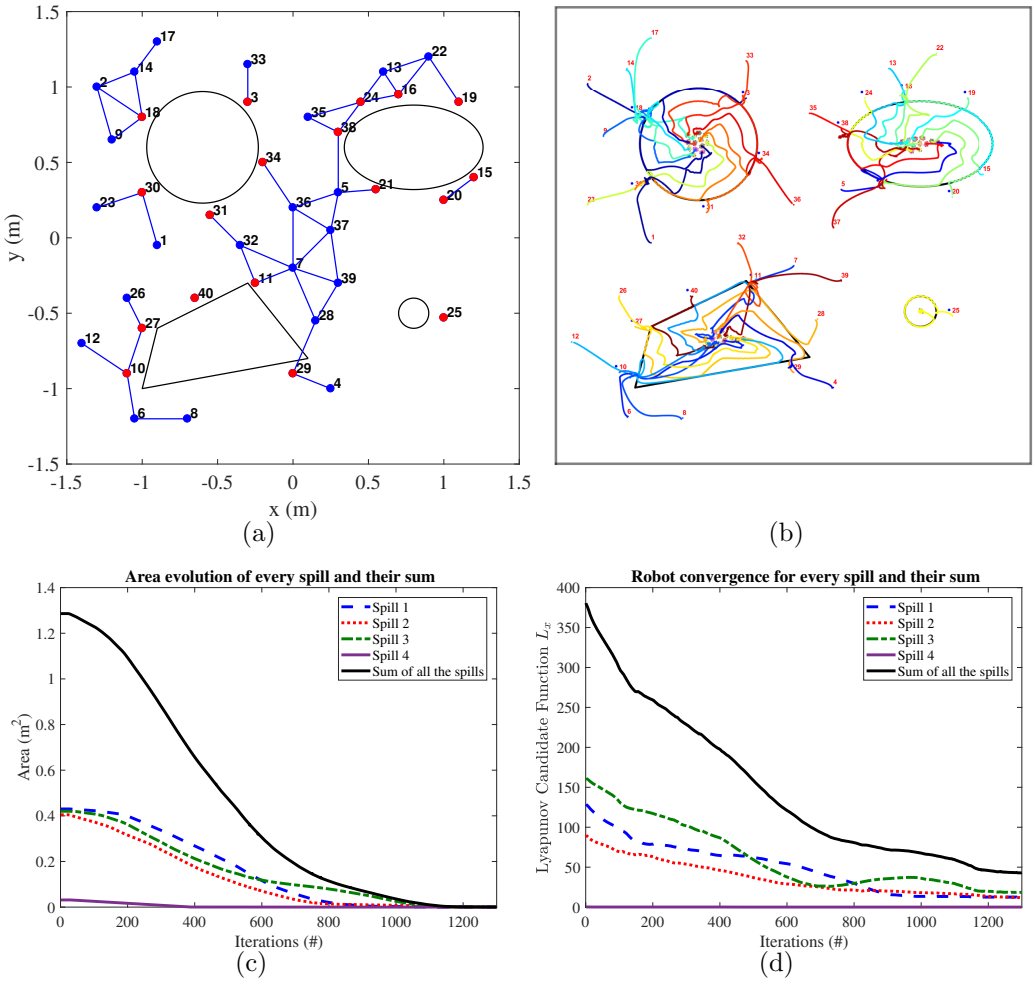


Figure 4.18. (a) The same initial distribution of robots as Case 1, with wireless connectivity represented as a graph in solid blue lines and rendezvous points D_m highlighted in red. (b) shows boundary shrink control trajectories with multi-point rendezvous. The area evolution and convergence of Case 2 are shown in (c) and (d), respectively.

Experiment Cases and Results With relatively small vision sensing range $r_{\mathcal{A}}$

With a small vision range but relatively wide wireless communication range, the robots that are far from the spill can maneuver to its vicinity and catch sight of the boundary with the help of hierarchical rendezvous based on wireless connectivity. The initial robot distribution is set randomly, which can be found in Figure 4.18(a). In this case, 18 shortest-path trees are constructed according to Section 5.7 with rendezvous points being $D_1(R_3), D_2(R_{18}), ...,$ and $D_{18}(R_{25})$, representing the robots that already detect the boundary at first. The rendezvous path trees are shown in Figure 4.17. The numerical results including robot association to these spills, rendezvous points D_m, and experiment data are shown in Figure 4.17.

4.2.6 Pivot-robot Based Coverage (PRBC)

To deal with a large-scale patch, we employ the divide-and-conquer method and partition the large patch into several small areas named *packing areas* with *geometric tessellation* (GT). This partition concept is demonstrated in Figure 4.19.

4.2.7 Problem Statement and Assumptions for PRBC

Assume the spill is static and its packing areas are available after the geometric packing process. We propose our research question to be: How can a team of robots collectively clean up a packing area with only local sensing but not a global coordinate frame or external positioning device? Specifically, the whole spill in the workspace $\mathcal{W} \in \mathbb{R}^2$ is denoted as $\mathcal{S}_{\mathcal{W}} \in \mathbb{R}^2$. The packing areas after the geometric packing are denoted as below,

$$\mathcal{P}_i = \{(x, y) : (x, y \in interior(PP_x)_i)\}, \forall i = 1, ..., \mathtt{M}, \tag{4.30}$$

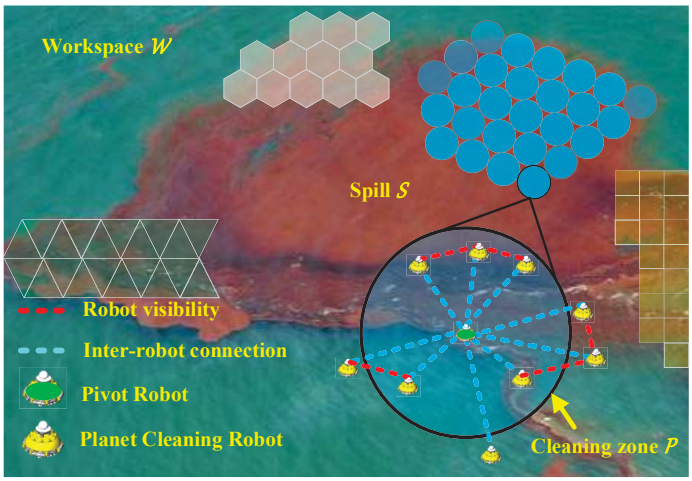


Figure 4.19. An image that illustrates our proposed collective coverage strategy for various geometric packing shapes, including circles, squares, triangles, and hexagons for irregular spill coverage. Blue dashline shows the connectivities between planet robots (yellow) and the pivot robot (green). Red dashline represents one robot being in the vision of another robot. (Photo credit to *Smithsonian.com*)

where PP_x stands for a specific *packing pattern* such as circle, triangle, square, or hexagon; $\mathtt{M}$ denotes the total number of packing areas for spill $\mathcal{S}_\mathcal{W}$. Since the spill remaining in the packing area $\mathcal{P}_i$ is changing because of ongoing cleanup, we define it as

$$\mathcal{S}_i = \{(x,y) : (x,y \in interior(PP_x)_i) \cap (x,y \in \mathcal{S}_\mathcal{W})\}, \forall i = 1,...,\mathtt{M}. \quad (4.31)$$

We use $\partial\mathcal{P}_i$ and $\partial\mathcal{S}_i$ to denote the boundary of $\mathcal{P}_i$ and $\mathcal{S}_i$, respectively. If $\partial\mathcal{S}_i$ does not exist or is not closed initially since $\mathcal{S}_i \nsubseteq \mathcal{P}_i$, the control algorithm should enable robots to create a closed boundary $\partial\mathcal{S}_i$ such that $\mathcal{S}_i \subseteq \mathcal{P}_i$. In this case, robots can track the spill boundary $\partial\mathcal{S}_i$ and realize complete coverage over $\mathcal{S}_i$.

We further state the following assumptions that our research is based on: The spill to be covered is overall integrated and has a smooth boundary; the spill

features colors or textures that can be distinguished from the base using image processing techniques.

The coverage operation starts when the packing process is completed. The targeted large-scale spill is partitioned into `M` number of small packing areas with specific shapes. Note that the centroid of the packing area should reside in $\mathcal{S}_i$. Meanwhile, the radius of the packing area cannot exceed the range of wireless connectivity, i.e., $r_{\mathcal{P}i} \leq r_{comm}$. Otherwise, the planet robots may not be able to identify the pivot robot and localize itself. We first deploy the pivot robot to the centroid of the first packing area $\mathcal{S}_1$. The pivot robot can be randomly selected, and it can move autonomously to the designated position under the operator's guidance. The planet robots start performing coverage after the pivot robot is ready. The connectivity can be built between the planet robot and the pivot robot by simply using a paired Wi-Fi access point (AP) for the pivot robot and an adapter combined with an antenna for the planet robot. We can localize the planet robot, including bearing and distance, through the Wi-Fi signal strength of the pivot robot (Fink & Kumar, 2010; Min et al., 2016).

4.2.8 Control Design for PRBC

Based on the coverage model shown in Section 4.2.2, we provide the coverage control algorithm that enables full coverage of the packing area to remove the spill and converges to the pivot when the operation reaches an end. Using $\mathbf{q}_i = (x_i, y_i)$ and $\mathbf{q}_{pv} = (x_{pv}, y_{pv})$ to denote the position of a planet robot R_i and the pivot robot R_{pv}, and $r_{\mathcal{P}j}$ to denote the radius of packing area $\mathcal{P}_j$, Algorithm 4 shows the procedure of the coverage control, which runs iteratively until the operation is completed.

To illustrate Algorithm 4, we present a finite state transition diagram consisting of *four* states in Figure 4.21. Meanwhile, we sketch two possible initial situations under circular packing with randomly distributed robots in Figure 4.20.

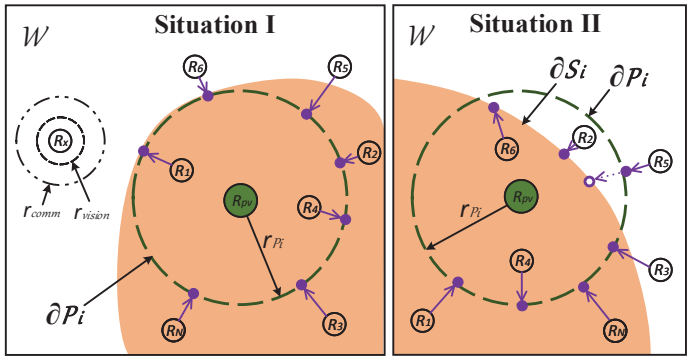


Figure 4.20. Two possible situations in coverage operation under circular packing. $\partial\mathcal{P}$ and $\partial\mathcal{S}$ represent the boundaries of the packing area and the spill, respectively. Pivot robot R_{pv} (green) is located at centroid of the packing area. r_{vision} and r_{comm} denote the vision sensor and wireless connection ranges for planet robots. The purple dots indicate the deployment goals for planet robots.

In the transition diagram, every robot starts with deployment (*State 1*), where it moves toward the goal position on $\partial\mathcal{P}_j$. Especially, if the robot experiences a traverse $\mathcal{S}_j \rightarrow \neg\mathcal{S}_j$, it stops at the traverse point on $\partial\mathcal{S}_j$ and terminates deployment. This case is demonstrated by R_6 in Situation II of Figure 4.20. *State 2* means if the robot is deployed but does not detect $\partial\mathcal{S}_j$ or reside in $\mathcal{S}_j$, it will start moving toward the pivot robot until it detects $\partial\mathcal{S}_j$. This state is demonstrated by R_5 in Situation II. Note that the robot can travel as fast as it can in *State 1 and 2*, since it is not performing the coverage. In *State 3*, if no $\partial\mathcal{S}_j$ is detected but the robot resides in the spill, it will maneuver along $\partial\mathcal{P}_j$ and create a boundary $\partial\mathcal{S}_j$ that trailing robots can follow in *State 4*. All the robots in Situation I and robots R_1, R_3, R_4, and R_N in Situation II demonstrate *State 3*, while R_2 and R_6 in Situation II demonstrate *State 4*. The coverage of this packing area ends up with all planet robots gathering around the pivot robot within a threshold ϵ according to the robot dimensions.

To motivate robots to track $\partial\mathcal{S}_i$ or $\partial\mathcal{P}_i$ in order to cover the spill collectively but avoid collision, we introduce APF-based motion controllers for each state. To

Algorithm 4: Collective Coverage Control Algorithm for every planet robot in packing area $\mathcal{P}_j$

repeat
  repeat
    R_i moves from $\mathbf{q}_i$ to $\mathbf{q}_{vt} + \frac{\mathbf{q}_i - \mathbf{q}_{pv}}{\|\mathbf{q}_i - \mathbf{q}_{pv}\|} r_{\mathcal{P}j}$
    if R_i *experiences a traverse* $\mathcal{S}_j \rightarrow \neg\mathcal{S}_j$ **then**
      R_i stops at $\partial\mathcal{S}_j$
  until R_i *under deployment /*(State 1)*/*
  repeat
    R_i moves from $\mathbf{q}_i$ to $\mathbf{q}_{vt} + \frac{\mathbf{q}_i - \mathbf{q}_{pv}}{\|\mathbf{q}_i - \mathbf{q}_{pv}\|} r_{\mathcal{P}j}$
    if *No* $\partial\mathcal{S}_j$ *is detected within* r_{vision} **then**
      if R_i *resides in* $\mathcal{S}_j$ **then**
        R_i covers the spill under (4.7) by tracking $\partial\mathcal{P}_j$ and hence creates $\partial\mathcal{S}_j$ */*(State 3)*/*
      else
        R_i moves toward $\mathbf{q}_{pv}$ */*(State 2)*/*
    else if $\partial\mathcal{S}_j$ *is detected within* r_{vision} **then**
      R_i covers the spill under (4.7) by tracking $\partial\mathcal{S}_j$ */*(State 4)*/*
      if $\|\boldsymbol{q}_i - \boldsymbol{q}_{pv}\| \geq r_{\mathcal{P}i}$ **then**
        R_i switches to track $\partial\mathcal{P}_j$ and continues the coverage operation */*(State 3)*/*
    else
      R_i fails in vision sensing; it stops and becomes an obstacle for other robots
    Bounding speed $\dot{q}_i = \frac{\mathcal{V}}{d}$
  until R_i *under coverage operation*
until $\|\boldsymbol{q}_i - \boldsymbol{q}_{pv}\| \leq \epsilon,\ \forall i \in \{1, ..., N\}$

avoid local minima and deadlock, we propose a pre-prioritized collision avoidance strategy to coordinate the robots working in different states. Meanwhile, all the robots performing coverage over the spill shall move counterclockwise (CCW) about the pivot robot when tracking $\partial\mathcal{S}_i$ or $\partial\mathcal{P}_i$. A uniform moving direction leads to a distributed coordinate control.

The proposed avoidance priority is indicated in Figure 4.21. The robots working in lower priority have to yield to those working in higher priority, and

robots performing the coverage have the highest priority. The avoidance priority for these states is determined as *State 3* = *State 4* > *State 1* = *State 2*. No transition exists between *State 2* and *State 3*, because the robot converges to the pivot only if nothing is detected. Furthermore, special consideration should be given to the robots working in the states of the same priority. As we stated above, robots working in *State 3* and *4* (i.e., tracking either $\partial\mathcal{S}_i$ or $\partial\mathcal{P}_i$) are moving uniformly CCW. Given a continuous and smooth spill boundary consisting of $\partial\mathcal{S}_i$ and $\partial\mathcal{P}_i$, the collision happens only between a leading robot and its trailing robot. However, the robot working in *State 3* does not have a leading or trailing robot. Hence, we will limit the possible collisions to be from the robots working in *State 4*.

Similarly, for the robots working in *State 1* and *2*, since they are moving either towards or against the pivot robot, no collision is foreseen if their trajectories are non-coincident. Such a pre-prioritized collision avoidance rule eliminates local minima that halt the robot, and instead relies only on local sensing. Deadlock, although not observed in the validation experiments, can be tackled with appropriate motion planning methods (Weerakoon, Ishii, & Nassiraei, 2015). In practice, robots can recognize each other's states with minimum communication or via light signals and near-field communication such as Radio Frequency Identification (RFID) (Makris, Michalos, & Chryssolouris, 2012).

Assuming a single integrator model for planet robot control, i.e.,

$$\dot{\mathbf{q}}_i(t) = \mathbf{u}_i(t),\ i \in 1, ..., N \tag{4.32}$$

where $\mathbf{u}_i(t) \in \mathbb{R}^2$ denotes the control input for planet robot R_i at time instant t, the detailed motion control law for every state is provided in the following subsections.

