UNIVERSITY OF OTTAWA

OTTAWA-CARLETON INSTITUTE FOR MECHANICAL AND AEROSPACE ENGINEERING

Distributed Coverage Control of Multi-Agent System in Convective–Diffusive Time Evolving Environments

Author:
Jian Mei

Supervisor:
Davide Spinello

A thesis submitted in partial fulfillment of the requirements for the Master of Applied Science degree
in
Mechanical Engineering

uOttawa

Distributed Coverage Control of Multi-Agent System in Convective–Diffusive Time Evolving Environments

Jian Mei

Abstract

Using multi-agent systems to execute a variety of missions such as environmental monitoring and target tracking has been made possible by the advances in control techniques and computational capabilities. Communication abilities between agents allow them to coact and execute several coordinated missions, among which there is optimal coverage. The optimal coverage problem has several applications in engineering theory and practice, as for example in environmental monitoring, which belongs to the broad class of resource allocation problems, in which a finite number of mobile agents have to be deployed in a given spatial region with the assignment of a sub-region to each agents with respect to a suitable coverage metric. The coverage metric encodes the sensing performance of individual agent with respect to points inside the domain of interest, and a distribution of risk density. Usually the risk density function measures the relative importance assigned to inner regions.

The optimal coverage problem in which the risk density is time-invariant has been widely studied in previous research. The solution to this class of problems is centroidal Voronoi tessellation, in which each agent is located on the centroid of the related Voronoi cell. However, there are many scenarios that require to be modelled by time-varying risk density rather than time-invariant one, as for example in area coverage problems where the environment evolves independently of the evolution fo the robotic agents deployed to cover the area.

In this work, the changing environment is modeled by a time-varying density function which is governed by a convection-diffusion equation. Mixed boundary conditions are considered to model a scenario in which a diffusive substance (e.g., oil from a leaking event or radioactive material from a nuclear accident) enters the area with convective component from the boundary. A non-autonomous feedback law is employed whose generated trajectories maximize the coverage metric. The asymptotic stability of the multi-agent system is proven by using Barbalat's lemma, and then theoretical predictions are illustrated by several simulations that represent idealized scenarios.

Dedication

I would like to dedicate my thesis to my parents, who help and encourage me despite many obstacles throughout my life, and to my girlfriend Fei Chen, who has accompanied me all along.

Acknowledgments

Foremost, I would like to express my gratitude to Dr. Davide Spinello for his continuous support and guidance, for his advice, which helped me throughout my master study, and for his supervision, which helped me to complete the thesis.

In addition, I would like to thank my parents, Mr. ZhanSheng Mei and Mrs. HanMeng Zhang; they not only give me financial support but also give me Spiritual encouragement.

Finally, I would like to thank my girlfriend, Fei Chen, for her presence in my life.

Contents

List of Figures

List of Matlab Routines

Chapter 1

Introduction

A Wireless Sensor Network (WSN) is a group of spatially distributed sensor nodes which can collect, store and share data with each other. In recent years, advances in computation and communication capabilities have allowed WSNs to be widely used in various tasks, such as target tracking, task assignment, and area coverage control. [1]. Area coverage control is an application of great interest in the environmental monitoring field, such as water contamination monitoring or landslide detection [2].

The area coverage tasks involves the deployment of a group of mobile agents inside a space of interest to cover the space optimally, or to mathematically maximize the coverage metric [3]. The coverage metric encodes the sensing performance of individual agents with respect to points inside the domain of interest, and a risk distribution defined in the domain, which measures the relative importance assigned to inner regions. In some tasks, the density is defined as the probability of random events occurring in the domain, so that maximizing the coverage metric is also maximizing the future event detection [4].

1.1 Motivation

Recently, the area coverage control with WSNs has received increasing attention due to the wide potential of its applications. For instance, groundwater contamination problems are commonly addressed by pumping water from a network of extraction wells located within a contaminant plume [5]. The success of this type of remediation strategy is highly dependent on the effective positioning of extraction wells in the field of contamination. This problem can be translated to an area

coverage control problem. In this scenario, the risk density can be defined as the density of the contamination; then a WSN can be deployed within the contaminated field to find the optimal configuration of the extraction wells.

The static area coverage control with stationary risk density function has been widely studied. Some studies formulated the coverage control problem with the time-varying risk density. In those studies, the time-varying risk density was modelled by a given time-varying function or by a diffusive evolving environment. The convection was not taken into account.

In the practical scenarios, many phenomena can be modelled by considering a convective–diffusive environment. For instance, in the water contamination monitoring case, a WSN can be deployed to detect pollutants. The mass transfer of the pollutant is caused by its own diffusion process and by convection in the water area. It is natural to describe the evolution of the risk distribution using a convection-diffusion equation [6]. A variety of other applications may exist in convective-diffusive environment. This thesis is motivated by previous research of optimizing the non-autonomous coverage control in time-varying environments and potential applications of coverage control. The main contribution of this work is to extend the analysis to a more general case in which the changing environment is defined by a time-varying density function which is governed by a convection-diffusion equation, and to illustrate some implication through simulations.

1.2 Thesis Objectives and Contributions

This work contributes to the research on area coverage control algorithms with multi-agent systems. The goals of this thesis include the formulation of a type of area coverage problem in which a multi-agent system is deployed in a convective-diffusive environment, and the designing of the coverage strategy based on the Voronoi partition. The area coverage control in this work is considered as an optimal control problem. Since the environment (density function) is time-varying, asymptotic equilibriums of the system are trajectories rather than fixed points in the domain. By choosing a suitable non-autonomous feedback law, the coverage metric can reach the local maximum when agents move along the trajectories. The asymptotic stability of the multi-agent system is proven by using Barbalat's lemma, and several simulations illustrate the convergence of the feedback control law in different scenarios.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter. 2, the literature pertaining fundamental concepts and applications of WSNs is reviewd and some coverage control algorithms based on Voronoi partition are discussed. In Chapter. 3, the area coverage control problem is formulated with a time-varying environment evolving according to a diffusive-convective field model. An area coverage strategy based on the classical Voronoi partition which can maximize the coverage metric is presented firstly. Then a different area coverage strategy based on order 2 Voronoi partition is developed, which explores how to deploy the multi-agent system optimally if the detection area of each agent are allowed to overlap. The numerical simulation of several scenarios is presented in this chapter. Finally, Chapter. 4 summarizes and concludes the findings of this study.

Chapter 2

Background and Literature Review

This chapter presents an overview of some of vast literature on WSNs and coverage control algorithms with multi-agent systems. Various technologies and applications of WSNs are reviewed.

The literature review is divided into three parts. Section 2.1 briefly introduces the definition of sensor node and WSN. Section 2.2 provides some details of applications of WSNs; this section also discusses the challenges and issues of WSNs. Section 2.3 includes an introduction to area coverage control and coverage control algorithms; some important concepts, such as Voronoi diagram and Lloyd's algorithm are introduced. Various algorithms to optimize the configuration of the coverage control model are also presented in this section.

2.1 Sensor Node and Wireless Sensor Networks

An object performing a sensing task is called a sensor or sensor node; when a group of sensor nodes cooperatively monitor large physical environments, they form a WSN [7]. Each sensor node within wireless sensor network is capable of sensing data and sharing data with connected sensor nodes [8]. To achieve this goal, new capabilities, such as onboard processing, communication, and storage capabilities are added to conventional sensors. With those enhancements, WSNs can not only monitor the physical phenomenon, but also assume responsibility for data processing and self-management [9].

Dargie and Poellabauer introduce the definition and background of sensor nodes and WSNs in [10]. A detailed discussion of protocols, algorithms, and technologies of a WSN are presented. Moreover, several additional techniques and solutions for

a variety of challenges and constraints are proposed. Some of the challenges often associated with WSNs include [11]:

1. Resource constraints; the implementation of WSNs is constrained by three types of resources: energy resource, memory resource, and processing resource.

2. Dynamic topologies and harsh environmental conditions; the connectivity and the topology of the underlying network may change due to link and node failures.

3. Radio bandwidth constraints; limited communication capacity, packet loss, error, and lag may occur, depending on the radio bandwidth. Unfortunately, the data redundancy caused by high density in the network topology greatly intensifies the problem.