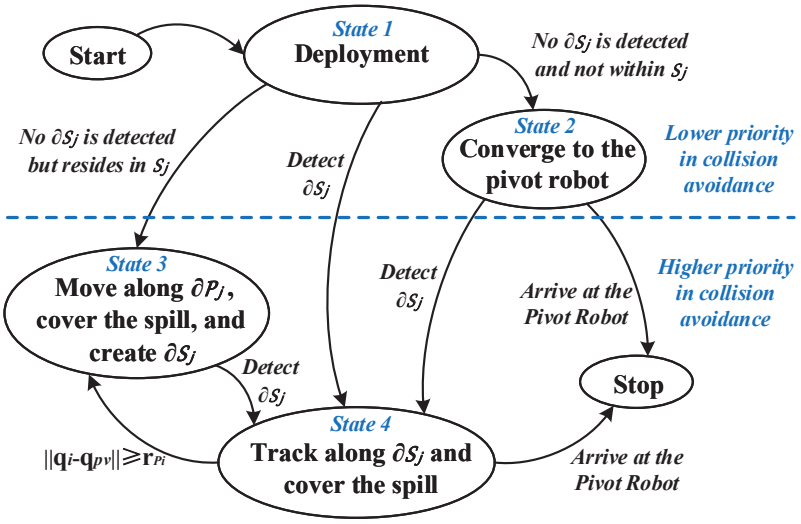


Figure 4.21. A finite-state machine diagram showing the hybrid robot coordination strategy. The robots perform collective coverage over the spill when it is moving along either $\partial\mathcal{S}_j$ or $\partial\mathcal{P}_j$, i.e., *State 3* and *State 4*, which are of higher priority than the others in terms of collision avoidance.

Motion Controller for State 1

For robots working in *State 1*, provided the goal positions for deployment $\mathbf{q}_g = \mathbf{q}_{vt} + \frac{\mathbf{q}_i - \mathbf{q}_{pv}}{\|\mathbf{q}_i - \mathbf{q}_{pv}\|} r_{\mathcal{P}j}$, we construct an attractive potential field formulated as

$$U_d^{(1)}(\mathbf{q}) = \frac{1}{2}\xi_1 \|d(\mathbf{q}, \mathbf{q}_g)\|_2^2 \tag{4.33}$$

where ξ_1 is a scaling parameter and $\|d(\mathbf{q}, \mathbf{q}_g)\|_2$ is the distance between the current positions of robots and their goals obtained from radio signal strength measurement (Min et al., 2016).

Additionally, we propose a repulsive potential exerted on the robots to avoid collision. This repulsive potential field is formulated as

$$U_i^{(1)}(\mathbf{q}) = \begin{cases} \frac{\xi_2}{2}\left(\frac{1}{\|d(\mathbf{q},q_i)\|_2} - \frac{1}{d_0}\right)^2, & \text{if } \|d(\mathbf{q},q_i)\|_2 \leq d_0, \\ 0, & \text{otherwise}, \end{cases} \tag{4.34}$$

where ξ_2 is a positive scaling factor and $\|d(\mathbf{q},q_i)\|_2$ denotes the distance between robot R_i that works in the states of higher or equal avoidance priority to other robots q_i that are within the effective range, which values are determined by the robot dimensions.

Due to this, for any planet robot, the final constructed potential field for motion control is

$$U_f^{(1)}(\mathbf{q}) = U_d^{(1)}(\mathbf{q}) + \sum_{i \in \mathcal{N}_{r_{vision}}^{(3,4)}} U_i^{(1)}(\mathbf{q}) \tag{4.35}$$

where $\mathcal{N}_{r_{vision}}^{(3,4)}$ represents robots working under *State 3 and 4* and within the vision sensing range r_{vision}.

With (4.35), the control input for the robot working in *State 1* is obtained:

$$\begin{aligned} \mathbf{u}^{(1)} &= -\nabla U_f^{(1)}(\mathbf{q}) \\ &= \begin{cases} \xi_1(\mathbf{q}_g - \mathbf{q}) + \\ \qquad \sum_{i \in \mathcal{N}_{r_{vision}}^{(3,4)}} \left(\frac{1}{d_0} - \frac{1}{\|d(\mathbf{q},q_i)\|_2}\right) \frac{\xi_2 \nabla d(\mathbf{q},q_i)}{\|d(\mathbf{q},q_i)\|_2^2}, \\ \qquad\qquad\qquad\qquad \text{if } \|d(\mathbf{q},q_i)\|_2 \leq d_0, \\ \xi_1(\mathbf{q}_g - \mathbf{q}), \qquad\qquad\qquad\qquad \text{otherwise}, \end{cases} \end{aligned} \tag{4.36}$$

where $\mathbf{u} = [u_1, u_2, ..., u_i]$ is the input velocity of all the robots under deployment. Moreover, the input velocity $\|u_i\|$ is bounded by a maximum value according to Algorithm 4.

Motion Controller for State 2

Similar to *State 1*, the source of attractive potential becomes the pivot robot R_{pv}, thus, we have the attractive potential field formulated as

$$U_d^{(2)}(\mathbf{q}) = \frac{1}{2}\xi_3 \|d(\mathbf{q}, \mathbf{q}_{pv})\|_2^2 \tag{4.37}$$

where ξ_3 is a scaling parameter and $\|d(\mathbf{q}, \mathbf{q}_{pv})\|_2$ the distance between robot current positions and the pivot robot.

According to the pre-decided priority for collision avoidance, the repulsive potential field for robots working in *State 2* is constructed the same as *State 1*, which results in a control law shown in (4.39):

$$U_f^{(2)}(\mathbf{q}) = U_d^{(2)}(\mathbf{q}) + \sum_{i \in \mathcal{N}_{rvision}^{(3,4)}} U_i^{(2)}(\mathbf{q}) \tag{4.38}$$

$$\begin{aligned} \mathbf{u}^{(2)} &= -\nabla U_f^{(2)}(\mathbf{q}) \\ &= \begin{cases} \xi_3(\mathbf{q}_{pv} - \mathbf{q}) + \\ \qquad \sum_{i \in \mathcal{N}_{rvision}^{(3,4)}} \left(\frac{1}{d_0} - \frac{1}{\|d(\mathbf{q},q_i)\|_2}\right) \frac{\xi_2 \nabla d(\mathbf{q},q_i)}{\|d(\mathbf{q},q_i)\|_2^2}, \\ \qquad\qquad\qquad\qquad \text{if } \|d(\mathbf{q}, q_i)\|_2 \leq d_0, \\ \xi_3(\mathbf{q}_{pv} - \mathbf{q}), \qquad\qquad\qquad \text{otherwise.} \end{cases} \end{aligned} \tag{4.39}$$

Motion Controller for State 3

For the robot moving along $\partial\mathcal{P}$ in *State 3*, since it has no leading or trailing robot in the same state, it requires only attractive potential to motivate movement. In order to avoid a discrete control method, which typically results in frequent position updates, and improve control efficiency, we propose a continuous control law (4.40) and enable robots moving along $\partial\mathcal{P}$.

We use attractive potential (4.37) to motivate the movement of the robot. However, different from *State 2*, the robot in *State 3* has to move in a tangent way along $\partial\mathcal{P}$ and CCW about the pivot. Thus, we determine the control law as below:

$$\begin{aligned} \mathbf{u}^{(3)} &= -T \cdot \nabla U_d^{(1)}(\mathbf{q}) = T \cdot \xi_3(\mathbf{q}_{pv} - \mathbf{q}), \\ T &= \begin{bmatrix} \cos(-\pi/2) & -\sin(-\pi/2) \\ \sin(-\pi/2) & \cos(-\pi/2) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \end{aligned} \tag{4.40}$$

Particularly, it is difficult to keep the robot in the orbit and tracking $\partial\mathcal{P}$ with only attractive potential, as a disturbance may yield $\|\mathbf{q}_{pv} - \mathbf{q}\| > r_{comm}$ and break the robot connection to the pivot robot. Due to this, we can introduce an asymmetric potential function such as (26) in (Zavlanos, Egerstedt, & Pappas, 2011) and let $\rho_2 = r_{comm}$ and $\rho_0 = r_{\mathcal{P}j}$ for a stabilized motion.

Motion Controller for State 4

Since collision happens only between a leading robot and its trailing robot if they are working in *State 4*, we construct an attractive potential $U_d^{(4)}$ exerted on robot R_i by its leading robot R_{i+1}. When R_i is tracking $\partial\mathcal{S}$, the measurement of the length along the spill boundary between itself and other agents within its vision sensing range is used to construct the potential field. Practically, such measurement can be performed with stereo vision sensors, LIDAR, or high-resolution laser rangefinders, along with the techniques developed in (Gowal et al., 2011; Mitiche & Aggarwal, 2014). Suppose there is a function $s = f(\mathbf{q})$ to represent the spill boundary $\partial\mathcal{S}$, where $s \in [0, \|\partial\mathcal{S}\|]$ indicates the length between a reference point and an agent of position $\mathbf{q}$ in CCW. We can show that once a robot R_{i+1} falls

within the vision range of its trailing robot R_i, the distance between the two neighboring robots, namely $l_i = \|q_i, q_{i+1}\|_{\partial\mathcal{S}}$, can be decided by

$$l_i = \begin{cases} s_{i+1} - s_i, & \text{if } s_{i+1} \geq s_i, \\ s_{i+1} - s_i + \|\partial\mathcal{S}\|, & \text{if } s_{i+1} < s_i. \end{cases} \tag{4.41}$$

If the leading robot of R_i is beyond its vision range r_{vision}, we define a virtual distance l_i^* and update (4.41) as below:

$$l_i^* = \begin{cases} l_i, & \text{if } l_i \leq r_{vision}, \\ r_{vision}, & \text{if } l_i > r_{vision} \text{ or } l_i \text{ is unknown.} \end{cases} \tag{4.42}$$

The attractive potential function is then defined as

$$U_d^{(4)} = \frac{1}{2}\xi_4 l_i^{*2} \tag{4.43}$$

where ξ_4 is a positive scaling factor. The linear velocity input u_i should be in the direction of the negative gradient of $U_d^{(4)}$ with respect to s_i such that

$$u_i^{(4)} = -\nabla_{s_i} U_d^{(4)} = \xi_4 l_i^*. \tag{4.44}$$

It can be easily proved that the attractive potential $U_d^{(4)}$ can motivate robot R_i to move forward and also prevent collision with its leading robot, since it is always non-negative and $u_i \to 0 \; iff \; l_i^* \to 0$. Furthermore, all the planet robots will gather and stop around the pivot robot when the coverage control is completed. Here, one can see that the motion controller for State 4 is similar to the one proposed for ABSC above.

4.2.9 Coverage to the Next Packing Area in PRBC

When a group of robots completes the coverage to the current packing area $\mathcal{P}_i$ and removes the spill $\mathcal{S}_i$, the pivot robot, after relocation by human operator teleoperation or navigation, can herd all the planet robots into the next packing area $\mathcal{P}_{i+1}$ to perform a new round of coverage operation. One of the practical herding strategies can be found in (Parasuraman et al., 2018), which features robot connectivity preservation.

4.3 Simulation Experiments for PRBC

We still use Robotarium and MATLAB to conduct simulation experiments. In the experiments, R_1 denotes the pivot robot and is located at the center of the arena. The linear velocity of the robot is bounded by $v_{max} = 0.075\,\text{m/iter}$. The average width d in Figure 4.8 is set to be 0.33 m.

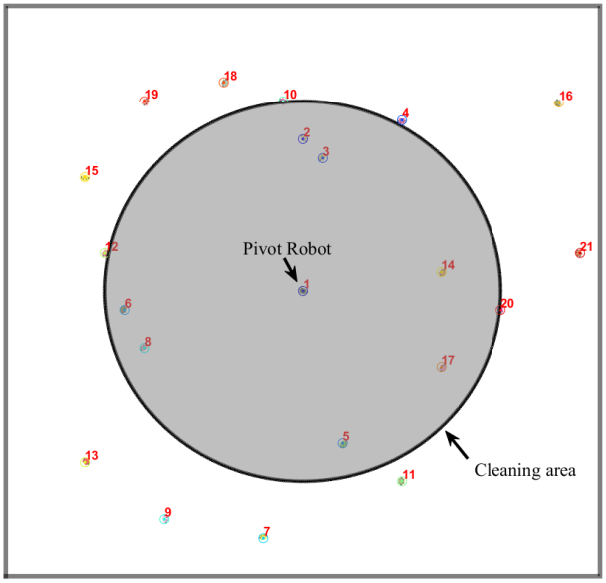


Figure 4.22. Workspace of simulation experiment with disk shape cleaning zone with a radius of 1 m. The pivot robot is at the center of the cleaning zone, while the planet robots are randomly distributed.

4.3.1 Evaluation Metrics

The following metrics are decided for our experiments to evaluate the performance of the proposed strategy:

1. Lyapunov candidate function (Convergence):

$$L = \sum_{R_i, R_j \in \mathcal{P}, i \neq j} \|q_i - q_j\|, \textbf{ or } L = \sum_{R_i \in \mathcal{P}} \|q_i - q_p\|. \tag{4.45}$$

   The second function is used if only one planet robot is involved, where q_p is the location of the pivot robot.

2. The number of iterations ($k_{stop} \leq k_{max}$) to reach the following stop condition:

$$\text{Stop at } k_{stop} \text{ if the current area } A(t) \leq A_{min}. \tag{4.46}$$

   Here A_{min} is defined to be 1% of the initial area.

4.3.2 Simulation Experiment Scenarios

The following four scenarios are designed to demonstrate the efficacy and efficiency of the proposed solution.

- *Scenario 1* - Adaptiveness to various packing shapes
- *Scenario 2* - Scalability which allows for multiple robots
- *Scenario 3* - Fault tolerance to the robot with coverage failures
- *Scenario 4* - Unknown obstacle during the operation
- *Scenario 5* - Sequential coverage to multiple packing areas

4.3.3 Simulation Results

In this section, we present the simulation results along with analyses corresponding to the proposed scenarios in the previous section.

Scenario 1 - Adaptiveness to Various Packing Shapes

To demonstrate the adaptiveness to different packing shapes with our proposed collective coverage control strategy, we select four representative geometric shapes including triangle, square, hexagon, and circle. In contrast, (Song, Gupta, Hare, & Zhou, 2013) and (Jin & Ray, 2014a) can deal with only square cell coverage.

Of all four shapes, the circular packing method can maximize the wireless communication range, i.e., can have possibly a greater packing area with less travel of robots than other shapes. However, as its main disadvantage, circular packing allows 90.69% coverage with unique radius circles and hexagonal lattice arrangement (Chang & Wang, 2010). To improve the coverage rate, one can use circles with different radii and increase the packing densities to be $> 91\%$ (Heppes, 2003). Triangular, square, and hexagonal packing methods can achieve 100% coverage; however, the packing shapes have to be aligned adequately to avoid overlaps or gaps. In practice, the alignment issue can be circumvented by assuming a uniform orientation for all packing areas using magnetic compasses. Note that $r_{\mathcal{P}j}$ in Algorithm 4 is not a constant for non-circular packing methods, hence (4.40) may not apply. Nevertheless, $r_{\mathcal{P}j}$ can be obtained by referencing the attitude of the pivot robot and using estimation methods such as (Narkhede, Joseph Raj, Kumar, Karar, & Poddar, 2018).

The coverage performance and robot trajectories of the four packing shapes are shown in Figure 4.23. One can see that complete coverage over these areas is achieved with an eight-robot team working under our proposed control strategy. Thus, any large-scale spill that can be partitioned into several packing areas can be

covered in the same manner. Particularly, we present area evolution snapshots for the circular packing coverage in Figure 4.24, which demonstrate the evolution of robot trajectories and depict the remaining area with shadow.

Figure 4.25(a) and 4.25(b) show the coverage performance under Situation II of Figure 4.20, where $\partial\mathcal{S}$ already exists at the beginning of the operation. In this case, robots R_5 and R_6 detected $\partial\mathcal{S}$ when continuously heading toward the pivot robot, then entered *State 4* and started following $\partial\mathcal{S}$, rather than moving along $\partial\mathcal{P}$ as the other robots did.

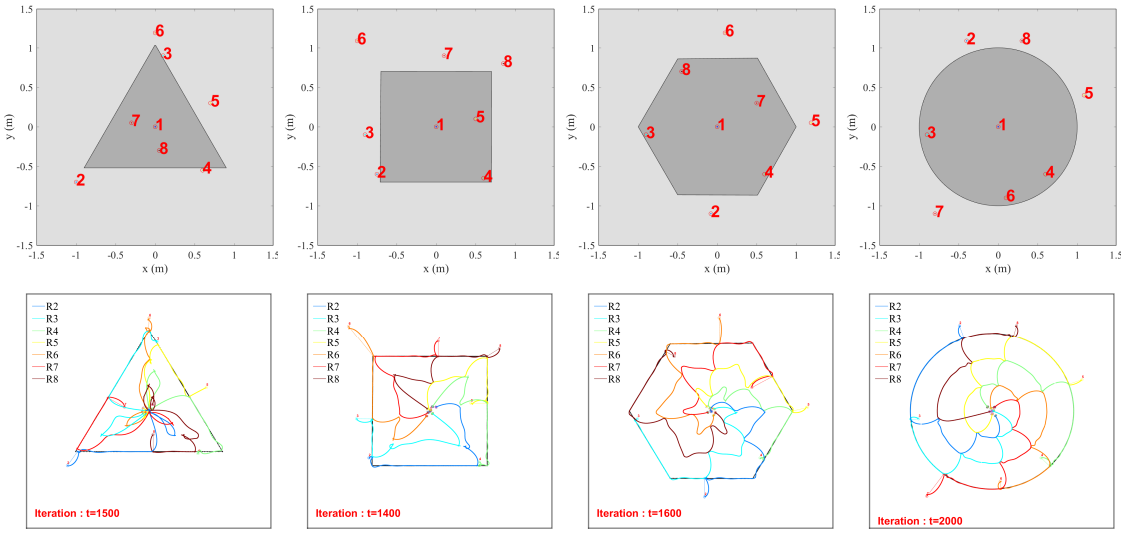


Figure 4.23. The figure shows a eight-robot team performing the proposed pivot-base collective coverage over the packing areas of different shapes including triangle, square, hexagon, and circle. This robot team consists of a pivot robot, denoted as "1" and located at the centroid of the area, and seven planet robots. The planet robots are randomly distributed before the task.

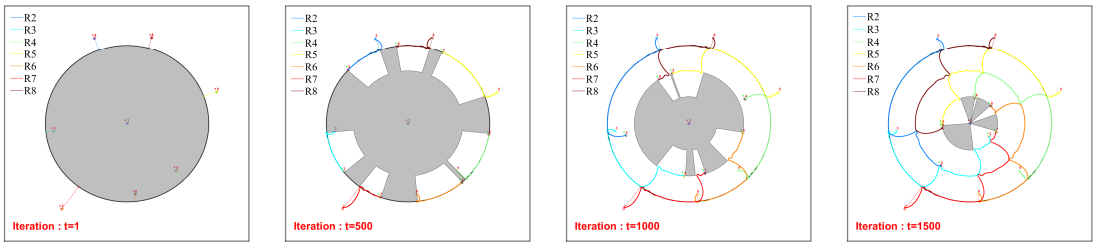


Figure 4.24. The snapshots show the evolution of the robot trajectories and remaining area in shadow, with a circular packing method.

Scenario 2 - Scalability That Allows for Multiple Robots

To improve the efficiency of coverage, we further demonstrate the scalability of our proposed solution that allows any number of planet robots. We take circular packing as an example; one can easily apply the same control scheme to other packing methods.

We start from the minimal number of robots $N = 2$, with one of them serving as pivot robot while the others being planet robots. We scale up N with the same interval of $five$ and therefore obtain four scenarios with $N = \{2, 6, 11, \text{ and } 16\}$. The initial distributions and trajectories of the robots during operation are shown in Figure 4.26.

The area evolution during the cleaning process is shown in Figure 4.27(a), from which we conclude the improvement in the efficiency of our proposed solution. The area of the cleaning zone decreased steadily as time elapsed. The k_{stop} was reduced significantly while having more robots deployed in the cleaning operation. Nevertheless, more robots employed does not always lead to a shorter operation time. Time may be consumed by robots avoiding each other in a crowd. The results in Figure 4.26 also validate the effectiveness of our robot motion controller and collision avoidance strategy by showing a smooth trajectory for every robot. The k_{stop} values for all the scenarios are listed in Table 4.3.

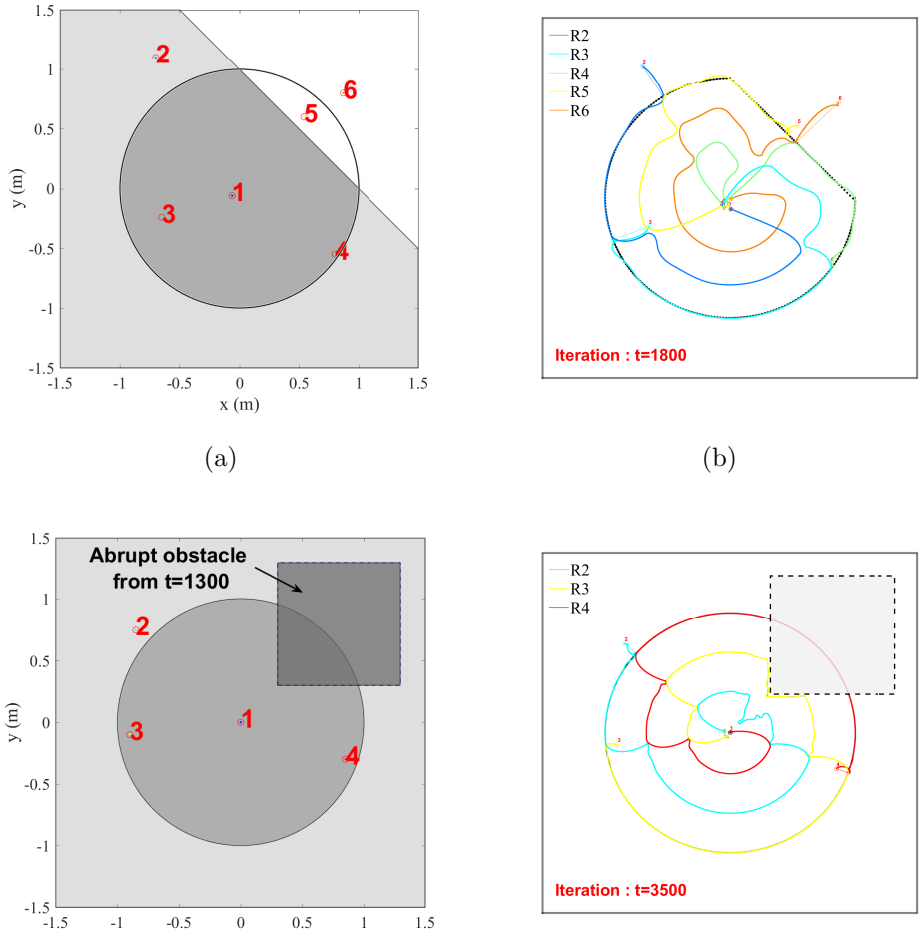


Figure 4.25. (a) Initial distribution of robots demonstrating Situation II of Figure 4.20, where $\partial\mathcal{S}$ already exists before operation. (b) The trajectories of robots. (c) An abrupt obstacle was introduced during the operation in time $t = 1300_+$. (d) Trajectories of the robots that succeeded in avoiding the obstacle and performed complete coverage.

At the end of the operation, all the robots gathered around the pivot robot as indicated in Figure 4.26. This characteristic potentially allows further deployment, such as moving to the next packing area or docking for recharging.

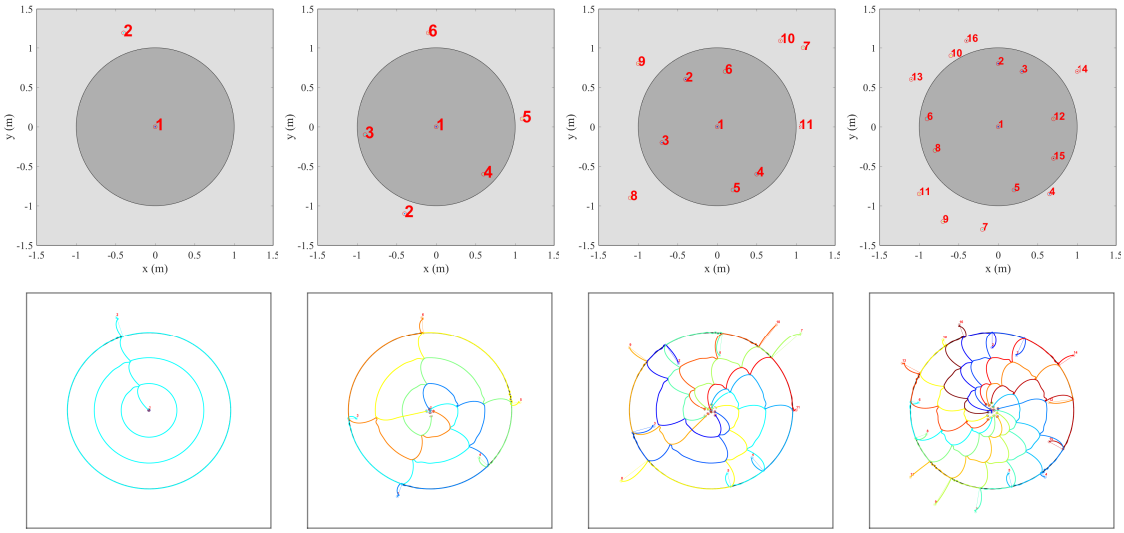


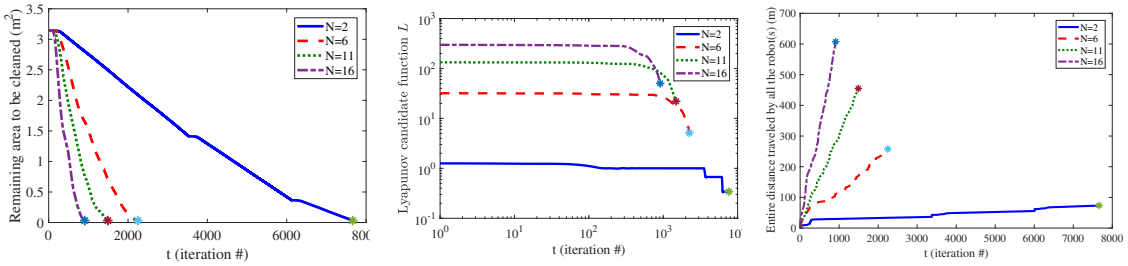
Figure 4.26. The first row shows the initial random distributions of robots; the darker disk is circular packing area in the workspace. The pivot robot is 1 and located at and located at the centroid of the area. The second row shows the robot trajectories, with the amount of robots being 2, 6, 11, and 16, respectively.

Table 4.3.

k_{stop} value and the total distance traveled by the planet robots until reaching k_{stop} for $N = 2, 6, 11,$ and 16.

Scenarios	$N = 2$	$N = 6$	$N = 11$	$N = 16$
k_{stop} value	7662	2247	1491	910
Distance traveled (m)	73.33	257.78	454.94	607.53

This convergence property is validated through the Lyapunov candidate function (4.45) and is demonstrated in Figure 4.27(b). In particular, convergence in scenario $N = 2$ was step-like during operation, because the second Lyapunov candidate function in (4.45) was used for evaluation. While the planet robot was covering $\mathcal{P}$, it approached the pivot in an intermittent way. Additionally, due to physical restraint of robots, the Lyapunov candidate function value did not converge to *zero* when the task was completed.



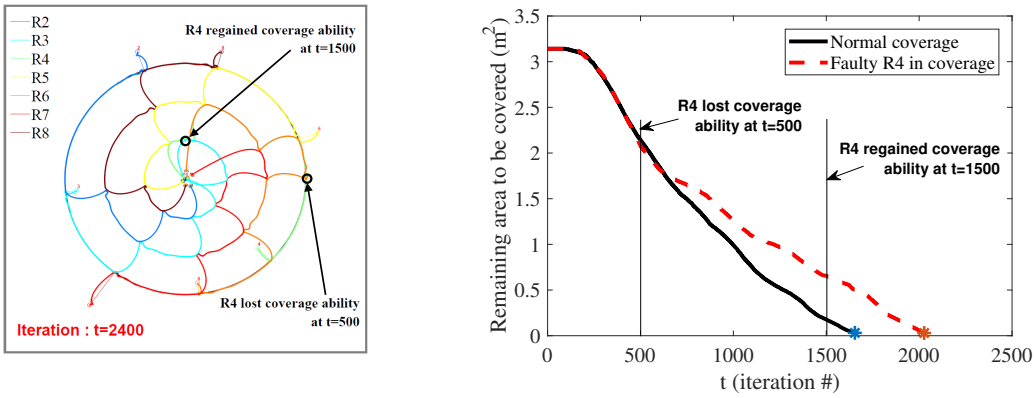
(a) (b) (c)

Figure 4.27. (a) The area evolution of the packing area as time elapsed, starting from the initial value of 3.14. The star sign at the end of every line indicates the k_{stop} value of the corresponding scenario. (b) The convergence of the robot team in logarithmic while the task is performed. The star sign at the end of every line indicates k_{stop} of the corresponding scenario. (c) The distance traveled by all the working robots while performing the cleaning task.

We also consider the energy consumption during the collective cleaning task, which can be reflected by the entire traveling distance of robots employed. The entire traveling distances with different N are indicated in Table 4.3. Notably, at the expense of a shorter operation time, the total traveled distance increased when more robots were deployed. The evolution of traveled distance is shown in Figure 4.27(c).

Scenario 3 - Fault Tolerance to the Robot with Coverage Failures

We demonstrate the fault tolerance capability of our proposed solution when the robot has coverage failure. Once employed in operation, a robot may face many issues that prevent it from removing the spill, e.g., a chemical substance removal robot facing filtering system failures, or an algae harvesting robot being fully loaded with algae. Even if the faulty robots can be simply isolated from the operation, we



(a) (b)

Figure 4.28. (a) R_4 encountered a coverage failure at $t = 500$ when moving along $\partial\mathcal{S}$ but recovered at $t = 1500$. It still converged to the pivot robot at the end. (b) The area evolution when R_4 was with (red) and without (black) a failure.

still want to let them converge to the pivot robot and collect them at the end of the operation, if they still have mobility.

In our experiment, the robots have the same circular packing setting demonstrated in Figure 4.23. However, R_4 lost its coverage ability at $t = 500$ but regained it at $t = 1500$. Figure 4.28(a) shows R_4 kept moving along $\partial\mathcal{S}$ and converged to the pivot robot when it encountered coverage failure. Due to this, the spill area evolution shown in Figure 4.28(b) as the red dashed line indicates a falling in the spill removal rate at $t = 500$, but it witnesses an increase at $t = 1500$ because R_4 regained the coverage ability.

Scenario 4 - Unknown Obstacle During the Operation

Unavoidably, the robots may encounter obstacles during the coverage operation in an unknown environment, such as rocks in water areas and robots that

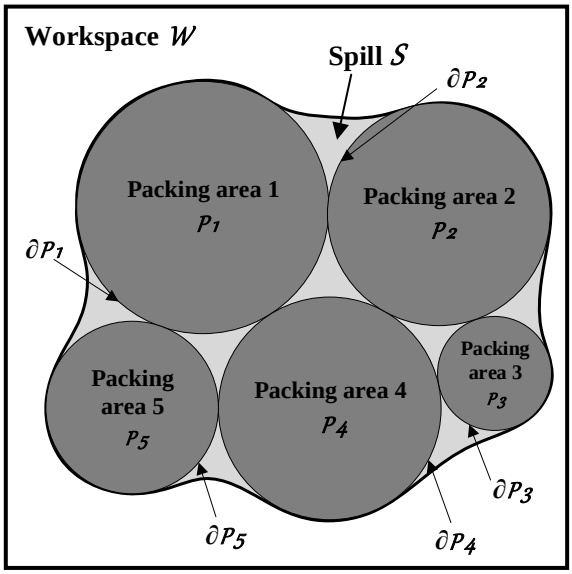


Figure 4.29. A large patch after partition with circular packing, resulting in five packing areas.

lose mobility. We demonstrate the capability of our strategy in abrupt obstacle avoidance. As shown in Figure 4.25(c), an abrupt obstacle was introduced in time $t = 1300_+$; however, the robot team succeeded in avoiding this obstacle using approaching sensors such as sonars and LIDARs and achieved complete coverage over the area still, as shown in Figure 4.25(d).

Scenario 5 - Sequential Coverage to Multiple Areas

We demonstrate a large spill coverage scenario using *four* robots after partition with circular packing shown in Figure 4.29 with the sequential packing method described in Section 4.2.9. The large spill was removed in sequence by covering every individual packing area, which process is partially depicted with snapshots in Figure 4.30.

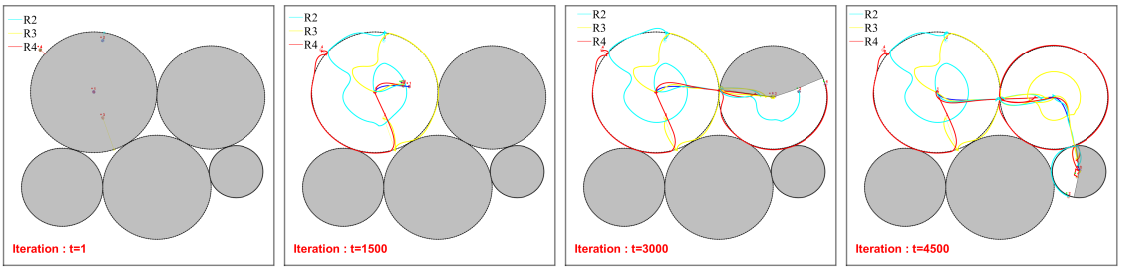


Figure 4.30. The coverage evolution of a large-scale spill partially shown with snapshots using *four* robots after circular packing partition.