Recent research in WSNs by Chong and Kumar is discussed in [12], including two important programs of the Defence Advanced Research Projects Agency: the "Distributed Sensor Networks" and "The Sensor Information Technology" programs. They also provide details on technology trends, which impact the development of WSNs. Moreover, details about new applications of WSNs such as infrastructure security and habitat monitoring are presented.

2.2 Some Applications of Wireless Sensor Networks

The capabilities of WSNs can vary widely depending on the hardware architecture. In [13] and [1], authors reduce the various applications to two different categories: tracking and monitoring, as shown in Figure 2.1.

A survey of applications of WSNs can be found in [14], where they are generally classified into military applications [15] [16] [17] [18], environmental monitoring [19] [20], and commercial applications [21] [22]. The concept of WSNs is well developed in the military field, where enemy tracking and target classification are typical areas of interest. Battlefield surveillance is also of interest, since prevention of intrusion is the requirement of the defense system. Missions usually involve high risk for soldiers and, therefore, using WSNs is of great practical importance [23].

Another critical application of WSNs is environmental monitoring. Mainwaring and Polastre proposed a study of designing of habitat monitoring in [24]. The

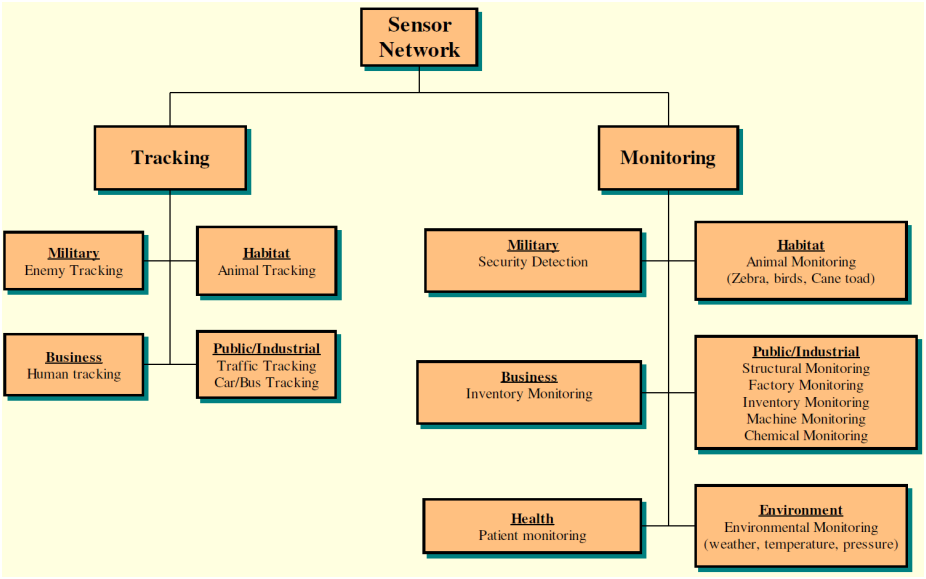


Figure 2.1: An overview of WSN applications [13].

study covers hardware designing, network designing, remote accessing, and data management. In their project on Great Duck Island, a WSN was deployed which consisted of 32 nodes. They aim to build a WSN which can streams the data live to the Internet. The system architecture is illustrated in Figure. 2.2.

WSNs also have applications for multiple commercial purposes. Research on how the health science and health care systems benefit from WSNs is demonstrated in [25]. Electric energy systems also benefit from deploying wireless sensor networks to households. The efficiency of energy generation and distribution is significantly improved by applying a WSN to information-technology system [26].

In recent years, application of WSNs in robotics have rapidly become popular. One important category of applications is area coverage control. Gage defines the notion of sensor coverage, where the objective is to achieve a static arrangement of sensors that maximizes the total detection area [27]. In coverage control, the total detection area can be quantified by the coverage metric.

2.3 Models and Algorithms of Coverage Control

2.3.1 Voronoi Partition

Named after Georgy Voronoi, the Voronoi partition (or Voronoi diagram) is one of the most crtitcal structures in computational geometry. Given several sites in a two-dimensional plane, the Voronoi partition can divide the plane into several re-

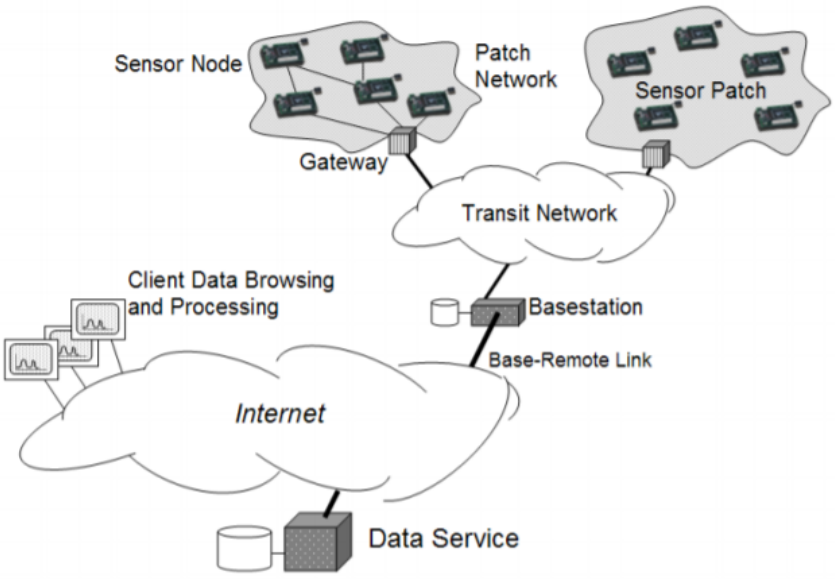


Figure 2.2: Overview of habitat monitoring system [24].

gions according to the nearest-neighbor rule [28]: for each site (referred to as seed or generator in computational geometry), there is a corresponding region consisting of all points which are nearest to this generator than to any others. These regions are called Voronoi cells.

Formal definition: Let $\mathbb{R}^d$ be a metric space with distance function d. Let K be a set of indices and let$(\mathbf{p}_k)_{k\in K}$ be a sequence of of non-empty subsets in the space $\mathbb{R}^d$. $\mathcal{V}_i$ is the Voronoi cell associated with the generator $\mathbf{p}_i$, which consists of all the points in X whose distance to $\mathbf{p}_i$ is less than or equal to their distance to other generators $\mathbf{p}_k$, where k is any index different from i. Let $d(x, y)$ be the distance between point x and point y, then the $k-$th Voronoi cell is defined by:

$$\mathcal{V}_i = \{\mathbf{q} \in \mathbb{R}^d \mid d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_k),\ \forall i \neq k\} \tag{2.1}$$

The Voronoi partition is the subdivision of the domain into cells, one for each site [29]. Depending on the distance used to compute them, Voronoi cells may not overlap. By definition, infinite seeds are permitted; however, typically finite seeds are the sole case gaining consideration.

The Euclidean space is perhaps the most popular case in this spectrum. Finite sites $\{\mathbf{p}_1, \cdots, \mathbf{p}_n\}$ are given and each site in the set is a seed. In this case, each

Voronoi cell is a convex polytope.