4.4 Conclusion

In this chapter, we proposed concrete strategies for multi-robot coverage control, which can facilitate spill removal or space exploration. We considered two types of control, namely centralized path planning-based coverage control and distributed coverage control, depending on whether the global information of the workspace can be accessed by all the robots or not. Specifically, we presented an optimal path planned for a robot to cover a bounded spill, taking into account the spill absorption speed of the robot. To confront spills in a distributed way, we devised an ABSC method similar to cyclic pursuit. In terms of large-scale spills, we applied the divide-and-conquer approach and designed PRBC. These methods are formulated mathematically and validated through intensive simulations.

Acknowledgment

Section 4.1 is published in (Luo, Singh, et al., 2019). In Section 4.2, the subsections including Section 4.2.3, Section 4.2.4, and Section 4.2.5 are included in (Luo et al., (Under review)), while all the other subsections are published in (Luo et al., 2018).

CHAPTER 5. MULTI-ROBOT RENDEZVOUS PROBLEM

In this chapter, we present our work on the multi-robot rendezvous problem. We start with our problem statement, then introduce our methods and validate our methods with both simulation experiments and field tests.

5.1 Rendezvous Problem Setting

We are interested in how to enable rendezvous of distributed mobile robots at any designated leader robot node without losing network connectivity. However, in practice, we usually need to deal with an environment without global coordinates, such as an indoor environment which is GPS-denied. That requires the rendezvous controller to be coordinate-free. Moreover, most of the robots are non-holonomic, thus a unicycle dynamic has to be employed. In our solution, a bearing-based algorithm for non-holonomic vehicles using hierarchical tracking of wireless network topology is proposed.

5.1.1 Graph Theory

Following graph theory (Douglas, 2001), $G = (V, E)$ denotes an undirected weighted graph with node set V and edge set E. We further define other concepts that are used in representing the robotic network.

Every edge in E has its weight $w(e) :\rightarrow Z^+$. A *path* is a sequence of edges, normally finite, in a graph such that every sequential vertex belonging to the edge is connected. By most definitions, those connected vertices are mutually distinct. A graph G is *connected* if there exists a path (sequence of connected edges) between every pair of distinct nodes. The *shortest-path distance* between two vertices in a

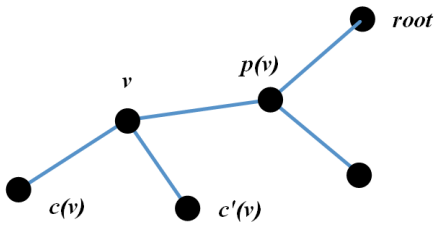


Figure 5.1. Illustration of a connected tree graph.

graph is the sum of weights for all edges in the shortest path connecting them. Let $d(v_i, v_j)$ denote the shortest-path distance between two vertices v_i and v_j. A tree is a connected graph without cycles. A *rooted tree* has one node, which is set as the *root*. Each node in a rooted tree has a parent-child relationship with its neighboring nodes. A *cycle* is a graph that consists of a certain number of vertices connected to form a closed chain. A connected graph without any cycle is a *tree graph*. A tree graph is called a *rooted tree* if one vertex has been designated the *root*. In a rooted tree, $p(v)$, the *parent* of a node v, is the node adjacent to v on the path to the root. $c(v)$, *child* of a node v, is a node of which v is the parent. A *leaf* in a tree graph is a node having no child. Figure 5.1 illustrates a connected tree graph, in which both $c(v)$ and $c'(v)$ are leaves of node v.

The *subgraph* of G induced by a node set $S \subset V$ is the graph (S, E_S) in which $E_S = \{\{x, y\} \in E : x, y \in S\}$. A *spanning tree* of a connected graph G is a subgraph containing all nodes, but containing no cycles. A single connected graph can yield many spanning trees with different forms. In a tree graph, a path between any two nodes is unique. Given a connected, edge-weighted graph G, a *shortest-path tree* $T \subset G$ is a tree, such that a path in T, between any node u and the root, is the shortest path in G between the two nodes.

5.1.2 Assumptions and Definitions

We introduce assumptions and definitions that drive our control laws. Let $\mathbf{q}_i \in R^2$ denote the position of a robot i. The dynamics of robot i are given by $\dot{\mathbf{q}}_i = v$. This single-integrator agent model is commonly used in the multi-robot system literature (Dimarogonas & Johansson, 2008; Sabattini, Secchi, Chopra, & Gasparri, 2013).

We assume that each robot is equipped with a wireless (ad hoc) network device and a bearing sensor to measure the relative bearing of neighboring robots. The robots sense neighboring robots and share this information across the network, which we use to build and dynamically update the network topology. The proximity information from local sensors (e.g., sonar, radar, and LIDAR), if available, can be used to assign weights to the edges in the network topology. If such range sensors are not available, then each edge will have an equal weight in the network graph. Note, we assume that the local proximity sensors and bearing sensors are not faulty.

We say that a robot *senses* another robot in the case where the distance between these two robots is shorter than R_m and a line of sight (LOS) is established between the two robots. We also claim that a robot *encounters* another robot in the case where the relative distance between them is shorter than $\epsilon \approx 0 \ll R_m$. The two robots are *neighbors* in the case where they can sense each other while satisfying that the relative distance between them is shorter than $R_m - \epsilon$.

In the case where a robot is equipped with range sensors, such as radar or LIDAR, these can be used by the robot to sense its neighbors. In the case where two robots sense each other, the line segment connecting the two robots must not intersect an obstacle. This way, as one robot moves towards its neighboring robot, it does not collide with any obstacle.

In our experiments, a robot is not equipped with range sensors. Rather, a robot uses a communication module to sense its neighboring robot. In the case where LOS between one robot A and another robot B is blocked, then signal

strength from A to B degrades significantly. Also, as the relative distance between A and B increases, signal strength from A to B decreases. Considering these aspects, we assume that A senses B in the case where signal strength from A to B is above a certain threshold.

We use the network topology to build the adjacency graph of the multi-agent system. Let $G = (V, E)$ represent the connectivity of the multi-agent system. In the graph G, every node in V represents a robot. Every edge, say $\{v_1, v_2\} \in E$, indicates that two robots, corresponding to v_1 and v_2, are neighbors. This further implies that the two robots can sense each other. The weight of an edge $\{v_1, v_2\}$ is the relative distance between two robots associated to v_1 and v_2, respectively. If range sensors cannot be utilized, equal weights are used.

Let $G_0 = (V_0, E_0)$ denote the *initial connectivity/interaction graph (at time step $t = 0$)*. We assume that G_0 is connected. We further assume that one robot, say D, is designated as a rendezvous robot, which can be re-assigned at any time.

5.1.3 Control Goal in Graph Theory

The goal of the rendezvous algorithm is to gather every robot in V at the root (rendezvous) robot D, i.e., when $\lim_{t\to\infty} \|\mathbf{q}_D - \mathbf{q}_i\| = 0$, $\forall i \in V(G) - D$.

5.2 Proposed Rendezvous Control Approach

To achieve the above goal, we propose a rendezvous controller using the following procedures: First, let every robot, except for D (the root/leader robot), rendezvous at D while maintaining connectivity; once all robots rendezvous at D, we let D head towards the designated rendezvous location, since D can lead the other robots to the final position.

5.2.1 Rendezvous Algorithm

In Algorithm 5, we present the core rendezvous control approach. The algorithm works as follows. The network graph G is available to all robots in the network (see Algorithm 6 for our approach on building the network graph in a distributed fashion). Based on the graph and the assigned rendezvous robot D, we build the shortest path tree $T = (V, E_T)$ from G rooted at node D, using Dijkstra's algorithm (Lavalle, 2006). Here, the number of edges $|E_T| = |V| - 1$. Since we assume an undirected and connected graph G with non-negative weights, the shortest-path tree T exists and is guaranteed by Algorithm 5. In graphs where all edges have equal weight, T becomes the shortest-hop tree generated by the breadth-first search (BFS) algorithm.

Algorithm 5: Rendezvous Control Algorithm

```
D is designated as the rendezvous location
Get the updated network graph G using Algorithm 6
Given G, we build a shortest-path tree T rooted at D
repeat
    u ← every robot
    if u is associated to a leaf of T then
        u begins heading towards p(u) After encountering p(u), u becomes
          the child of p(p(u))
    else if u is a parent robot with at least one child then
        if u encounters all its children then
            u becomes a leaf node and heads towards p(u)
    if there is a change in neighbor set of a robot (e.g., addition or
      removal of an edge because of a communication interruption) or
      change in the root node assignment then
        Update the network graph G utilizing Algorithm 6
        Update the shortest-path tree T using the updated G and/or D
until all children of D encounter D
```

Every node, say u, that does not have any children is called a *leaf node* in T. We represent the immediate parent of a robot u as $p(u)$. At $t = 0$, all u nodes begin heading towards their parents. These movements initiate all of the rendezvous

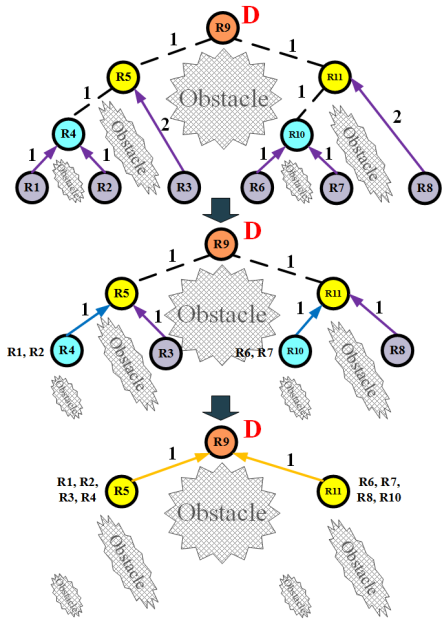


Figure 5.2. Illustration of the procedure following Algorithm 5. The obstacles in the environment are illustrated as red circles.

maneuvers. Since every robot associated with a leaf of T begins moving initially, $p(u)$ encounters all its children as the time goes on.

A robot can decide if it has met its parent or its child using its local sensor information. For example, in our experiments, we use the RSSI from the Wi-Fi receiver as the parameter to decide if it is close enough to encounter its parent or its child by applying a RSSI threshold.

After encountering all its children, $p(u)$ heads toward its parent, say $p(p(u))$. On the other hand, once u encounters its immediate parent $p(u)$, the robot u becomes a child of its next parent in the hierarchy of the tree, say $p(p(u))$, and moves toward $p(p(u))$. This process is repeated until all robots reach the topmost parent, the root robot D, where they should rendezvous. Thus, the above iterative approach is carried out until all robots rendezvous at D, which is the stop condition of the algorithm.

Figure 5.2 illustrates the procedure of the proposed algorithm. At the top of Figure 5.2, the initial graph T of a randomly generated graph is illustrated, in which $R_1, R_2, R_3, R_6, R_7, R_8$ are leaves of T. For instance, R_3 is only connected to R_2, R_6, and R_5. Initially, all nodes head toward their corresponding parents, following the hierarchy in T. The movement of each robot is depicted as an arrow in this figure. In further iterations, R_4 and R_{10} encounter all their children, and then all of them move to their parents R_5 and R_{11}, shown in the middle of Figure 5.2. Finally, after R_5 (or R_{11}) encounters all its children, it begins heading toward D (bottom of the figure). Note that the obstacle boundaries are depicted as red circles. Note that the communication link is preserved during a maneuver.

Thus, the rendezvous is achieved using the iterative tracking movements of the child robots to their parent robots using the hierarchy in the tree T. Using the definition of T, the movement of a child robot towards its parent in T is along the shortest path to D contained in G_0. The velocity control input for an individual robot is driven by its local bearing sensors. See Section 5.2.2 for more details on the robot controller.

To initialize Algorithm 5 following a distributed pattern, each robot senses its neighbors utilizing local sensors and generates a set named neighbor list L_i. Every robot thereafter shares its neighbor list utilizing a distributed algorithm (see Algorithm 6) and generates G utilizing the accumulated list L containing all vertices and edges of the networked system. Algorithm 6 was designed taking inspiration from the distributed consensus algorithm in (Alonso-Mora, Javier, Schwager, & Rus, 2016). As an example, let's consider a cycle graph consisting three nodes a, b, and c. Obviously, L of the cycle graph is the aggregation of the following three unordered lists: $L_a = \{b, c, w_{bc}\}$, $L_b = \{a, c, w_{ac}\}$, and $L_c = \{a, b, w_{ab}\}$. Here, w_{ij} for every edge is calculated utilizing available range sensors of robots. Otherwise, equal weights are used.

In Algorithm 6, N_i is the *neighbor set* of a robot i. Also, $\mathbb{D}(\mathbb{G})$ represents the diameter of G (the number of nodes from D to a node which is the farthest from D

Algorithm 6: Distributed Algorithms for Building a Distance List L

Initialize $L_{i,-1}$ as an empty set
Initialize $L_{i,0}$ as the *local neighbor lists* which are built using robots in N_i
$k = 0$
for *every robot* $i \in V$ **do**
  repeat
    Detect the change in the local list L_i, $\bar{L}_{i,k} = L_{i,k} - L_{i,k-1}$
    Send $\bar{L}_{i,k}$ to all $u_j \in N_i$
    Receive $\bar{L}_{j,k}$ from all $u_j \in N_i$
    $L_{i,k+1} = L_{i,k} + \bar{L}_{j,k}$ for all $u_j \in N_i$
    $k = k + 1$
  until $k \geq \mathbb{D}(G)$

along the shortest path). In this algorithm, a robot shares the update in the local neighbor list with its neighbors. Hence, the global network graph G can be derived in a distributed manner. The convergence to a global list L (hence the global network graph G) at every robot is achieved in a maximum of $\mathbb{D}(G)$ iterations. Therefore, $L_{i,\mathbb{D}(G)} = L \; \forall i \in V$. This statement can be proved similarly to Proposition 1 of (Alonso-Mora et al., 2016). Therefore, the proof is omitted here.

The most severe computation burden of Algorithm 6 relies on $\mathbb{D}(G)$ (the graph diameter of G) and the highest number of edges (neighbors) at every node. From (Caccetta & Smyth, 1992), we note that $\mathbb{D}(\mathbb{G})$ has an upper bound of $\frac{(N-1)}{K}$. Here, K implies the K-connectedness of the graph. In practical applications, the number of robots is significantly smaller than the number of edges in the network, and as a consequence, Algorithm 6 is scalable. Moreover, a dense network (a completely connected graph, for example) leads to a smaller diameter and a larger number of edges when compared to a sparse graph such as a path graph. Hence, the workload of both communication and computation can be well balanced concerning graph density. It is worth noting that Algorithm 6 is used only in the case where the network topology changes (for example, a robot detects a new neighbor), after initially building the list L.

5.2.2 Bearing-aided Robot Velocity Controller

Following the bearing-aided consensus controllers in (Zhao & Zheng, 2017; R. Zheng & Sun, 2014), we devise a velocity controller $\dot{\mathbf{q}}_i = [\dot{x}_i, \dot{y}_i]$ for robot i to head toward its parent robot $p(i)$ as below:

$$\dot{\mathbf{q}}_i = v_{ip} \begin{bmatrix} \cos\alpha_{ip} \\ \sin\alpha_{ip} \end{bmatrix} \tag{5.1}$$

where $\alpha_{ip} = \tan^{-1}\frac{(y_p - y_i)}{(x_p - x_i)}$ represents the relative bearing of the parent robot $p(i)$ (parent of robot i) with respect to the robot i, in the coordinate frame of robot i. We use the rotation matrix of the robot $R(\theta_i) = \begin{vmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{vmatrix}$ to compensate the relative bearing from its own orientation θ_i. Thus, we have $\begin{bmatrix} \cos\alpha_{ip} \\ \sin\alpha_{ip} \end{bmatrix} = R(\theta_i)\frac{(\mathbf{q}_p - \mathbf{q}_i)}{\|\mathbf{q}_p - \mathbf{q}_i\|}$. We assume the robots are equipped with a bearing sensor that provides an estimate of the relative bearing α_{ip}.

The velocity control factor $v_{ip} \leq S_m \in \mathbb{R}^+$ is set to a value proportional to the distance between the robot and its parent robot, i.e., $v_{ip} = f(\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|)$, if a range sensor is available. Otherwise, the v_{ip} is set to a positive constant. For instance, in our experiments, although we did not use a range sensor, we used the RSSI from the Wi-Fi measurements, which is a function of the distance between the nodes, as the function to determine the magnitude of the linear velocity. Note, the velocity is bounded by S_m.

According to (5.1), the robot i moves and converges to its parent $p(i)$ as long as $p(i)$ is stationary, which is guaranteed by Algorithm 5.

In Section 5.4.1, we elaborate on how we use a rotating directional antenna as a bearing sensor that provides an estimate of the relative bearings of the neighboring robots based on the direction of arrival (DOA) of wireless signals. We also describe how we integrate the velocity controller with a bearing-based obstacle avoidance strategy.

5.3 Theoretical Analysis of the Proposed Algorithm

We analyze the theoretical properties of the proposed method such as convergence and connectivity maintenance.

5.3.1 Convergence

We propose a theorem (Theorem 5.3.1) showing that the robots will eventually encounter D over time to complete the rendezvous process.

Theorem 5.3.1 *As $t \to \infty$, every robot's position converges to D using the control law in (5.1), provided that G_0 is a connected graph.*

Proof To prove convergence, we need to show that the distance between all the robots with respect to the root robot will converge. We consider the following Lyapunov candidate function,

$$\mathbb{V} = \sum_{i \in V-D} \|\mathbf{q}_{p(i)} - \mathbf{q}_i\| \tag{5.2}$$

where $p(i)$ is the parent of node i in T. It is clear that $\mathbb{V} = 0$ if and only if all robots encounter D. The time derivative of $\mathbb{V}$ is

$$\dot{\mathbb{V}} = \sum_{i \in V-D} \frac{(\mathbf{q}_{p(i)} - \mathbf{q}_i)^T}{\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|} (\dot{\mathbf{q}}_{p(i)} - \dot{\mathbf{q}}_i). \tag{5.3}$$

In Algorithm 5, the parent node does not maneuver until encountering all its children, i.e., $\dot{\mathbf{q}}_{p(i)} = 0$. Hence, (5.3) leads to

$$\dot{\mathbb{V}} = -\sum_{i \in V-D} \frac{(\mathbf{q}_{p(i)} - \mathbf{q}_i)^T}{\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|} \dot{\mathbf{q}}_i, \tag{5.4}$$

$$\dot{\mathbb{V}} = -\sum_{i \in V-D} v_{ip} \left(\frac{(\mathbf{q}_{p(i)} - \mathbf{q}_i)}{\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|} \right)^T \left(\frac{(\mathbf{q}_{p(i)} - \mathbf{q}_i)}{\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|} \right) \leq 0. \tag{5.5}$$

Using LaSalle invariance principle (La Salle, 1976), the system converges to the set in which $\dot{\mathbb{V}} = 0$. $\dot{\mathbb{V}} = 0$ if and only if all robots encounter D.

In our assumption (Section 5.1.2), two robots encounter each other in the case where $\|\mathbf{q}_i - \mathbf{q}_j\| \leq \epsilon$. Once a robot encounters its parent, it re-assigns its associate parent to the parent with a higher order in the tree. This can generate a jump in $\mathbb{V}$ (a discontinuous fall in $\mathbb{V}$). Nevertheless, the amount of such jumps is finite and upper bounded by the number of robots. Therefore, the number of such discontinuities does not influence the system convergence.

This concludes the proof that all robots $\mathbf{q}_i, \forall i \in V$, converge (rendezvous) at $\mathbf{q}_D$. ■

5.3.2 Connectivity Maintenance

In the next proposition, we show that the proposed algorithm preserves connectivity while performing the rendezvous.

Proposition 5.3.1 *Using Algorithm 5, a robot is connected to at least one other robot in the graph G in time of rendezvous.*

Proof To prove this proposition, we first show that the robots do not lose the connection with their parents and that the robots are within the maximum sensing range R_m of at least one other robot in the tree T.

Since the movement of a robot is heading toward its immediate parent, we first show that every robot i is always detected by its parent $p(i)$ before they encounter each other. As per the Algorithm 5, a parent robot $p(i)$ does not move until all its children satisfy the *merge* condition. Therefore, $\|\dot{\mathbf{q}}_p\| = 0$ if for any $i \in \mathbb{C}_p$, $\|\mathbf{q}_p - \mathbf{q}_i\| > \epsilon$. Here, $\mathbb{C}_p$ is the children set of a robot p. Note that the function $\|\mathbf{q}_p - \mathbf{q}_i\|$ is always non-increasing, as per the algorithm. This way, the movement of the robot toward its parent is restricted along the line segment where the robot always stays connected. Thus, all children robots are sensed by their parent (and vice versa) until they encounter their parents. After a robot i

encounters its parent $p(i)$, both the robot i and its next successive parent $p(p(i))$ can sense each other because
$\|\mathbf{q}_{p(p(i))} - \mathbf{q}_{p(i)}\| \leq R_m - \epsilon$[1]$; and\ \|\mathbf{q}_{p(i)} - \mathbf{q}_i\| \leq \epsilon \implies \|\mathbf{q}_{p(p(i))} - \mathbf{q}_i\| \leq R_m$. Hence, every robot will be sensed by at least one other robot in the tree, and the graph stays connected. This concludes the proof. ■

5.4 Robot Control and Obstacle Avoidance

In this section, we briefly summarize our method, which we adopted from (Min & Matson, 2014), to estimate the direction of arrival (relative bearing) from the RSSI of the rotating directional antenna and use it to control the robot.

5.4.1 Estimation of Relative Bearings

The directional antenna is mounted on the robot's pan servo system and uses a USB Wi-Fi adapter to continuously measure the RSSI of the parent robot's AP. Then, with this measured RSSI, a Weighted Centered Algorithm (WCA) is used to calculate the DOA with the AP of the parent robot. The pan servo system has a scanning range of 180°. With a resolution of 10°, the robot measures the RSSI and records the corresponding angle.

First, a weight is calculated at robot i based on the RSSI measurements from robot j at any instant k.

$$w_k^j = 10^{\left(\frac{RSSI_k^j}{\gamma_1}\right)} \tag{5.6}$$

where γ_1 is a positive constant (empirically determined) that can be adjusted to the nature of the environment. The relative bearing of robot j from robot i is estimated using

$$\widetilde{\alpha}_{ij} = \frac{\sum_{k=1}^{M} w_k^j \theta_k^i}{\sum_{k=1}^{M} w_k^j}, \tag{5.7}$$

[1]Remember that two robots are neighbors if and only if they sense each other while satisfying the relative distance between them being less than or equal to $R_m - \epsilon$.

where θ_k^i is the orientation of the antenna at which $RSSI_k$ is measured, and M is the number of measurements per rotation scan. This DOA is then used to help the child robots track and move toward their parent robots.

This method of estimating the relative bearings has been shown to be reasonably accurate and robust to various environments based on our field experiments in the past (Min & Matson, 2014; Min et al., 2016). For more details on the bearing estimation and control, readers are referred to (Min et al., 2016).

5.4.2 Obstacle Avoidance

In a case where LOS between a robot and its neighbor is blocked by a (big) obstacle, the path connecting the two robots is not traversed using our algorithms. However, there may be a case where there is a small obstacle between two robots and the obstacle does not block the LOS between the two robots. In this case, the two robots become neighbors to each other, and the path that connects the two robots may be traversed using our algorithms. As a robot moves along the path, it may collide with the small obstacle. Thus, we require an additional controller to avoid collision using local sensors.

We assume that unknown obstacles in the working space are convex in shape. To avoid such obstacles (static or dynamic) that are present between a robot and its parent robot, we use the strategy described below.

The robots are equipped with an array of eight ultrasound sensors on the front side of the robot, as mentioned in Section 5.5.1, covering an angular range of 180°. In one scan that happens at 10 Hz, the sonar sensors provide the obstacle distance information over the entire angular range with a resolution of up to 1°. In every 10°, we obtain a distance d_k and compute a weight w_k, defined as

$$w_k = 10^{\left(\frac{-d_k}{\gamma_2}\right)} \tag{5.8}$$

where γ_2 is a positive gain depending on the quality of the sonar signal. The direction guiding the robot to a safe region can be estimated by means of weighted centroid approaches as follows:

$$\widetilde{\alpha}_{obs} = \frac{\sum_{k=1}^{N_s} w_k \theta_k}{\sum_{k=1}^{N_s} w_k} \tag{5.9}$$

where N_s is the number of distance measurements per scan, and θ_k is the angle at instant k where the distance value d_k is measured.

5.4.3 Velocity Control

To control the robots on a Cartesian plane, we apply the control law in (5.1) with a switching controller that selects the α_{ip} as follows:

$$\alpha_{ip} = \begin{cases} \widetilde{\alpha}_{ij}, & \text{if the obstacles are in a safe region,} \\ \widetilde{\alpha}_{obs}, & \text{otherwise.} \end{cases} \tag{5.10}$$

Thus, we integrate both bearings of the parent robot and the obstacles and navigate the robot to a safe region when it should avoid the obstacles first.

The dynamics are then transferred to the internal kinematics of the robots, where a PID motion controller is applied to the left and right motors as it is a differential-drive vehicle.

Also, the velocity of the child robot is set to be proportional to the RSSI (which is a log-normal function of the distance) from the parent robot. The linear velocity control factor $v_{ip} = \omega_1 - RSSI - \omega_2 RSSI$. Here, $RSSI$ represents the RSSI value at the relative bearing angle of the parent, and ω_1 and ω_2 are positive values such that v_{ip} is non-negative. This function is chosen such that when the robot is far from the parent robot, it moves at a higher speed, and moves more slowly when it is closer to the parent.

5.5 Field Experiments

In this section, we first present the experimental setup. Next, we discuss the implementation of the above rendezvous control algorithm integrated with an obstacle avoidance strategy. Then, we present the experiment design and the scenarios tested.

5.5.1 Experiment Setup and Distance Estimation

Figure 5.3 presents the configuration of each robot in detail. All three robots have the same configuration. The base platform is a commercial Pioneer P3AT mobile robot platform. For communication, we install a state-of-the-art, low-cost small wireless AP, *PicoStation* M2-HP, manufactured by *Ubiquiti Networks Inc.* This AP is equipped with a 5-dBi omnidirectional antenna and supports passive Power over Ethernet (PoE), so it does not require an additional power cord. Also, it runs with an IEEE 802.11g protocol on an operating frequency of 2.4 GHz and produces up to 28 dBm output power. In addition to this *PicoStation* AP, we installed a small and light Yagi directional antenna (manufactured by *PCTEL*), connected to a Wi-Fi USB adapter. This antenna is used for measuring the RSSI from different directions (through a rotation tracking system), which is then used for DOA estimation as detailed in Section 5.2.1. The beamwidth of this antenna is 60° at 1/2 power for horizontal and vertical planes. An Asus Eee laptop running Linux mounted on the P3AT robot is used to achieve high-level motion planning. This device has shown strong adaptability for outdoor point-to-point connections (Min et al., 2016, 2018).

To enable robot movements, we adopted the leader-follower robotic system introduced in (Min & Matson, 2014), which is composed of a bearing estimation using a rotational directional antenna and an obstacle avoidance algorithm using an ultrasound sensor array. In the field experiments, the network topology updates at a frequency of 1 Hz to synchronize collected data and update the shortest-path tree T

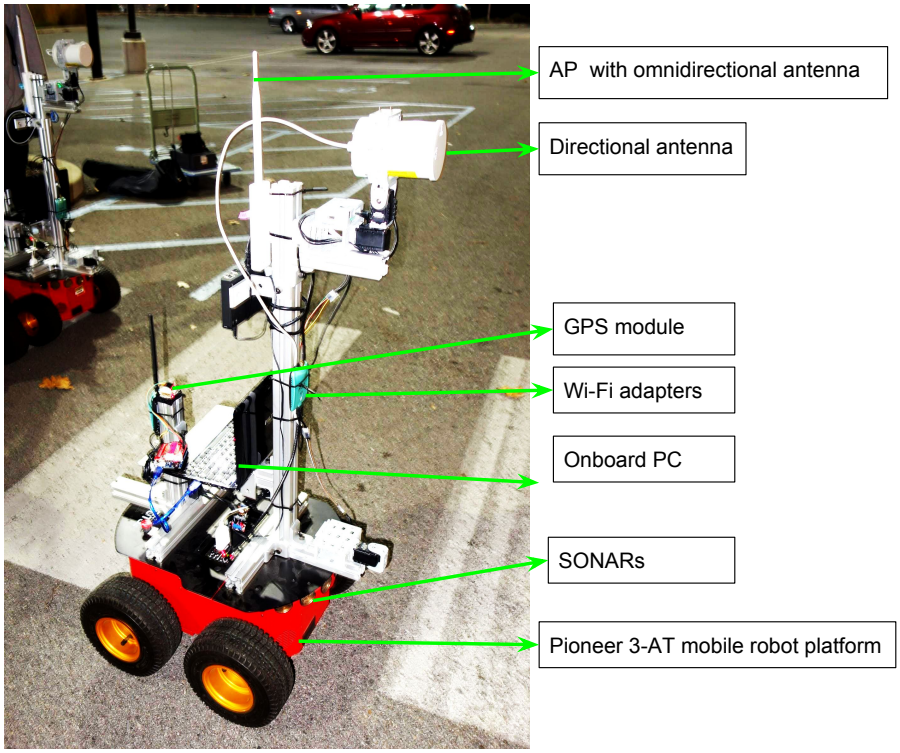


Figure 5.3. The mobile robot platform used in the experiments.

in Algorithm 5 and the distance list L in Algorithm 6. The selection of this update frequency is determined with careful consideration of the moving velocity of the robots and the time required to detect and react to a failed node.

5.5.2 Experiment Design

To test our theory and demonstrate its feasibility, we designed a multi-agent rendezvous experiment involving three mobile robots and a static workstation. A large and obstacle-filled parking lot (approx 5000 m^2) is used as the experiment site. For convenience, we name three robots R_1, R_2, and R_3. We name the static workstation R_0, which is the designated rendezvous point in the experiments. Therefore, all the robots are supposed to gather at R_0 after triggering a rendezvous

signal. We assume that there is no communication between R_0 and R_2 or between R_1 and R_3 due to their communication range limits and the non-direct line of sight.

In our experiments, we set equal edge weights to all edges in the tree, which is not an ideal implementation. However, due to the low density of the network graph (number of edges = 3) in the field experiments, the weights do not play a key role in algorithm performance[2]. When we assume equal weights for all edges, T becomes the shortest-hop tree generated by the BFS algorithm. Thus, each robot moves along the determined shortest-hop path to reach the root node. In this case, the traversal distance of a robot may be long, since the shortest-hop path does not assure the shortest-distance path. Therefore it is desirable to use range information (as the weights in G) obtained from a reliable range sensor, if available.

The Algorithm 6 resulted in the same graph G at all the robots, which is then used in the main algorithm (Algorithm 5). Through our experiments, we verified that the proposed rendezvous algorithm works without any range sensors. The maximum sensing (communication) range R_m is set to 40 m (by applying a minimum RSSI threshold $RSSI_{min} = -80\,\text{dBm}$ at which the robot can reliably communicate with another robot), and the maximum velocity is set to be 0.2 m/s. The maximum RSSI threshold that is used to decide if the robots encounter each other is set to -22 dBm, which approximately corresponds to a threshold distance of $\epsilon = 2$ m.

5.5.3 Experiment Scenarios and Result Analysis

We experimented with three different scenarios. In all the scenarios, the initial communication hierarchy is the same as in Figure 5.4 (left).

[2]Note, in Section 5.6, we simulate the case where every robot is equipped with both range and bearing sensors. The simulation experiments evaluate the algorithm for scalability and make use of the range sensors to determine the weights of the edges.

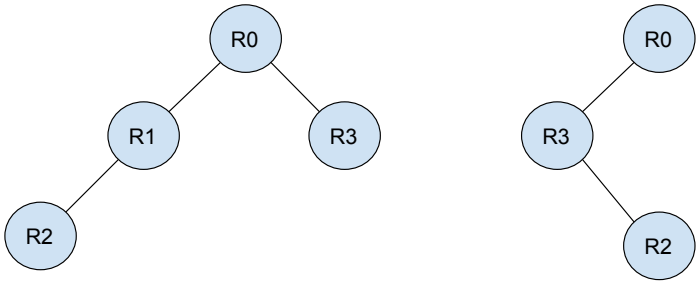


Figure 5.4. Initial hierarchical connected tree for all the three scenarios (left figure). Here, the diameter (depth) of the tree is $\mathbb{D}(T) = 2$. The updated hierarchical connected tree in Scenario 3 is shown on right figure.

Scenario 1

In this scenario, a typical rendezvous behavior is considered. The node R_0 is the rendezvous point, with children R_1 and R_3. The node R_2 has R_1 as its parent. Thus, upon initiating the rendezvous task, the node R_3 starts moving toward R_0, while R_2 moves toward R_1. After R_2 reaches R_1, both R_1 and R_2 move toward R_0, finishing the rendezvous objective.

Figure 5.5(a) shows several images of the experiment in Scenario 1 as a sequence. It shows that R_3 moved toward R_0 and R_2 moved toward R_1 in the images A and B. The images B, C, and D show that after R_2 met R_1 within the threshold distance ϵ, both R_2 and R_1 moved and arrived at R_0. Figure 5.5(b) shows the trajectory taken by the robots in this scenario, where you can also see that all three robots could successfully rendezvous at the designated point.