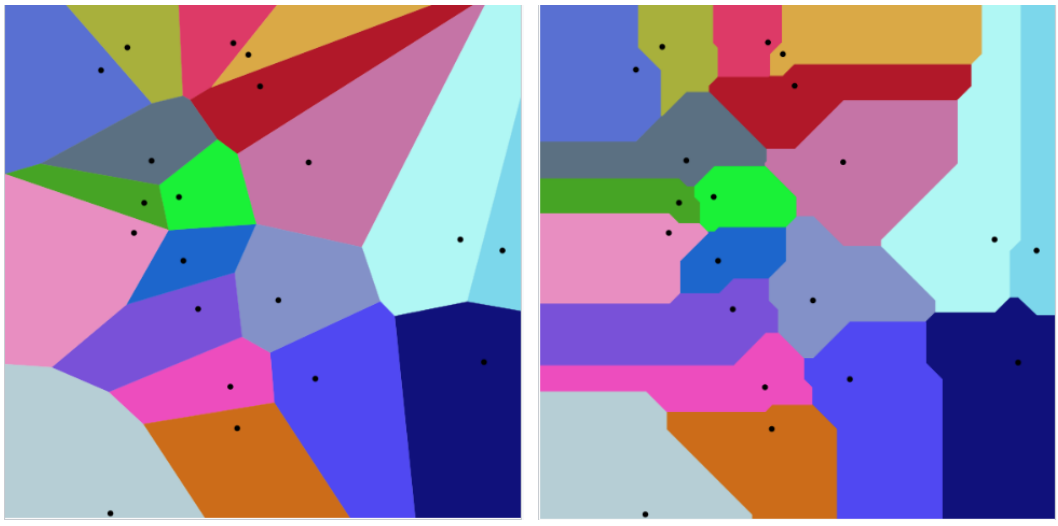
Illustration [30]: A basic illustration is presented below. Consider a group of shopping centers in a city, wherein one has been tasked with estimating the number of potential customers going through each shopping center. Assume that customers choose their preferred shopping center by proximity; namely, they choose the shopping center which is nearest to them. In this case, Voronoi partition can be used to get Voronoi cell $\mathcal{V}_k$ for each seed $\mathbf{p}_k$. The distance between two seeds $\mathbf{p}_1(x_1, y_1)$, $\mathbf{p}_2(x_2, y_2)$ can be measured by Euclidean distance:

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

The Voronoi partition undertakes different forms in accordance with the varying distance function it is associated with. In accordance with the aforementioned case, replacing Euclidean distance with Manhattan distance would gives

$$d(\mathbf{p}_1, \mathbf{p}_2) = \mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid$$

then another Voronoi partition is generated, see Figure 2.3.



(a) The Voronoi partition based on Euclidean distance.

(b) The Voronoi partition based on Manhattan distance.

Figure 2.3: The Voronoi partition of 20 seeds.

2.3.2 Higher Order Voronoi diagram

Higher-order Voronoi diagrams are essentially generalized Voronoi diagrams. The order k Voronoi diagram is a partition of domain into subregions such that any point within a subregion has the same k nearest sites [31]. The definition of a order k Voronoi diagrams is discussed in [32].

Formal Definition: Let $\mathcal{S}$ be a set of n sites in d-dimensional space $\mathbb{R}^d$, which is given by

$$\mathcal{S} = \{\mathbf{p}_1,\ \mathbf{p}_2,\ \cdots,\ \mathbf{p}_n\} \tag{2.2}$$

and $\mathcal{T}$ is a subset of k elements in $\mathcal{S}$. The Voronoi partition of order k is defined as the collection of following cells:

$$\mathcal{V}_{\mathcal{T}_\ell} = \{\mathbf{q} \in \mathbb{R}^d \mid d(\mathbf{q}, v) \leq d(\mathbf{q}, w),\ \forall v \in \mathcal{T}_\ell, \forall w \in \mathcal{S} \setminus \mathcal{T}_\ell, |\mathcal{T}_\ell| = k\} \tag{2.3}$$

where $S \setminus \mathcal{T}$ is the complementary set of $\mathcal{T}$, with respect to S.

These cells are convex polyhedra that form a cell complex in domain; in the case $k = 1$, the classical Voronoi diagram is obtained [28]. Additional properties of the higher-order Voronoi diagram are discussed in [33] and [34]. Algorithms to generate higher-order Voronoi diagram have been widely employed; some mature and simple algorithms are summarized in [35].

An algorithm to construct order 2 Voronoi partition is presented in the following. The order 2 Voronoi partition can be generated based on order 1 Voronoi partition. Assuming ℓ_1 and ℓ_2 are two elements in $\mathcal{T}_\ell$. $\mathcal{V}_{\ell_1}$ and $\mathcal{V}_{\ell_2}$ are the Voronoi cells for ℓ_1 and ℓ_2 in the classical Voronoi partition. Furthermore, $\mathcal{V}_{\mathcal{T}_\ell}$ can be divided into two subset, namely $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_1}$ and $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_2}$, where $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_1}$ consists of all points in $\mathcal{V}_{\mathcal{T}_\ell}$ closer to site ℓ_1 than site ℓ_2, whereas in $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_2}$ the opposite is the case. According to the definition of Voronoi partition, $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_1}$ is exactly the intersection of $\mathcal{V}_{\ell_1}$ and $\mathcal{V}_{\mathcal{T}_\ell}$:

$$\mathcal{V}_{\mathcal{T}_\ell} \cap \mathcal{V}_{\ell_1} = \mathcal{V}_{\mathcal{T}_\ell}^{\ell_1} \tag{2.4}$$

If site ℓ_1 is eliminated, site ℓ_2 will be the nearest site to all points in $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_1}$. Let $\mathcal{V}_{\ell_2}^{\mathcal{S}\setminus\ell_1}$ be the classical Voronoi cell related to site ℓ_1 after site ℓ_2 was eliminated, we have

$$\mathcal{V}_{\ell_2}^{\mathcal{S}\setminus\ell_1} \cap \mathcal{V}_{\ell_1} = \mathcal{V}_{\mathcal{T}_\ell}^{\ell_1} \tag{2.5}$$

Similarly, let $\mathcal{V}_{\ell_1}^{\mathcal{S}\setminus\ell_2}$ be the Voronoi cell related to site ℓ_1 after site ℓ_2 was eliminated.

Then we have

$$\mathcal{V}_{\ell_1}^{\mathcal{S}\setminus\ell_2} \cap \mathcal{V}_{\ell_2} = \mathcal{V}_{\mathcal{T}_\ell}^{\ell_2} \tag{2.6}$$

As shown in Algorithm 1, the cell $\mathcal{V}_{\mathcal{T}}$ is calculated through the following operations:

$$\mathcal{V}_{\mathcal{T}_\ell} = \mathcal{V}_{\mathcal{T}_\ell}^{\ell_2} \cup \mathcal{V}_{\mathcal{T}_\ell}^{\ell_1} \tag{2.7}$$

Algorithm 1 Order 2 Voronoi partition [36]

Input: n sites' positions $\mathcal{S}$
Output: Order 2 Voronoi partition

function ORDER2VORONOIPARTITION($\mathcal{S}$)
  $\ell \leftarrow 0$
  for $\mathcal{T}_\ell$ in $\mathcal{S}$ **do**
    $\ell \leftarrow \ell + 1$
    $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_1} \leftarrow \mathcal{V}_{\ell_2}^{\mathcal{S}\setminus\ell_2} \cap \mathcal{V}_{\ell_1}$
    $\mathcal{V}_{\mathcal{T}_\ell}^{\ell_2} \leftarrow \mathcal{V}_{\ell_1}^{\mathcal{S}\setminus\ell_2} \cap \mathcal{V}_{\ell_2}$
    $\mathcal{V}_{\mathcal{T}_\ell} \leftarrow \mathcal{V}_{\mathcal{T}_\ell}^{\ell_2} \cup \mathcal{V}_{\mathcal{T}}^{\ell_1}$
    $VoronoiCells[\ell] \leftarrow \mathcal{V}_{\mathcal{T}_\ell}$
  end for
  return VoronoiCells
end function

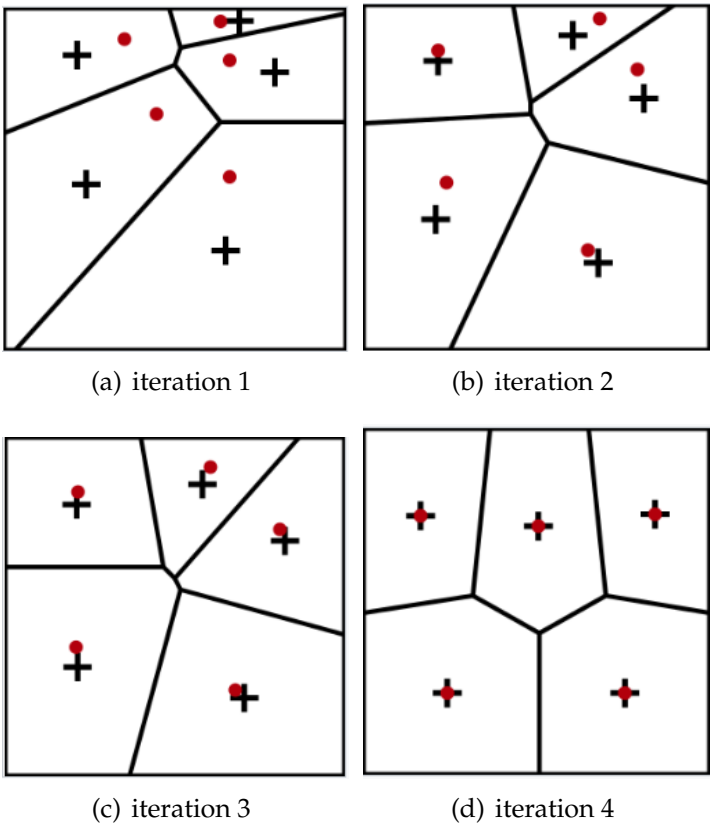
2.3.3 Lloyd's Algorithm

Centroidal Voronoi partition is ultimately a subset of Voronoi partitions. A Voronoi diagram is deemed centroidal when the seed of each Voronoi cell is also its centroid. Centroidal Voronoi diagrams enjoy an optimization characterization so that they turn out to be useful in diverse applications such as optimal placement of sensors and actuators for control [37]. Lloyd's algorithm can be used to generate centroidal Voronoi tessellation. The description of Lloyd's algorithm is presented in following.

Algorithm description: Lloyd's algorithm starts with an initial placement of n seeds in the domain, then repeatedly executes the following steps [38]:

- For $k = 1,\ 2 \cdots,\ n$, the Voronoi cell R_k of each seed P_k is computed.
- Each Voronoi cell is integrated and the centroid is computed.

- Each seed is moved to the centroid of its Voronoi cell.



(a) iteration 1 (b) iteration 2

(c) iteration 3 (d) iteration 4

Figure 2.4: Lloyd's algorithm.

An example of Lloyd's algorithm is shown in Figure 2.4. Once each relaxation step is performed, the seed may disperse in a slightly more even distribution; since closely spaced seeds move farther from one another, and widely spaced seeds move closer together. In one dimension, Lloyd's algorithm has been shown to converge with the centroidal Voronoi diagram. In higher dimensions, some slightly weaker convergence results are known. Through the applications of Lloyd's algorithm in the real world, the iteration usually stops when the distribution meets the expected goals. For instance, a termination criterion commonly used is to stop the iteration when the maximum distance moved by any seeds falls below a critical value. Xiao Xiao presents a method to accelerate the convergence by applying over-relaxation schemes in the iteration process [39]. The numerical experiment shows faster convergence of the proposed over-relaxtion Lloyd method.

2.3.4 Coverage Control with Time-Invariant Density Function

Cortes and Martinez present a control coordination algorithm for mobile agent systems. They focus on gradient descent algorithms based on Lloyd's algorithm for a class of utility functions, which generate optimal coverage and sensing policies. [40].

Consider n mobile sensing nodes in a bounded workspace Ω ($\Omega \subset \mathbb{R}^N$), and the i-*th* sensor node location is given by $\mathbf{p}_i$ ($\mathbf{p}_i \in \Omega$). Then $\mathbf{P}$ ($\mathbf{P} = \{\mathbf{p}_1^{\mathrm{T}}, \cdots, \mathbf{p}_n^{\mathrm{T}}\}$) can be considered as the configuration of the multi-agent system. The definition of coverage metric is given by

$$\mathcal{H} = \int_{\Omega} \max_{i \in \{1,\cdots,n\}} f\big(d(\mathbf{q}, \mathbf{p}_i)\big)\phi(\mathbf{q})\mathrm{d}\Omega \tag{2.8}$$

where $\mathrm{d} : \mathbb{R}^N \to \mathbb{R}$ is the distance function defined by the distance between point $\mathbf{q} \in \Omega$ and seed $\mathbf{p}_i$. The function ϕ represents the risk density, which provides a weight to each point in Ω [41]. The function f defines the sensing performance of the agent with respect to distance. In our definition, it is a continuously differentiable, strictly decreasing function over the range d. In coverage control, the density function is not only be considered as the probability of the occurrence of events in different regions, but also represents the measures of relative importance of different regions in Ω. As per general expectations, points with higher densities should typically be covered better than points with lower densities. The area coverage problem becomes an optimization to maximize the coverage metric, as follows:

$$\max_{\mathbf{P}} \mathcal{H}(\mathbf{P}) = \max_{\mathbf{P}} \int \max_{i} f\big(\mathrm{d}(\mathbf{q}, \mathbf{p}_i)\big)\phi(\mathbf{q})\mathrm{d}\Omega$$

The definition of Voronoi cell gives the following:

$$\mathcal{H}(\mathbf{P}) = \sum_{i=1}^{n} \mathcal{H}(\mathbf{p}_i) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} f(r_i)\phi(\mathbf{q})\mathrm{d}\Omega \tag{2.9}$$

$$\max_{\mathbf{P}} \ \mathcal{H}(\mathbf{P}) = \max_{\mathbf{P}} \sum_{i=1}^{n} \int_{\mathcal{V}_i} f(r_i)\phi(\mathbf{q})\mathrm{d}\Omega \tag{2.10}$$

where r_i represent the distance between $\mathbf{p}_i$ and $\mathbf{q}$. $\mathcal{V}_i$ is defined as:

$$\mathcal{V}_i(\mathbf{p}) = \{\mathbf{q} \in \Omega \mid \ f(r_i) \leq f(r_j),\ \forall j \neq i\} \tag{2.11}$$

(2.11) is equivalent to:

$$\mathcal{V}_i(\mathbf{p}) = \{\mathbf{q} \in \Omega \mid \ r_i \leq r_j, \ \forall j \neq i\} \tag{2.12}$$

because function $f(r)$ is a strictly decreasing function.

Notations relating to Voronoi cells are introduced. Let $\partial\mathcal{V}_i$ and $\partial\Omega$ represent the boundary of $\mathcal{V}_i$ and Ω respectively. $\mathbf{q}_{\partial\mathcal{V}_i}$ is a point on $\partial\mathcal{V}_i$. Let $\mathbf{n}_{\partial\mathcal{V}_i}$ be an outward unit normal of $\partial\mathcal{V}_i$. The index set $\mathcal{N}_i$ which shared Voronoi boundaries with $\mathcal{V}_i$ is defined as:

$$\mathcal{N}_i = \{j \mid \mathcal{V}_i \bigcap \mathcal{V}_j \neq \emptyset\}$$

The set $l_{ij} = \mathcal{V}_i \cap \mathcal{V}_j$ consist of all points on the shared boundary of i*th* and j*th* agent. $\mathbf{q}_{l_{ij}}$ is a point on that shared boundary, and $\mathbf{n}_{l_{ij}}$ is a unit normal of l_{ij} from $\mathbf{p}_i$ to $\mathbf{p}_j$. In accordance with the definition of Voronoi partition, we have the following:

$$\partial\mathcal{V}_i = (\cup_{j\in\mathcal{N}_i} l_{ij}) \cup (\partial\mathcal{V}_i \cap \partial Q) \tag{2.13a}$$

$$l_{ij} = l_{ji} \tag{2.13b}$$

$$\mathbf{n}_{l_{ij}} = -\mathbf{n}_{l_{ji}} \tag{2.13c}$$

In 2-D Euclidean space, the shared boundaries are segments or rays, and Voronoi cells are convex polygons. The following lemma provides considerations for optimizing the coverage metric.