Figure 5.6(a) represents the changes in RSSI readings (filtered) measured by the rotational directional antenna of the robots (the best RSSI in a 180° scan). R_1 and R_3 have R_0 as its parent (AP), while R_2 has R_1 as its parent (AP). As shown in this figure, the RSSI at R_3 increases as it moves toward R_0. Similarly, the RSSI at R_2 increases as it moves toward R_1. However, after R_2 reaches R_1, R_2 maintains its

(a) Scenario 1 - temporal sequence

(b) Scenario 1 - trajectories

(c) Scenario 2 - temporal sequence

(d) Scenario 2 - trajectories

(e) Scenario 3 - temporal sequence

(f) Scenario 3 - trajectories

Figure 5.5. (a), (c), and (e) show a sequence of stills from experiments in Scenarios 1, 2, and 3. The temporal sequence is sorted alphabetically. (b), (d), and (e) are trajectories of the robots in Scenarios 1, 2, and 3 overlaid on the satellite image of the actual experimental site (a large and obstacle-filled parking lot). The trajectory of R_1 is depicted in red, the trajectory of R_2 is depicted in green, and the trajectory of R_3 is depicted in blue.

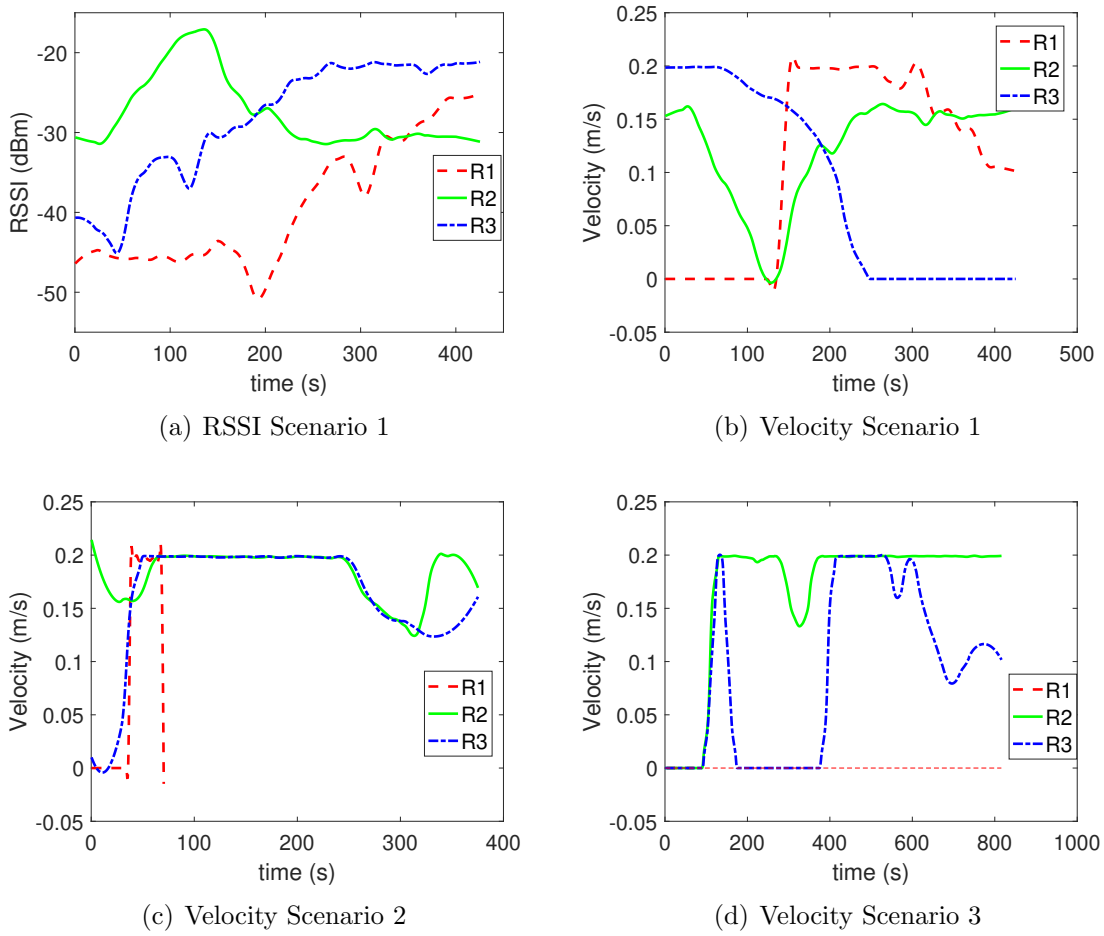
parent as R_1 while R_1 moves to R_0. Thus, R_2 also follows R_1, while the RSSI at R_2 balances with the movement of R_1, but the RSSI at R_1 increases.

In Figure 5.6(b), the corresponding changes in robots' velocity (filtered) are presented. The robot's maximum velocity is set as 0.2 m/s as mentioned above. See around 150 seconds, which shows the node R_1 waited until R_2 reached it, and then both moved toward R_0., whereas R_3 moved toward R_0 from the beginning and arrived early as expected. Note, the velocity was proportional to the relative distance between the parent and the child, as described in Section 5.5.1. Therefore, when the child was farther from the parent initially, the velocity was higher, whereas the velocity was reduced when they were closer.

Scenario 2

This scenario is similar to the previous one, but a node movement fault during the rendezvous task is simulated. That is, the R_1 node fails to move properly but is still able to communicate with other nodes. Therefore, as soon as the node R_2 reaches R_1, R_2 switches to R_0 as its parent and moves toward R_0. On the other hand, R_3 starts moving toward R_0 from the beginning, as R_0 is its direct parent. Thus, at the end of rendezvous, only R_3 and R_2 arrive at R_0, while R_1 is broken and does some random movements around its original position due to its mobility fault.

Figure 5.5(d) shows the trajectory taken by the robots in Scenario 2. Figure 5.6(c) shows the velocity plots for this task. Note that R_1 stopped recording the velocity information after around 80 seconds due to technical reasons. As shown in Figures 5.5(c) and 5.5(d), R_2 and R_3 were able to rendezvous at R_0, while R_1 exhibited chaotic movement and circles around its original position. Notably, from Figure 5.6(c), one can see that the speed of R_2 was decreasing as it approached R_1 from $t = 0$ seconds to $t = 40$ seconds. However, as soon as the parent of R_2 was switched from R_1 to R_0 after the detection of the failure of R_1, the velocity of R_2



(a) RSSI Scenario 1

(b) Velocity Scenario 1

(c) Velocity Scenario 2

(d) Velocity Scenario 3

Figure 5.6. (a) shows the RSSI changes during the rendezvous task in Scenario 1. (b)-(d) show the change in robot velocity during each of the scenarios.

was increasing. This result demonstrates that the proposed rendezvous control strategy can handle fault cases in nodes.

Scenario 3

This is similar to Scenario 2, with the addition of a complete node fault with R_1 (both mobility and communication faults). We simulate the node fault by suddenly powering off R_1 while R_2 is moving toward R_1. Recall that at the

beginning, R_3 moves toward R_0 and R_2 moves toward R_1, and R_1 waits until R_2 reaches it. Thus, when R_1 is switched off, R_2 is unable to communicate and track R_1 in its vicinity, resulting in the network hierarchy being updated with R_1 removed, as can be seen in Figure 5.4 (right). As soon as this update is made, R_2 will switch its parent to R_3, and R_3 will stop moving toward R_0 because it has to wait for its new child R_2. After R_2 reaches R_3, both move towards R_0 and complete the rendezvous task.

Figure 5.5(e) shows a sequence of the experiment, and Figure 5.5(f) shows robots' trajectories in Scenario 3 when the node R_1 was powered off at a random instant before R_2 reached R_1. The robot trajectories were observed as expected in Section 5.5.3, which was also evident in Figure 5.6(d) showing the velocities in one of the trials. Observe that as soon as R_1 was out of the network hierarchy, node R_2 switched its parent to R_3 and moved toward R_3, and R_3 stopped moving until R_2 reached R_3. After that, both R_3 and R_2 moved to R_0.

The results of Scenario 3 demonstrate the advantage of a dynamic hierarchical tree in handling a situation in which one of the nodes, including its ability to communicate, has failed.

5.6 Simulation Experiments

We conducted simulation experiments in MATLAB to evaluate the proposed rendezvous control method on a larger scale (regarding the number of robots). We simulated a 2D environment (plane of 50 m $\times$ 50 m) setup with predefined obstacles. In the MATLAB simulation, we introduced a constraint on the sensing capability; thus one robot can detect another robot only if a LOS path exists between them (i.e., obstacle-free movement). We assumed that both range and bearing sensors are available, and thus we were able to obtain the relative positions of the neighboring robots. Each robot moves along the shortest path to the root,

since the constructed tree is the shortest-path tree (compared to a shortest-hop tree in the field experiments).

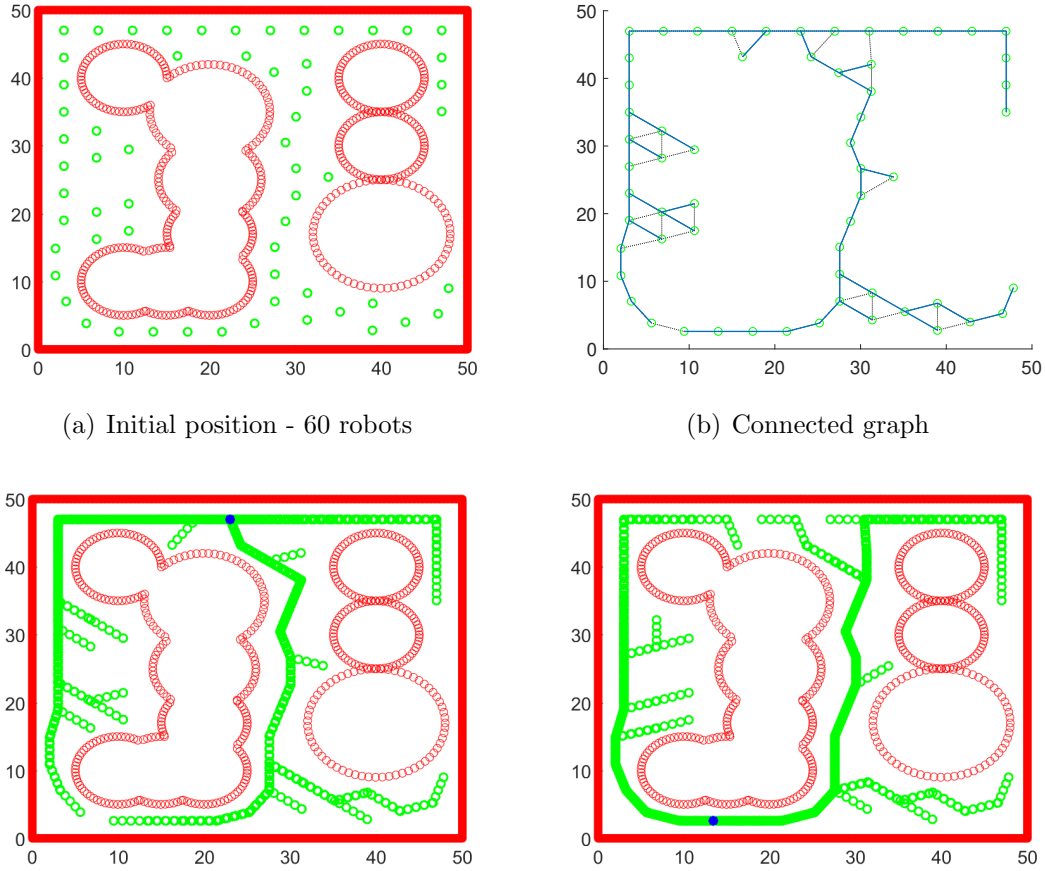
In the simulations, the sampling interval of a robot's velocity controller (5.1) is set as 0.1 seconds. This implies that the velocity of a robot changes at a rate of 10 Hz. Note, we consider a fixed network graph and update the network tree at the same rate of the velocity controller.

The maximum velocity of each robot is $S_m = 0.5$ m/s. We set the sensing range $R_m = 5$ m and distance threshold $\epsilon = 0.1$ m. Other settings are similar to the ones in the field experiments.

We conducted simulation experiments with 60 robots with two trials with different rendezvous points. Figure 5.7 shows the results of the simulations validating the scalability of the proposed method. Figures 5.7(a) and 5.7(b) show the node's position and the resulting connected tree (both G_0 and T) respectively. The outcomes of the robots' trajectories in each trial are shown in Figures 5.7(c) and 5.7(d). Note that each robot moves along T. The only difference is the rendezvous point in both cases. The denser (darker green) the circles are in the plots, the higher the number of robots that visited those positions.

5.7 Multi-point Rendezvous

Let us suppose that the set of leader robots D is selected (or pre-known) depending on their capabilities such as advanced sensing systems, recharging stations, etc. Alternatively, the leader robots can be chosen from a given graph G using the proposed method in Section 5.7.1 which optimizes the maximum traversal distance by any robots in the rendezvous process.



(a) Initial position - 60 robots

(b) Connected graph

(c) Trajectories - Trial 1

(d) Trajectories - Trial 2

Figure 5.7. Simulation with 60 robots is shown. (a) Initial positions of all robots. Green circles are robot nodes and red curves are obstructions for communication or sensing. (b) Connected graph and network hierarchy. All edges are depicted as black dotted-line segments. All shortest-path connections are depicted as blue line segments. (c) and (d) Trajectories of all robots after rendezvous at the designated position (marked as blue asterisk) in different trials.

Since our objective is to gather all robots to their leader robots with the least amount of time and energy, we assign each robot $u_i \in \mathbb{U}$ a leader robot $R(u_i) \in \mathbb{D}$ by optimizing the following objective function:

$$R(u_i) = \arg\min_{D_m \in \mathbb{D}} d_{A(k)}(u_i, D_m). \tag{5.11}$$

Here, the function $d_{A(k)}(u_i, u_j)$ represents the shortest-path distance between u_i and u_j in the graph $A(k)$ using Dijkstra's algorithm (Lavalle, 2006).

Using the initial graph $A(0)$, the robots are grouped together to form M sub-graphs $A_m = (V_m, E_m, W_m) \subset A(0)$, where each sub-graph contains nodes (robots) that satisfy the condition

$$u_i \in V_m \iff R(u_i) = D_m, \forall u_i \in V. \tag{5.12}$$

Note, the sub-graphs can be dynamically reconstructed as the graph evolves, but we consider a special case in which the graph is split only initially to reduce computational load. On the other hand, each sub-graph is dynamically updated $A_m = A_m(k)$.

Using the sub-graphs A_m, we construct M shortest-path trees $T_m = (V_m, E_{T_m}, W_{T_m})$ with root nodes as D_m by optimizing the distance cost (using weights WT_m) from each node to the root node. Each node u_i in a tree is assigned a parent $p(u_i)$ and children (if any) $c(u_i)$. Nodes without any children are the leaf nodes (see Section 5.1.2).

Converting the sub-graphs into tree structures significantly reduces the number of edges from (potentially) $|E_{T_m}| = O(|V_m|^2)$ to $|E_{T_m}| = |V_m| - 1$. Thus, it results in higher computational efficiency.

5.7.1 Choosing Leader Robots (Rendezvous Points)

In situations where the leader robots are not known or cannot be chosen based on their capabilities[3], we present a simple but efficient strategy to optimally choose M leader robots (rendezvous points) among all the robots in the network.

In multi-robot applications, it is highly desirable to achieve rendezvous in the shortest possible time. This goal can be met if we minimize the (anticipated)

[3]An example of such a situation is a homogeneous multi-robot system where all the robots have equal capabilities.

maximum distance traveled by any robot during the rendezvous process. This limits the time and energy consumed by any robot in the system. Therefore, we search the optimal leader robots set D from the set V in the initial graph $A(0)$ by minimizing the following cost function

$$C(D) = \max_{\forall D_m \in D} \max_{\forall u \in V_m} d_{A(0)}(u, D_m), \tag{5.13}$$

subjected to the following constraints

$$L_m \leq |V_m| \leq U_m \tag{5.14}$$

$$d_{A(0)}(u, R(u)) \leq TD, \ \forall u \in (V - D) \tag{5.15}$$

where $|V_m|$ denotes the number of robots in the group in which D_m is the leader after applying the grouping algorithm in (5.11), L_m and U_m are the bounds for $|V_m|$, and TD is a distance bound. The first constraint balances the sizes of all M groups, whereas the second constraint restricts the maximum distance traveled by any robot in the rendezvous task. The rendezvous simulation results using Algorithm 5 are presented in Figure 5.8, where rendezvous points M are 2, 3, and 4.