Lemma 1 (Distributed Gradient)**.**

$$\frac{\partial\mathcal{H}}{\partial\mathbf{p}_i} = \int_{\mathcal{V}_i} \frac{\partial}{\partial\mathbf{p}_i} f(r_i)\phi(\mathbf{q})\mathrm{d}\Omega \tag{2.14}$$

Proof.

$$\begin{aligned}\frac{\partial\mathcal{H}}{\partial\mathbf{p}_i} = &\int_{\mathcal{V}_i} \frac{\partial}{\partial\mathbf{p}_i} f(r_i)\phi(\mathbf{q})\mathrm{d}\Omega \\ &+ \int_{\partial\mathcal{V}_i\cap\partial\Omega} f(r_i)\phi(\mathbf{q})\frac{\partial\mathbf{q}_{\partial\mathcal{V}_\mathbf{i}}}{\partial\mathbf{p}_i}\mathrm{d}\Omega \\ &+ \sum_{j\in\mathcal{N}_i}\int_{l_{ij}} \Big(f(r_i) - f(r_j)\Big)\phi(\mathbf{q})\frac{\partial\mathbf{q}_{l_{ij}}}{\partial\mathbf{p}_i}\mathbf{n}_{l_{ij}}\mathrm{d}\Omega\end{aligned}$$

According to the definition of the Voronoi partition, we have:

$$r_i = r_j,\ \forall \mathbf{q} \in l_{ij}$$
$$\frac{\partial \mathbf{q}_{\partial \mathcal{V}_i}}{\partial \mathbf{p}_i} = 0,\ \forall \mathbf{q} \in \partial \mathcal{V}_i \cap \partial \Omega$$

then the second and third term vanishes [1]. □

This lemma provides a method to calculate the gradient of $\mathcal{H}$ with respect to $\mathbf{p}_i$. If the workspace is a 2-D planer and the distance is Euclidean, the generalized mass, the generalized centroid and the generalized density function are defined as follows:

$$\begin{aligned}
\tilde{\phi}_i(\mathbf{q}, t) &= -2\phi(\mathbf{q}, t)\frac{\partial f(r_i)}{\partial (r_i^2)} \\
\tilde{m}(\mathcal{V}_i(t)) &= \int_{\mathcal{V}_i} \tilde{\phi}_i(\mathbf{q}, t)\mathrm{d}\Omega \\
\tilde{\mathbf{c}}(\mathcal{V}_i(t)) &= \frac{1}{\tilde{m}(\mathcal{V}_i(t))}\int_{\mathcal{V}_i} \mathbf{q}\tilde{\phi}_i(\mathbf{q}, t)\mathrm{d}\Omega
\end{aligned} \tag{2.15}$$

where $f(r_i)$ is the sensing performance of i-*th* agent. By using these definitions, the expression of $\frac{\partial \mathcal{H}}{\partial t}$ can be written in the form [42]:

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \tilde{m}(\mathcal{V}_i(t))(\tilde{\mathbf{c}}(\mathcal{V}_i(t)) - \mathbf{p}_i) \tag{2.16}$$

A centroidal diagram can be considered as an optimal configuration provides that $\phi(\mathbf{q})$ is a time-invariant density function; in the sense that it maximizes the coverage metric. Consider a group of n mobile agents, and motion of the i-*th* mobile agent is described by:

$$\dot{\mathbf{p}}_i = \mathbf{u}_i \tag{2.17}$$

where $\mathbf{u}_i(t)$ is the velocity of i-*th* agent at time t. The evolution of this n agents system can be defined by

$$\dot{\mathbf{p}}(t) = \mathbf{u}(t) \tag{2.18a}$$
$$\mathbf{p}(0) = \mathbf{p}_0 \tag{2.18b}$$

where $\mathbf{u}(t) = (\mathbf{u}_1,\ \mathbf{u}_2,\ \cdots,\ \mathbf{u}_n)^{\mathrm{T}} \in \mathbb{R}^{2n}$. Assuming a time-invarient density $\phi(\mathbf{q})$ is given, the Lloyd's algorithm is established by the following feedback law, which

generates the optimal trajectories and maximize the coverage metric [1].

$$\mathbf{u}_i = \kappa \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \kappa \tilde{m}(\mathcal{V}_i)(\tilde{\mathbf{c}}(\mathcal{V}_i) - \mathbf{p}_i),\ \kappa > 0 \tag{2.19}$$

2.3.5 Coverage Control with Time-Varying density function

In many applications of mobile agent systems the risk density function is time-varying, as for example when a group of mobile agents is deployed in a evolving environment, or in surveillance problems in which hostile agents may enter a perimeter and move in it. Lee and Egerstedt presented a control framework for a multi-agent system by choosing with the following time-varying density function [43]:

$$\phi(\mathbf{q}, t) = e^{-(\mathbf{q}_x - 2sin\frac{t}{\tau})^2 - \mathbf{q}_y^2} \tag{2.20}$$

They designed a non-autonomous control algorithm that provides optimal coverage with respect to the density (2.20), and demonstrated the convergence and performance by implementing it on a hardware.

A more general case is presented in subsequent work by Lee and Egerstedt [44]; wherein the main idea of which is to combine a proportional term with a controller, to track the time-varying evolution, where the following a non-autonomous feedback law is shown to track time-varying centroids provided the special initial condition in which the agents' locations coincide with the centroids:

$$\dot{\mathbf{p}} = (\mathbf{I} + \frac{\partial \mathbf{c}}{\partial t})(-\kappa(\mathbf{p} - \mathbf{c}) + \frac{\partial \mathbf{c}}{\partial t}) \tag{2.21}$$

where κ is a positive gain and $\mathbf{c}$ is the centroid of the Voronoi cell.

Miah et al. advanced the study of non-autonomous coverage control with evolving density. They consider two cases, in which the time evolving density is related to a set of external threats/targets entering the domain, and to a time evolving environment in which a substance enters the domain and evolves according to a diffusive law, described a by a partial differential equation [42] [45]. They proved that the following non-autonomous feedback law asymptotically tracks time varying centroids without any additional assumption on the initial conditions, therefore handling also the transient:

$$\mathbf{u}_i(t) = \frac{\partial \mathcal{H}/\partial \mathbf{p}_i}{\|\partial \mathcal{H}/\partial \mathbf{p}_i\|^2} \left(\kappa \|\tilde{\mathbf{c}}_i - \mathbf{p}_i\|^2 - \frac{\partial \mathcal{H}}{\partial t} \right) \tag{2.22}$$

where $\mathcal{H}$ is the coverage metric and $\tilde{\mathbf{c}}_i$ is the generalized centroid of the i-*th* Voronoi cell.

The main contribution of this work is to consider a diffusive-convective environment, modeling several important cases of area coverage in evolving environment in which there is a diffusive substance injected in a domain of interest, with a transport component due for example to an underlying vector field, which in the most obvious case could be the velocity field of a fluid in which the operation of the robotic system occurs. It is shown that the feedback law (2.22) generally applies when accounting for the specific coverage metric induced by the density function considered here. Theoretically predicted convergence and tracking are illustrated in simulation.

Chapter 3

Area Coverage in a Diffusive-Convective Environment

3.1 Problem statement

In our increasingly industrialized world, events such as oil spills with diffusion in the environments and chemical leakages can occur frequently because of the expanding offshore development activities and terrestrial industrial plants [46]. The Fukushima Daiichi nuclear power plant accident in 2011, triggered by an earthquake and followed by a tsunami, caused the month-long discharge of radioactive materials into the sea [47]. The Mexico Gulf oil spill in 2010 is recognized as a major disaster in US history [48]. The Gulf War oil spill resulting from the Gulf War in 1991 was also one of the largest oil spills in history. In case of marine pollution caused by leakage of diffusive substances, it is important to detect sources and give predictions about their evolution and trajectory [49]. Oil is usually insoluble in water. However, complex physical and chemical changes may occur after petroleum and its products entering the ocean and other waters, such as diffusion, evaporation, dissolution, emulsification and photochemical oxidation, which makes it diffusing in seawater [50]. By deploying a multi-agent system, the attempt is to respond precisely, quickly and coordinately to minimize the environmental impact when a marine pollution event is produced [51]. A marine pollution event can be modeled as following. Consider a 2-D domain Ω with a convective component, which could be the velocity field of the water in Ω. Let $\phi(\mathbf{q}, t)$ denote the density of diffusive substances. We can model the mass transport of the diffusive substance by using the continuity equation, which is the differential form of the law

of conservation of ϕ in Ω:

$$\frac{\partial \phi}{\partial t} + \text{div}\mathbf{\Phi} = 0 \tag{3.1}$$

where $\mathbf{\Phi}$ is the total flux of the diffusive substance, and div is the divergence operator. The flux due to the diffusion $\mathbf{\Phi}_{\text{diff}}$ is assumed to be described by Fick's first law, which is the linear constitutive equation stating that the flux due to diffusion is proportional to the concentration gradient:

$$\mathbf{\Phi}_{\text{diff}} = -D\nabla\phi \tag{3.2}$$

where D is the diffusive coefficient of diffusive substance, and ∇ is the spatial gradient operator. When overall convection or flow exists in this domain, there is an associated flux called advective flux:

$$\mathbf{\Phi}_{\text{adv}} = \mathbf{U}\phi \tag{3.3}$$

where $\mathbf{U}$ is the flow velocity field. The total flux is given by

$$\mathbf{\Phi} = \mathbf{\Phi}_{\text{diff}} + \mathbf{\Phi}_{\text{adv}} \tag{3.4}$$