5.8 Conclusion

We proposed rendezvous control laws for the coordinate-less multi-agent rendezvous problem based on network topology. Our framework included building and dynamically updating a hierarchical network tree. The main characteristics of the proposed control are as follows: boundedness, scalability, and global connectivity preservation. Our control laws can handle realistic application scenarios in cluttered environments and are easy to implement in practice. Additionally, the proposed method can handle faulty cases such as mobility and communication faults without disrupting the whole rendezvous task. Ultimately, we extended our scope to a multi-point rendezvous scenario using the same rendezvous control strategy.

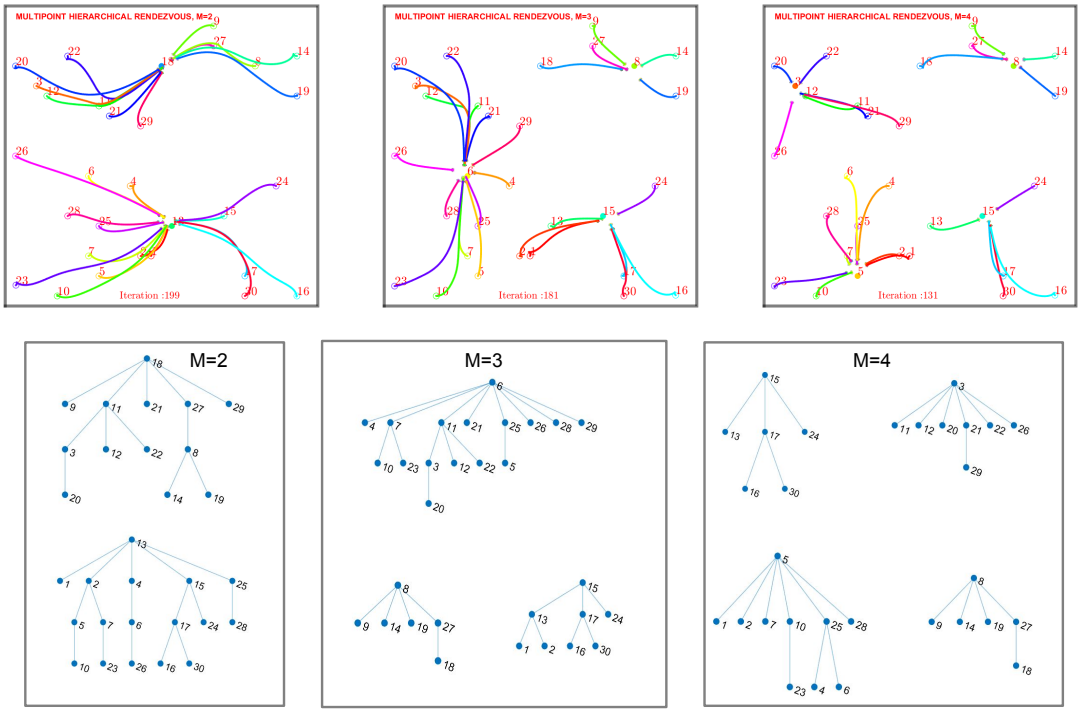


Figure 5.8. Results of multi-point rendezvous with $M = 2$ (left), $M = 3$ (center), and $M = 4$ (right). The initial trees (created using the approach in (5.11), rooted at the leader robots of each group, are depicted in the bottom row.

We conducted extensive field experiments and simulations to validate the proposed control laws under practical application-oriented rendezvous scenarios. In field experiments, we used three mobile robots and demonstrated that the robots could rendezvous at the desired point in different scenarios even if there were mobility and communication faults in one node. In simulations, we further demonstrated the scalability of the proposed control scheme, especially multi-point rendezvous.

Acknowledgment

Sections from Section 5.1 to Section 5.6 are published in (Luo, Kim, et al., 2019). Section 5.7 is published in (Parasuraman et al., 2018).

CHAPTER 6. SUMMARY AND FUTURE WORK

In this dissertation, we addressed the problems inherent to multi-robot coverage control from three aspects: deployment, coverage, and rendezvous. After summarizing previous work, the future research plan is listed in sequence.

6.1 Summary

A multi-robot deployment problem that arose for a team of networked robots was presented in this study. Using remote sensing and image processing techniques, we are able to identify the spills that need to be covered and cleaned. After identification of the workspace and areas (e.g., spills), we determined the optimal allocation of networked robots to the spills using optimization. With a binary ILP solver in MATLAB, we obtained the optimal partition of the robot team and the goal location for each robot. This allocation result yields two advantages: i) It is guaranteed that at least one robot is associated with each spill due to the constraint. ii) The number of robots allocated to a specific spill is in proportion to the spill's area. Furthermore, this result achieves a balanced workload for every deployed robot by assigning a uniform "density" of robots to every spill. As each robot has the same cleaning capability, a uniform density of robots implies a uniform completion time for algae cleaning. That makes the completion time predictable and unique, thus facilitating robot recall after the work is completed. Based on the allocation result, a motion controller with APF method is proposed to navigate the robots to their goals free of collision and deadlock.

For the multi-robot coverage problem, we devised both centralized and distributed coverage control strategies, for different problem settings. Appropriate control laws were designed that motivate the robots to move and cover the spills.

We validated these strategies through intensive simulation experiments in MATLAB and showed some critical features such as scalability, moving spill adaptability, and collision avoidance. In terms of a large-scale spills, we proposed a PRBC method combined with GT. We used the weighted centroid algorithm to estimate the bearing of the pivot robot and used radio signal strength to estimate the distance between robots. The robot team will cover every individual packing area in sequence, regardless of the shape of the patch and packing area. The stability and convergence of the coverage control law is proved mathematically.

For the rendezvous problem, we utilized inter-robot wireless connections to realize rendezvous, which is the same technique used in coverage control. We realized both single-point and multi-point rendezvous with the proposed hierarchical rendezvous algorithms. We further validated the efficacy and fault tolerance of the proposed strategy via both simulation experiments and field tests.

6.2 Future Work

Additional contributions to the current work presented in this dissertation can be done from the following aspects.

In Chapter 3, the robot allocation did not consider the heterogeneity of the robot team. However, it will be desirable to see a team of heterogeneous robots working together and being partitioned based on their uniqueness. The heterogeneous robot deployment can be applied in complex tasks that involve multiple procedures or targets, while certain constraints such as connectivity maintenance are still in effect.

In Chapter 4, more coverage patterns can be realized and performed in the scenario of spill cleaning or USAR. In this research, collision avoidance among robots is still the primary research question, compared with other practical challenges such as connectivity maintenance and team convergence. For environments or robots that are too complicated to be precisely modeled, or for

coverage operations under system uncertainties and unknown disturbances, a Reinforcement Learning (RL)-based multi-robot coverage control has to be implemented. We can manually drive a single robot to complete the training in coverage. Then the robot may start coverage operation autonomously even if there are obstacles. However, the training of a team of robots is still open to further study.

In Chapter 5 and other chapters, the energy constraint for a robot in operation was not considered. In practice, the robot has to move to a place or meet another robot to recharge itself. When robots need to move to the recharging station, guaranteeing that the robot can arrive at the place before running out of power is essential. This can be formulated as a time-constraint rendezvous problem, where power consumption can be formulated as a function of time.

With the research outcome depicted in one or many chapters, we can enable a collective transportation similar to multi-robot caging. This will facilitate item transportation in manufacturing factories and oceanic environments.

Although this dissertation strives for a solution involving autonomous multi-robot operation, there are needs to include humans in the loop to implement an immediate and flexible response to confront abrupt changes in working situations or physical conditions. In the future, I want to explore how a multi-robot team can effectively interact with humans and improve the performance and versatility of human-robot teams, especially in challenging workspaces such as space, deep oceans, and others.

LIST OF REFERENCES

LIST OF REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... others (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P. N., & Hull, D. (2002). Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, *21*(4), 331–344.

Almeida, J., Ferreira, A., Matias, B., Dias, A., Martins, A., Silva, F., ... others (2016). Air and underwater survey of water enclosed spaces for vamos! project. In *Oceans 2016 mts/ieee monterey* (pp. 1–5).

Alonso-Mora, Javier, E. M., Schwager, M., & Rus, D. (2016). Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In *Robotics and automation (icra), 2016 ieee international conference on* (p. 1-8).

An, V., Qu, Z., & Roberts, R. (2017). A rainbow coverage path planning for a patrolling mobile robot with circular sensing range. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

An, V., Qu, Z., & Roberts, R. (2018). A rainbow coverage path planning for a patrolling mobile robot with circular sensing range. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(8), 1238–1254.

Ando, H., Oasa, Y., Suzuki, I., & Yamashita, M. (1999). Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, *15*(5), 818–828.

Bae, J. H., Luo, S., Kannan, S. S., Singh, Y., Lee, B., Voyles, R. M., ... Min, B.-C. (2019). Development of an unmanned surface vehicle for remote sediment sampling with a *van veen* grab sampler. In *2019 mts/ieee oceans.*

Batalin, M. A., & Sukhatme, G. S. (2002). Spreading out: A local approach to multi-robot coverage. In *Distributed autonomous robotic systems 5* (pp. 373–382). Springer.

Bednorz, W. (2008). Advances in greedy algorithms. *Wienna: I-Tech Education and Publishing KG*, *14*(6).

Bella, S., Belbachir, A., & Belalem, G. (2018). A hybrid architecture for cooperative uav and usv swarm vehicles. In *International conference on machine learning for networking* (pp. 341–363).

Bhattacharya, S., Ghrist, R., & Kumar, V. (2014). Multi-robot coverage and exploration on riemannian manifolds with boundaries. *The International Journal of Robotics Research*, *33*(1), 113–137.

Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Improving object detection with one line of code. *CoRR*, *abs/1704.04503*. Retrieved from `http://arxiv.org/abs/1704.04503`

Breitenmoser, A., Schwager, M., Metzger, J.-C., Siegwart, R., & Rus, D. (2010). Voronoi coverage of non-convex environments with a group of networked robots. In *Robotics and automation (icra), 2010 ieee international conference on* (pp. 4982–4989).

Brunskill, E., Kollar, T., & Roy, N. (2007). Topological mapping using spectral clustering and classification. In *Intelligent robots and systems, 2007. iros 2007. ieee/rsj international conference on* (pp. 3491–3496).

Bullo, F., Cortés, J., & Martinez, S. (2009). *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press.

Burgard, W., Moors, M., Fox, D., Simmons, R., & Thrun, S. (2000). Collaborative multi-robot exploration. In *Robotics and automation, 2000. proceedings. icra'00. ieee international conference on* (Vol. 1, pp. 476–481).

Caccetta, L., & Smyth, W. (1992). Graphs of maximum diameter. *Discrete Mathematics*, *102*(2), 121 - 141. doi: https://doi.org/10.1016/0012-365X(92)90047-J

Câmara, G., Cartaxo, R., Souza, M., Freitas, U. M., Garrido, J., & Ii, F. M. (1996). Spring: Integrating remote sensing and gis by object-oriented data modelling. *Computers & graphics*, *20*(3), 395–403.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*(6), 679–698.

Casbeer, D. W., Kingston, D. B., Beard, R. W., & McLain, T. W. (2006). Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, *37*(6), 351–360.

Chang, H.-C., & Wang, L.-C. (2010). A simple proof of thue's theorem on circle packing. *arXiv preprint arXiv:1009.4322*.

Chen, S.-H. (2010). *Multi-agent applications with evolutionary computation and biologically inspired technologies: Intelligent techniques for ubiquity and optimization: Intelligent techniques for ubiquity and optimization*. IGI Global.

Choi, Y.-H., Lee, T.-K., Baek, S.-H., & Oh, S.-Y. (2009). Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *Intelligent robots and systems, 2009. iros 2009. ieee/rsj international conference on* (pp. 5788–5793).

Choset, H., & Burdick, J. (2000). Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, *19*(2), 96–125.

Choset, H. M. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Cortés, J., Martínez, S., & Bullo, F. (2006). Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, *51*(8), 1289–1298.

Danner, T., & Kavraki, L. E. (2000). Randomized planning for short inspection paths. In *Proceedings 2000 icra. millennium conference. ieee international conference on robotics and automation. symposia proceedings (cat. no. 00ch37065)* (Vol. 2, pp. 971–976).

De Hoog, J., Cameron, S., Visser, A., et al. (2010). Selection of rendezvous points for multi-robot exploration in dynamic environments. In *International conference on autonomous agents and multi-agent systems (aamas).*

Dimarogonas, D. V., & Johansson, K. H. (2008). Decentralized connectivity maintenance in mobile networks with bounded inputs. In *Proc. of ieee international conference on robotics and automation* (p. 1507 - 1512). USA.

Dimarogonas, D. V., & Kyriakopoulos, K. J. (2007). On the rendezvous problem for multiple nonholonomic agents. *IEEE Transactions on automatic control*, *52*(5), 916–922.

Douglas, B. W. (2001). *Introduction to graph theory* (2nd ed.). Illinois, USA: Prentice Hall.

Dudek, G., & Roy, N. (1997). Multi-robot rendezvous in unknown environments, or, what to do when you're lost at the zoo. In *Proceedings of the aaai national conference workshop on online search, providence, rhode island.*

Dunbabin, M., & Marques, L. (2012). Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, *19*(1), 24–39.

Elango, M., Nachiappan, S., & Tiwari, M. K. (2011). Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms. *Expert Systems with Applications*, *38*(6), 6486–6491.

Englert, M., Röglin, H., & Vöcking, B. (2007). Worst case and probabilistic analysis of the 2-opt algorithm for the tsp. In *Proceedings of the eighteenth annual acm-siam symposium on discrete algorithms* (pp. 1295–1304).

Fink, J., & Kumar, V. (2010). Online methods for radio signal mapping with mobile robots. In *Robotics and automation (icra), 2010 ieee international conference on* (pp. 1940–1945).

Ganguli, A., Cortés, J., & Bullo, F. (2008). Distributed coverage of nonconvex environments. In *Networked sensing information and control* (pp. 289–305). Springer.

Ganguli, A., Cortés, J., & Bullo, F. (2009). Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, *25*(2), 340–352.

Gowal, S., Prorok, A., & Martinoli, A. (2011). Two-phase online calibration for infrared-based inter-robot positioning modules. In *Intelligent robots and systems (iros), 2011 ieee/rsj international conference on* (pp. 3313–3319).

Heppes, A. (2003). Some densest two-size disc packings in the plane. *Discrete & Computational Geometry*, *30*(2), 241–262.

Honegger, D., Meier, L., Tanskanen, P., & Pollefeys, M. (2013). An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *2013 ieee international conference on robotics and automation* (pp. 1736–1741).

Hsu, P.-M., Lin, C.-L., & Yang, M.-Y. (2014). On the complete coverage path planning for mobile robots. *Journal of Intelligent & Robotic Systems*, *74*(3-4), 945–963.

Jin, X., & Ray, A. (2014a). Navigation of autonomous vehicles for oil spill cleaning in dynamic and uncertain environments. *International Journal of Control*, *87*(4), 787–801.

Jin, X., & Ray, A. (2014b, April). Navigation of autonomous vehicles for oil spill cleaning in dynamic and uncertain environments. *International Journal of Control*, *87*, 787–801.

Jo, W., Hoashi, Y., Aguilar, L. L. P., Postigo-Malaga, M., Garcia-Bravo, J. M., & Min, B.-C. (2019). A low-cost and small usv platform for water quality monitoring. *HardwareX*, *6*, e00076.

Jo, W., Park, J. H., Hoashi, Y., & Min, B.-C. (2019). Development of an unmanned surface vehicle for harmful algae removal. In *2019 mts/ieee oceans.*

Jung, S., Cho, H., Kim, D., Kim, K., Han, J.-I., & Myung, H. (2017). Development of algal bloom removal system using unmanned aerial vehicle and surface vehicle. *IEEE Access*, *5*, 22166–22176.

Kakalis, N. M., & Ventikos, Y. (2008). Robotic swarm concept for efficient oil spill confrontation. *Journal of hazardous materials*, *154*(1-3), 880–887.

Kakalis, N. M. P., & Ventikos, Y. (2008, June). Robotic swarm concept for efficient oil spill confrontation. *Journal of Hazardous Materials*, *154*(1), 880–887.

Kapur, S., & Thakkar, N. (2015). *Mastering opencv android application programming.* Packt Publishing Ltd.

Keshmiri, S. (2011). Multi-robot, multi-rendezvous recharging paradigm: An opportunistic control strategy. In *Robotic and sensors environments (rose), 2011 ieee international symposium on* (pp. 31–36).

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

Kong, C. S., Peng, N. A., & Rekleitis, I. (2006). Distributed coverage with multi-robot system. In *Robotics and automation, 2006. icra 2006. proceedings 2006 ieee international conference on* (pp. 2423–2429).

Konolige, K., Agrawal, M., Bolles, R. C., Cowan, C., Fischler, M., & Gerkey, B. (2008). Outdoor mapping and navigation using stereo vision. In *Experimental robotics* (pp. 179–190).

Kumar, M., Garg, D. P., & Kumar, V. (2010). Segregation of heterogeneous units in a swarm of robotic agents. *IEEE Transactions on Automatic Control*, *55*(3), 743–748.

Kunwar, F., & Benhabib, B. (2006). Rendezvous-guidance trajectory planning for robotic dynamic obstacle avoidance and interception. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *36*(6), 1432–1441.

LaSalle, J. P. (1976). *The stability of dynamical systems* (Vol. 25). Siam.

La Salle, J. P. (1976). *The stability of dynamical systems*. SIAM.

Lavalle, S. M. (2006). *Planning algorithms*. Cambridge University Press.

Li, B., Moridian, B., & Mahmoudian, N. (2016). Underwater multi-robot persistent area coverage mission planning. In *Oceans 2016 mts/ieee monterey* (pp. 1–6).

Litus, Y., Zebrowski, P., & Vaughan, R. T. (2009). A distributed heuristic for energy-efficient multirobot multiplace rendezvous. *IEEE Transactions on Robotics*, *25*(1), 130–135.

Liu, Y., & Nejat, G. (2013). Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, *72*(2), 147–165.

Liu, Y., & Weisberg, R. H. (2011). A review of self-organizing map applications in meteorology and oceanography.

Lots, J.-F., Lane, D. M., Trucco, E., & Chaumette, F. (2001). A 2d visual servoing for underwater vehicle station keeping. In *Robotics and automation, 2001. proceedings 2001 icra. ieee international conference on* (Vol. 3, pp. 2767–2772).

Luo, S., Bae, J. H., & Min, B.-C. (2018). Pivot-based collective coverage control with a multi-robot team. In *2018 ieee international conference on robotics and biomimetics (robio)* (pp. 2367–2372).

Luo, S., Kim, J., & Min, B.-C. ((Under review)). Asymptotic boundary shrink control with multi-robot systems. In *Ieee transactions on systems, man, and cybernetics: Systems.*

Luo, S., Kim, J., Parasuraman, R., Bae, J. H., Matson, E. T., & Min, B.-C. (2019). Multi-robot rendezvous based on bearing-aided hierarchical tracking of network topology. *Ad Hoc Networks*, *86*, 131–143.

Luo, S., Singh, Y., Yang, H., Bae, J. H., Dietz, J. E., Diao, X., & Min, B.-C. (2019). Image processing and model-based spill coverage path planning for unmanned surface vehicles. In *2019 mts/ieee oceans.*

Makris, S., Michalos, G., & Chryssolouris, G. (2012). Rfid driven robotic assembly for random mix manufacturing. *Robotics and Computer-Integrated Manufacturing*, *28*(3), 359–365.

Mathew, N., Smith, S. L., & Waslander, S. L. (2013). A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *Robotics and automation (icra), 2013 ieee international conference on* (pp. 3497–3502).

Mathew, N., Smith, S. L., & Waslander, S. L. (2015). Multirobot rendezvous planning for recharging in persistent tasks. *IEEE Transactions on Robotics*, *31*(1), 128–142.

Meghjani, M., & Dudek, G. (2011). Combining multi-robot exploration and rendezvous. In *Computer and robot vision (crv), 2011 canadian conference on* (pp. 80–85).

Meghjani, M., & Dudek, G. (2012). Multi-robot exploration and rendezvous on graphs. In *Intelligent robots and systems (iros), 2012 ieee/rsj international conference on* (pp. 5270–5276).

Meghjani, M., & Dudek, G. (2014). Multi-agent rendezvous on street networks. In *Robotics and automation (icra), 2014 ieee international conference on* (pp. 5792–5797).

Min, B.-C., Lewis, J., Matson, E. T., & Smith, A. H. (2013). Heuristic optimization techniques for self-orientation of directional antennas in long-distance point-to-point broadband networks. *Ad Hoc Networks*, *11*(8), 2252–2263.

Min, B.-C., & Matson, E. T. (2014). Robotic follower system using bearing-only tracking with directional antennas. In *Robot intelligence technology and applications 2* (pp. 37–58). Springer.

Min, B.-C., Matson, E. T., & Jung, J.-W. (2016). Active antenna tracking system with directional antennas for enhancing wireless communication capabilities of a networked robotic system. *Journal of Field Robotics*, *33*(3), 391–406.

Min, B.-C., Parasuraman, R., Lee, S., Jung, J.-W., & Matson, E. T. (2018). A directional antenna based leader–follower relay system for end-to-end robot communications. *Robotics and Autonomous Systems*, *101*, 57–73.

Mitiche, A., & Aggarwal, J. K. (2014). *Computer vision analysis of image motion by variational methods*. Springer.

Morgenthal, G., & Hallermann, N. (2014). Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures. *Advances in Structural Engineering*, *17*(3), 289–302.

Mukhopadhyay, S., Wang, C., Patterson, M., Malisoff, M., & Zhang, F. (2014). Collaborative autonomous surveys in marine environments affected by oil spills. In *Cooperative robots and sensor networks 2014* (pp. 87–113). Springer.

Narasimhan, S. G., & Nayar, S. K. (2003). Shedding light on the weather. In *2003 ieee computer society conference on computer vision and pattern recognition, 2003. proceedings.* (Vol. 1, pp. I–I).

Narkhede, P., Joseph Raj, A. N., Kumar, V., Karar, V., & Poddar, S. (2018). Least square estimation-based adaptive complimentary filter for attitude estimation. *Transactions of the Institute of Measurement and Control*, 1-11.

Ngo, T. D., Raposo, H., & Schiøler, H. (2008). Multiagent robotics: toward energy autonomy. *Artificial Life and Robotics*, *12*(1-2), 47–52.

Olfati-Saber, R., & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, *49*(9), 1520–1533.

Parasuraman, R., Kim, J., Luo, S., & Min, B.-C. (2018). Multipoint rendezvous in multirobot systems. *IEEE transactions on cybernetics*.

Penmetcha, M., Luo, S., Samantaray, A., Dietz, J. E., Yang, B., & Min, B.-C. (2019). Computer vision-based algae removal planner for multi-robot teams. In *2019 ieee international conference on systems, man, and cybernetics (smc).*

Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., & Egerstedt, M. (2017). The robotarium: A remotely accessible swarm robotics research testbed. In *Robotics and automation (icra), 2017 ieee international conference on* (pp. 1699–1706).

Pickem, D., Lee, M., & Egerstedt, M. (2015). The gritsbot in its natural habitat-a multi-robot testbed. In *Robotics and automation (icra), 2015 ieee international conference on* (pp. 4062–4067).

Pinto, E., Santana, P., Marques, F., Mendonça, R., Lourenço, A., & Barata, J. (2014). On the design of a robotic system composed of an unmanned surface vehicle and a piggybacked vtol. In *Doctoral conference on computing, electrical and industrial systems* (pp. 193–200).

Pyo, J. C., Ligaray, M., Kwon, Y. S., Ahn, M.-H., Kim, K., Lee, H., ... Cho, K. H. (2018). High-spatial resolution monitoring of phycocyanin and chlorophyll-a using airborne hyperspectral imagery. *Remote Sensing*, *10*(8), 1180.

Qutub, S., Alami, R., & Ingrand, F. (1997). How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. In *Intelligent robots and systems, 1997. iros'97., proceedings of the 1997 ieee/rsj international conference on* (Vol. 3, pp. 1610–1615).

Rekleitis, I., New, A. P., & Choset, H. (2005). Distributed coverage of unknown/unstructured environments by mobile sensor networks. In *Multi-robot systems. from swarms to intelligent automata volume iii* (pp. 145–155). Springer.

Rekleitis, I., New, A. P., Rankin, E. S., & Choset, H. (2008). Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, *52*(2-4), 109–142.

MIT Senseable City Lab. (n.d.). `http://senseable.mit.edu/seaswarm/`. (Accessed: July 31, 2018)

Roy, N., & Dudek, G. (2001). Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, *11*(2), 117–136.

Sabattini, L., Secchi, C., Chopra, N., & Gasparri, A. (2013). Distributed control of multirobot systems with global connectivity maintenance. *IEEE Transactions on Robotics*, *29*(5), 1326 - 1332.

Sattar, J., Giguere, P., Dudek, G., & Prahacs, C. (2005). A visual servoing system for an aquatic swimming robot. In *Intelligent robots and systems, 2005.(iros 2005). 2005 ieee/rsj international conference on* (pp. 1483–1488).

Saxena, A., Chung, S. H., & Ng, A. Y. (2006). Learning depth from single monocular images. In *Advances in neural information processing systems* (pp. 1161–1168).

Schwager, M., Rus, D., & Slotine, J.-J. (2009). Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, *28*(3), 357–375.

Schwager, M., Rus, D., & Slotine, J.-J. (2011). Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *The International Journal of Robotics Research*, *30*(3), 371–383.

Silverman, M. C., Nies, D., Jung, B., & Sukhatme, G. S. (2002). Staying alive: A docking station for autonomous robot recharging. In *Robotics and automation, 2002. proceedings. icra'02. ieee international conference on* (Vol. 1, pp. 1050–1055).

Simmons, R., Apfelbaum, D., Fox, D., Goldman, R. P., Haigh, K. Z., Musliner, D. J., ... Thrun, S. (2000). Coordinated deployment of multiple, heterogeneous robots. In *Intelligent robots and systems, 2000.(iros 2000). proceedings. 2000 ieee/rsj international conference on* (Vol. 3, pp. 2254–2260).

Smith, L., Smith, M., & Ashcroft, P. (2011). Analysis of environmental and economic damages from british petroleum's deepwater horizon oil spill.

Soltero, D. E., Smith, S. L., & Rus, D. (2011). Collision avoidance for persistent monitoring in multi-robot systems with intersecting trajectories. In *Intelligent robots and systems (iros), 2011 ieee/rsj international conference on* (pp. 3645–3652).

Song, J., Gupta, S., Hare, J., & Zhou, S. (2013). Adaptive cleaning of oil spills by autonomous vehicles under partial information. In *Oceans-san diego, 2013* (pp. 1–5).

Um, D., Ryu, D., & Kal, M. (2011). Multiple intensity differentiation for 3-d surface reconstruction with mono-vision infrared proximity array sensor. *IEEE Sensors Journal*, *11*(12), 3352–3358.

Valbuena, L., & Tanner, H. G. (2012). Hybrid potential field based control of differential drive mobile robots. *Journal of intelligent & robotic systems*, *68*(3-4), 307–322.

Van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and automation, 2008. icra 2008. ieee international conference on* (pp. 1928–1935).

Viguria, A., Maza, I., & Ollero, A. (2010). Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions. *Advanced Robotics*, *24*(1-2), 1–23.

Villmann, T., & Bauer, H.-U. (1998). Applications of the growing self-organizing map. *Neurocomputing*, *21*(1-3), 91–100.

Weerakoon, T., Ishii, K., & Nassiraei, A. A. F. (2015). An artificial potential field based mobile robot navigation method to prevent from deadlock. *Journal of Artificial Intelligence and Soft Computing Research*, *5*(3), 189–203.

Xiao, X., Dufek, J., Woodbury, T., & Murphy, R. (2017). Uav assisted usv visual navigation for marine mass casualty incident response. In *2017 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 6105–6110).

Zavlanos, M. M., Egerstedt, M. B., & Pappas, G. J. (2011). Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, *99*(9), 1525–1540.

Zebrowsk, P., & Vaughan, R. T. (2005). Recharging robot teams: A tanker approach. In *Advanced robotics, 2005. icar'05. proceedings., 12th international conference on* (pp. 803–810).

Zhang, F., & Leonard, N. E. (2010). Cooperative filters and control for cooperative exploration. *IEEE Transactions on Automatic Control*, *55*(3), 650–663.

Zhang, G., Fricke, G. K., & Garg, D. P. (2013). Spill detection and perimeter surveillance via distributed swarming agents. *IEEE/asme Transactions on Mechatronics*, *18*(1), 121–129.

Zhao, S., & Zheng, R. (2017, July). Flexible bearing-only rendezvous control of mobile robots. In *2017 36th chinese control conference (ccc)* (p. 8051-8056). doi: 10.23919/ChiCC.2017.8028630

Zheng, R., & Sun, D. (2014). Multirobot rendezvous with bearing-only or range-only measurements. *Robotics and Biomimetics*, *1*(1), 4.

Zheng, X., Koenig, S., Kempe, D., & Jain, S. (2010). Multirobot forest coverage for weighted and unweighted terrain. *IEEE Transactions on Robotics*, *26*(6), 1018–1031.

Zhou, X. S., & Roumeliotis, S. I. (2006). Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *Intelligent robots and systems, 2006 ieee/rsj international conference on* (pp. 1785–1792).

Zhou, Y., HU HS, L., et al. (2017). A real-time and fully distributed approach to motion planning for multirobot systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

VITA

VITA

Shaocheng Luo was born in Hotan, China, in 1986. He received a B.S. degree in Mechanical Engineering from Harbin Institute of Technology in China in 2009, and an M.S. degree in Mechantronic Engineering from Zhejiang University in China in 2010. He is currently pursuing his Ph.D. in Technology with a specialization in Robotics at Purdue University in West Lafayette, Indiana, USA. His research interests span the areas of multi-robot systems, wireless networks, robotic system applications in environmental monitoring and operations, and cyber-physical systems.

Refereed Publications

- **Shaocheng Luo**, Jonghoek Kim, and Byung-Cheol Min, "Asymptotic Boundary Shrink Control with Multi-robot Systems", *IEEE Transactions on Systems, Man, and Cybernetics: Systems.* (With Minor Revision)
- Tamzidul Mina, Wonse Jo, Shyam Sundar Kannan, **Shaocheng Luo**, Galen B. King, and Byung-Cheol Min, "Distributed Multi-robot Arbitrary Object Transportation", *IEEE Robotics and Automation Letters.* (Under Review)
- Ramviyas Parasuraman, Jonghoek Kim, **Shaocheng Luo**, and Byung-Cheol Min, "Multi-Point Rendezvous in Multi-Robot Systems", *IEEE Transactions on Cybernetics,* Vol. 50, Issue 1, pp. 310-323, Jan. 2020.
- **Shaocheng Luo**, Yogang Singh, Hanyao Yang, Jun Han Bae, J. Eric Dietz, Xiumin Diao, and Byung-Cheol Min, "Image Processing and Model-Based Spill Coverage Path Planning for Unmanned Surface Vehicles", *2019 MTS/IEEE OCEANS,* Seattle, WA, USA, October 27-31, 2019.

- Jun Han Bae, **Shaocheng Luo**, Shyam Sundar Kannan, Yogang Singh, Bumjoo Lee, Richard M. Voyles, Mauricio Postigo-Malaga, Edgar Gonzales Zenteno, Lizbeth Paredes Aguilar, and Byung-Cheol Min, "Development of an Unmanned Surface Vehicle for Remote Sediment Sampling with a Van Veen Grab Sampler", *2019 MTS/IEEE OCEANS,* Seattle, WA, USA, October 27-31, 2019.
- Manoj Penmetcha, **Shaocheng Luo**, Arabinda Samantaray, J. Eric Dietz, Baijian Yang, and Byung-Cheol Min, "Computer Vision-based Algae Removal Planner for Multi-robot Teams", *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC),* Bari, Italy, 6-9 October, 2019.
- **Shaocheng Luo**, Jonghoek Kim, Ramviyas Parasuraman, Jun Han Bae, Eric T Matson, and Byung-Cheol Min, "Multi-robot Rendezvous Based on Bearing-aided Hierarchical Tracking of Network Topology", *Ad Hoc Networks,* Vol. 86, pp. 131-143, April 2019.
- **Shaocheng Luo**, Jun Han Bae, and Byung-Cheol Min, "Pivot-based Collective Coverage Control with a Multi-robot Team", *2018 IEEE International Conference on Robotics and Biomimetics (IEEE ROBIO 2018),* Kuala Lumpur, Malaysia, December 12-15, 2018.
- Walter D. Leon-Salas, Thomas Fischer, Xiaozhe Fan, Golsa Moayeri, and **Shaocheng Luo**, "A 64x64 Image Energy Harvesting Configurable Image Sensor", *IEEE International Symposium on Circuit and Systems (ISCAS 2016),* Montreal, Canada, May 22-25, 2016.
- Xinglai Jin, Shiqiang Zhu, Wenxiang Wu, and **Shaocheng Luo**, "A Novel Robotic Motion Control Strategy Based on Improved Fuzzy PID and Feedforward Compensation", *Applied Mechanics and Materials,* Vol. 365-366, p. 821-826, 2013.

- **Shaocheng Luo**, Shiqiang Zhu, and Huifang Wang, "Adaptive PID saturate output feedback control of robot manipulators", *Transducer and Microsystem Technologies (Chinese)*, Vol. 31 (3), p. 66-70, 2012.
- **Shaocheng Luo** and Shiqiang Zhu, "Open Architecture Multi-Axis Motion Control System for Industrial Robot Based on CAN Bus", *Automatic Control and Artificial Intelligence (ACAI 2012)*, Xiamen, China, March 24-26, 2012.