Therefore, the evolution of $\phi(\mathbf{q}, t)$ is described by the convection-diffusion equation:

$$\frac{\partial \phi}{\partial t} + \text{div}(-D\nabla\phi + \mathbf{U}\phi) = 0 \tag{3.5}$$

Assume n mobile agents are deployed inside the domain with the mission of monitoring the diffusive substance. In this work, we consider the simplest kinematic model, where the motion of i*th* agent is described by

$$\dot{\mathbf{p}}_i = \mathbf{u}_i \tag{3.6}$$

under the assumption of a fully actuated, kinematic model with no constraints in turning rates and no additional degrees of freedom other than the two translations associated to tht eplanar motion of a mass particle. Therefore, the agents are assumed to be able to move with any given velocity, and to not be affected by the motion of the environmental field in which they are embedded.

The sensing performance of each agent with respect to the substance is assumed to strictly decrease with distance from the agent's position. Therefore the sensing performance function $f(r_i)$ should be a strictly decreasing function, where

$r_i = \|\mathbf{p}_i - \mathbf{q}\|$ is the Euclidean distance between the ith agent's position $\mathbf{p}_i$ and point $\mathbf{q}$. For the purpose of theoretically formulate this problem, we also make the assumption that $f(r_i)$ is continuously differentiable in Ω. In this chapter, two coverage strategies based on classical Voronoi partition and order 2 Voronoi partition are presented respectively.

3.2 Area Coverage with Order 1 Voronoi Tessellations

To achieve the optimal deployment of a group of mobile agents, the classical Voronoi partition (order 1 Voronoi partition) is used to divide the domain into n subdomains, wherein each subdomain does not overlap. Each subdomain is assigned to an agent. Furthermore, the coverage metric is defined by

$$\mathcal{H} = \sum_{i=1}^{n} \int_{\mathcal{V}_i} \phi(\mathbf{q}, t) f(r_i) \mathrm{d}\Omega \tag{3.7}$$

where $\mathcal{V}_i$ is the i-*th* Voronoi cell [52]. The coverage metric, which encodes the sensing performance and the time-varying density function, is used to evaluate the performance of the multi-agent system. Additional agents should be deployed in instances where the density of the substance is higher. Mathematically, the goal is to optimize the configuration of the multi-agent system, to thereon maximize the coverage metric accordingly.

In this section, we present an area coverage strategy based on the classical Voronoi partition whose generated trajectories converge to a configuration that maximizes the coverage metric defined by (3.7) for the general diffusive-convective environment.

3.2.1 Asymptotic Stability of Feedback law

In this section we prove that the non-autonomous feedback law proposed in [42] stabilizes the trajectories of groups of agents navigating time varying environments described by diffusive-convective evolving densities. This generalizes previous results established for purely diffusive environnments. The generalized density, generalized mass and generalized centroid are defined by

$$\tilde{\phi}_i(\mathbf{p}_i, \mathbf{q}, t) = -2\frac{\partial f(r_i)}{\partial r_i^2}\phi(\mathbf{q}, t) \tag{3.8a}$$

$$\tilde{m}(\mathcal{V}_i) = \int_{\mathcal{V}_i} \tilde{\phi}(\mathbf{p}_i, \mathbf{q}, t) \mathrm{d}\Omega \tag{3.8b}$$

$$\tilde{\mathbf{c}}(\mathcal{V}_i) = \frac{1}{\tilde{m}(\mathcal{V}_i)} \int_{\mathcal{V}_i} \mathbf{q}\tilde{\phi}(\mathbf{p}_i, \mathbf{q}, t) \mathrm{d}\Omega \tag{3.8c}$$

and the non-autonomous feedback control law is given by

$$\mathbf{u}_i(t) = \frac{\partial \mathcal{H}/\partial \mathbf{p}_i}{\|\partial \mathcal{H}/\partial \mathbf{p}_i\|^2} \left(\kappa \|\tilde{\mathbf{c}}(\mathcal{V}_i) - \mathbf{p}_i\|^2 - \frac{\partial \mathcal{H}_i}{\partial t} \right) \tag{3.9}$$

where $\mathcal{H}$ is the coverage metric defined in (3.7). For every smooth density ϕ, this feedback control law asymptotically maximizes the non-autonomous coverage metric [42], therefore offering a general solution that applies to the class of non-autonomous area coverage control problems with single integrator kinematic agents and sufficiently regular risk densities ϕ. When a time-invariant density function is given, this control law reduce to the Lloyd's algorithm. Once it is shown that the feedback law maximizes the non-autonomous coverage metric, we still need to proof that the agents' trajectories asymptotically stabilize to bounded trajectories in Ω. This will be established by using Barbalat's lemma to show that the trajectories asymptotically converge to time varying centroids of a Voronoi partition of Ω.

The partial derivative $\frac{\partial \mathcal{H}}{\partial t}$ depends on the diffusion coefficient D and the velocity $\mathbf{U}$, which account for the time-varying environment. This term is needed for the computation of (3.9). We first present a Lemma which states that the partial derivative $\frac{\partial \mathcal{H}}{\partial t}$ is finite. This Lemma will be an important building block to prove the asymptotic stability of the system in the following.

Lemma 2. *Suppose that the time-varying density $\phi(\mathbf{q}, t)$ evolves according to the convection-diffusion equation defined by (3.5), with coverage metric is defined by (3.7). Then the term $\frac{\partial \mathcal{H}}{\partial t}$ is bounded if the flux Φ across the boundary $\partial\Omega$ and the velocity field $\mathbf{U}$ are finite.*

Proof. The partial time derivative of coverage metric is given by

$$\frac{\partial \mathcal{H}(\mathbf{q}, t)}{\partial t} = \sum_{i=1}^{n} \int_{\mathcal{V}_i} f(r_i) \frac{\partial \phi(\mathbf{q}, t)}{\partial t} \mathrm{d}\Omega \tag{3.10}$$

Substitution from (3.5) into (3.10) gives

$$\frac{\partial \mathcal{H}}{\partial t} = -\sum_{i=1}^{n} \int_{\mathcal{V}_i} f(r_i) \mathrm{div}\left(-D\nabla\phi + \mathbf{U}\phi\right) \mathrm{d}\Omega \tag{3.11}$$

By using the following vector calculus identities:

$$f(r_i)\text{div}(-D\nabla\phi(\mathbf{q},t)) = \text{div}(-Df(r_i)\nabla\phi(\mathbf{q},t)) + D\nabla\phi(\mathbf{q},t)\cdot\nabla f(r_i) \tag{3.12a}$$

$$f(r_i)\text{div}(\mathbf{U}\phi(\mathbf{q},t)) = \text{div}(f(r_i)\mathbf{U}\phi(\mathbf{q},t)) - \phi(\mathbf{q},t)\mathbf{U}\cdot\nabla f(r_i) \tag{3.12b}$$

the spatial integral becomes

$$\frac{\partial\mathcal{H}}{\partial t} = \sum_{i=1}^{n}\int_{\mathcal{V}_i}(D\nabla\phi - \mathbf{U}\phi)\cdot\nabla f + \text{div}\left((-D\nabla\phi + \phi\mathbf{U})f\right)\text{d}\Omega \tag{3.13}$$

By using the divergence theorem we obtain

$$\sum_{i=1}^{n}\int_{\mathcal{V}_i}\text{div}\left(f(-D\nabla\phi + \mathbf{U}\phi)\right)\text{d}\Omega = \int_{\partial\Omega} f(-D\nabla\phi + \mathbf{U}\phi)\text{d}\ell \tag{3.14}$$

where $\text{d}\ell$ is an element of the boundary $\partial\Omega$. As usual in boundary values problems, the boundary is partitioned into two non overlapping portions $\partial\Omega_\phi$ and $\partial\Omega_\Phi$, respectively where the density ϕ is prescribed, and where the normal derivative of the flux is prescribed:

$$\phi(\mathbf{q},t) = \bar{\phi}(\mathbf{q},t) \text{ on } \partial\Omega_\phi \tag{3.15a}$$

$$(-D\nabla\phi + \mathbf{U}\phi)\cdot\mathbf{n} = \bar{\Phi} \text{ on } \partial\Omega_\Phi \tag{3.15b}$$

where $\bar{\Phi}$ is the prescribed flux of $\phi(\mathbf{q},t)$ on $\partial\Omega$, $\bar{\phi}$ is the prescribed density in $\partial\Omega_\phi$, and $\mathbf{n}$ is the outward unit normal field. Here, we consider $\partial\Omega_\phi = \emptyset$, and therefore the flux is prescribed everywhere on the boundary. This implies

$$\frac{\partial\mathcal{H}}{\partial t} = \sum_{i=1}^{n}\int_{\mathcal{V}_i}\nabla f\cdot(D\nabla\phi - \mathbf{U}\phi)\,\text{d}\Omega - \int_{\partial\Omega_\Phi} f(r_i)\bar{\Phi}(\mathbf{q},t)\text{d}\ell \tag{3.16}$$

The first term on the right-hand side of (3.16) is bounded provided that

1. $f(r_i)$ and $\phi(\mathbf{q},t)$ are quadratically integrable functions on Ω, so that they belong to the Hilbert space of functions with bounded square gradient in the domain.

2. $\|\mathbf{U}\|$ is bounded.

The definition of the sensing performance function gives that $f(r_i)$ is a bounded continuous differentiable function, which implies that $\nabla f(r_i)$ is finite on the bounded

domain Ω. For the smooth numerical solution of (3.5) to exist, the normal flux $\bar{\Phi}$ across the boundary must be finite. Then we have bounded gradient of $\phi(\mathbf{q}, t)$. Therefore, ∇f and $\nabla\phi$ are square summable functions on Ω. The second term is bounded if $\bar{\Phi}$ is finite. Therefore, $\frac{\partial\mathcal{H}}{\partial t}$ is finite provide that $\bar{\Phi}$ and $\mathbf{U}$ are finite. □

The Lyapunov stability theorem states that the system is stable if the solution of the differential equation which describe the system can be confined to an arbitrary small circle centred at a point of equilibrium as t approaches infinity. In this work, since the environment evolves, the optimization metric is non-autonomous and therefore the equilibria are the trajectories $\tilde{\mathbf{c}}(\mathcal{V})$ in Ω rather than stationary points. In this case, stronger conditions that Lyapunov's need to be satisfied in order to prove asymptotic convergence of the states $\mathbf{p}_i$ to the centroids' trajectories, conditions that are formally stated through Barbalat's lemma, which allow us to analysis the stability of the non-autonomous system by adding a condition on the second derivative of a non-autonomous Lyapunov function.

Lemma 3 (Barbalat's lemma)**.** *Let $V \geq 0$ be a candidate Lyapunov function, satisfying the following conditions:*

- *$V(x, t)$ is lower bounded;*
- *$\dot{V}(x, t)$ is negative semi-definite;*
- *$\dot{V}(x, t)$ is uniformly continuous in time, which is implied by $\ddot{V}$ being finite;*

then $\dot{V} \to 0$ as $t \to \infty$.

By directly checking that Barbalat's Lemma conditions are satisfied, we can now prove that the error dynamics $\|\tilde{\mathbf{c}}(\mathcal{V}) - \mathbf{p}\|$ converges to zero asymptotically.

Lemma 4. *Let $\phi(\mathbf{q}, t)$ be the density function in the convection-diffusion equation model governed by (3.5), and $\mathcal{H}$ be the coverage metric defined in (3.7). Trajectories generated by the feedback control law (3.9) maximize the coverage metric and are asymptotically stable around the centroids' trajectories.*

Proof. In order to prove asymptotic stability, we consider the candidate Lyapunov function

$$V(\mathbf{p}, t) = \frac{1}{\mathcal{H}(\mathbf{p}, t)} \tag{3.17}$$

Given the properties of the coverage metric $\mathcal{H}$, this candidate Lyapunov function V satisfies the following conditions:

- $V(\mathbf{p},t) > 0$.
  Since $f(r_i)$ and $\phi(\mathbf{q},t)$ are all positive functions according to their definition, it follows that $\int_\Omega f(r_i)\phi(\mathbf{q},t)\mathrm{d}\Omega > 0$. Therefore, $V(\mathbf{p},t)$ is lower bounded and positive.

- $\dot{V}(\mathbf{p},t) < 0$.
  The time derivative of $V(\mathbf{p},t)$ is given by

$$\dot{V}(\mathbf{p},t) = -\frac{\dot{\mathcal{H}}(\mathbf{p},t)}{\mathcal{H}^2(\mathbf{p},t)} \tag{3.18}$$

  The time derivative of $\mathcal{H}$ is given by

$$\begin{aligned}\dot{\mathcal{H}}(\mathbf{p},t) &= \sum_{i=1}^{n}\left(\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}\cdot\dot{\mathbf{p}}_i + \frac{\partial \mathcal{H}_i}{\partial t}\right)\\ &= \sum_{i=1}^{n}\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}\cdot\dot{\mathbf{p}}_i + \frac{\partial \mathcal{H}}{\partial t}\end{aligned} \tag{3.19}$$

  From (3.6) we have $\dot{\mathbf{p}} = \mathbf{u}$, and therefore by substituting the control law (3.9) into (3.19) we obtain

$$\dot{\mathcal{H}}(\mathbf{p},t) = \sum_{i=1}^{n}\kappa\|\tilde{\mathbf{c}}(\mathcal{V}_i) - \mathbf{p}_i\|^2 > 0 \tag{3.20}$$

  Therefore, from definition (3.17), $\dot{V}$ is negative definite. Moreover, (3.20) also implies that the the coverage metric increases along the trajectories generated by (3.9) and further more proves that (3.9) asymptotically maximize the coverage metric. See [53] for more details.

- $\ddot{V}(\mathbf{p},t)$ is finite.
  The second derivative of the candidate Lyapunov function with respect to time is given by

$$\ddot{V}(\mathbf{p},t) = \frac{-\ddot{\mathcal{H}}(\mathbf{p},t)\mathcal{H}^2(\mathbf{p},t) + 2\dot{\mathcal{H}}^2(\mathbf{p},t)}{\mathcal{H}^3(\mathbf{p},t)} \tag{3.21}$$

  with the term $\ddot{\mathcal{H}}$ given by

$$\ddot{\mathcal{H}}(\mathbf{q},t) = \sum_{i=1}^{n}\frac{\partial \dot{\mathcal{H}}}{\partial \mathbf{p}_i}\cdot\dot{\mathbf{p}}_i + \frac{\partial \dot{\mathcal{H}}}{\partial t} \tag{3.22}$$

Substituting (3.20) into (3.22) gives:

$$\ddot{\mathcal{H}}(\mathbf{p},t) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(2\kappa(\tilde{\mathbf{c}}(\mathcal{V}_i)-\mathbf{p}_i)^{\mathrm{T}}\left(\frac{\partial\tilde{\mathbf{c}}(\mathcal{V}_i))}{\partial\mathbf{p}_j}-\mathbf{I}\delta_{ij}\right)\mathbf{u}_j\right) + \sum_{i=1}^{n}(\kappa\left(\tilde{\mathbf{c}}(\mathcal{V}_i)-\mathbf{p}_i\right)^{\mathrm{T}}\frac{\partial\tilde{\mathbf{c}}(\mathcal{V}_i)}{\partial t}\right) \tag{3.23}$$

The workspace Ω is a bounded area in $\mathbb{R}^2$; therefore, the time derivative $\dot{\mathcal{H}}$ which is given by (3.20) is bounded. In order for $\ddot{V}$ to be finite, each term in $\ddot{V}$ must be bounded. Since $\ddot{\mathcal{H}}$ is bounded, the boundedness of $\ddot{V}$ only depends on the boundedness of $\ddot{\mathcal{H}}$. Because of the boundedness of the workspace, $\kappa(\tilde{\mathbf{c}}(\mathcal{V}_i)-\mathbf{p}_i)$ is finite. $\frac{\partial\tilde{\mathbf{c}}(\mathcal{V}_i)}{\partial\mathbf{p}_j}$ is also finite due to the boundedness of the domain [37] [43]. Furthermore, consider the term $\frac{\partial\tilde{\mathbf{c}}(\mathcal{V}_i)}{\partial t}$, which is given by:

$$\frac{\partial\tilde{\mathbf{c}}_i}{\partial t} = \frac{\tilde{m}(\mathcal{V}_i)\int_{\mathcal{V}_i}\mathbf{q}\frac{\partial\tilde{\phi}(\mathbf{q},t)}{\partial t}\mathrm{d}\Omega - \int_{\mathcal{V}_i}\mathbf{q}\tilde{\phi}(\mathbf{q},t)\mathrm{d}\Omega\int_{\mathcal{V}_i}\frac{\partial\tilde{\phi}(\mathbf{q},t)}{\partial t}\mathrm{d}\Omega}{\tilde{m}(\mathcal{V}_i)^2} \tag{3.24}$$

From (3.8c) we know that the boundedness of $\frac{\partial\tilde{\mathbf{c}}}{\partial t}$ depends on the $\frac{\partial\tilde{\phi}}{\partial t}$. (3.8c) allows to calculate

$$\frac{\partial\tilde{\phi}}{\partial t} = -2\frac{\partial f(r_i)}{\partial r_i^2}\frac{\partial\phi}{\partial t} \tag{3.25}$$

Then the boundedness of $\ddot{\mathcal{H}}$ depends on $\frac{\partial\phi}{\partial t}$. The normal flux $\bar{\Phi}$ must be finite in order for the convection-diffusion equation (3.5) to admit a physical solution. From Lemma 2 we have that $\frac{\partial\mathcal{H}}{\partial t}$ is finite. Since $\frac{\partial\mathcal{H}}{\partial t}$ is given by (3.10), $\frac{\partial\phi}{\partial t}$ must be bounded if $\frac{\partial\mathcal{H}}{\partial t}$ is bounded.

Therefore, the candidate Lyapunov function (3.17) satisfies the conditions of Barbalat's Lemma 3, which implies that $\lim_{t\to\infty}\dot{V}=0$. From (3.18) we have:

$$\lim_{t\to\infty}\dot{\mathcal{H}} = 0 \tag{3.26}$$

The expression (3.20) implies

$$\lim_{t\to\infty}\|\mathbf{p}_i-\tilde{\mathbf{c}}(\mathcal{V}_i)\| = 0 \tag{3.27}$$

which also implies that $\mathbf{p}_i$ converges to $\tilde{\mathbf{c}}(\mathcal{V}_i)$. Therefore, the asymptotic stability of generated trajectories $\mathbf{p}_i$ around the centroids' trajectories $\tilde{\mathbf{c}}(\mathcal{V}_i)$ is proven. □

3.3 Area Coverage with Higher-Order Voronoi Partitioning

In this section, an area coverage strategy based on order 2 Voronoi partitions is analyzed. In order 1 Voronoi partitions, each subdomain is assigned to an agent with a non-overlapping area induced by the metric used to compute Voronoi cells. In this Section, by considering a control feedback law based on order 2 Voronoi partitions, each Voronoi cell is hierarchically assigned to a pair of agents, see Subsection 2.3.2 for a discussion of the interpretation of higher order Voronoi partitions.

In order to define arbitrary integer order Voronoi tessellations, let

$$\mathcal{S} = \{1, 2, \dots, n\} \tag{3.28}$$

be the set of indexes labeling the n agents' states $\mathbf{p}_i$ in Ω. Let $\mathcal{T}$ be a subset of $\mathcal{S}$ which contains k elements:

$$\mathcal{T} \in \mathcal{S}, |\mathcal{T}| = k \tag{3.29}$$

From basic combinatorics, the number of possible subsets of k integers (not repeated) from $\mathcal{S}$ is $\binom{n}{k}$, and for each one it is in principle possible to define a Voronoi cell of order k by [54]

$$\mathcal{V}_{\mathcal{T}_\ell} = \{\mathbf{q} \in \Omega \mid d(\mathbf{q}, \mathbf{p}_j) \leq d(\mathbf{q}, \mathbf{p}_h),\ \forall j \in \mathcal{T}, \forall h \in \mathcal{S} \setminus \mathcal{T}\},\ \ell = 1, \dots, \binom{n}{k} \tag{3.30}$$

where d is a suitable distance. For $k = 1$ we simply have the ordinary Voronoi tesselation of order 1. For $k = 2$, each subdomain is assigned to a pair of agents. From now on we are going to consider the case for $k = 2$, in which each Voronoi cell $V_{\mathcal{T}_\ell}$ is assigned to a pair of agents; given n agents, in this case the number of possible cells is $\frac{n(n-1)}{2}$. As in the case for $k = 1$, Voronoi cells in (3.30) are determined by using the sensing performance f as the distance metric d; Moreover, in defining the coverage metric for $k > 1$, we have to define a joint sensing performance $\varphi(\mathcal{T}_\ell)$, which quantifies the collective actions of agents assigned to cell $\mathcal{T}_\ell$. There are different ways to join the sensing performance function [54]; a simple possibility is to consider the additive effect of the two sensing performances, so that the coverage of the cell $\mathcal{V}_{\mathcal{T}_\ell}$ results from the sum of the two contributions $\int_{\mathcal{V}_{\mathcal{T}_\ell}} \phi(\mathbf{q}, t) f(r_{\ell_1}) \mathrm{d}\Omega$ and $\int_{\mathcal{V}_{\mathcal{T}_\ell}} \phi(\mathbf{q}, t) f(r_{\ell_2}) \mathrm{d}\Omega$, where ℓ_1 and ℓ_2 are integer labels of

agents in $\mathcal{T}_\ell = \{\ell_1, \ell_2\} \in \mathcal{S}$, so that

$$\varphi(\mathcal{T}_\ell) = \sum_{j \in \mathcal{T}_\ell} f(r_{\ell_j}) = f(r_{\ell_1}) + f(r_{\ell_2}). \tag{3.31}$$

The order 2 Voronoi partition is build by adding an additional criterion that may result in some of the cells $\mathcal{T}_\ell$ being empty. Specifically, we adopt the hierarchical criterion established in [34], which allows to iteratively construct Voronoi partitions of order k from Voronoi partitions of order $k-1$, starting from the well known order $k = 1$. According the this procedure, the Voronoi cell $\mathcal{V}_{\mathcal{T}_\ell}$ of order 2 is non empty if and only if there exists a pair of agents labeled as ℓ_1, ℓ_2 such that their order 1 cells have non empty intersecting boundaries. Formally:

$$\mathcal{V}_{\mathcal{T}_\ell} \neq \emptyset \iff \exists \{\mathbf{p}_{\ell_1}, \mathbf{p}_{\ell_2}\} : \partial\mathcal{V}_{\ell_1} \cap \partial\mathcal{V}_{\ell_2} \neq \emptyset \tag{3.32}$$

where $\partial\mathcal{V}_{\ell_j}$, $j = 1, 2$ is the boundary of the Voronoi cell of order 1 $\mathcal{V}_{\ell_j}$.

By using the notations just introduced, the coverage metric for order 2 Voronoi partitions is defined by

$$\begin{aligned} \mathcal{H} &= \sum_{\ell=1}^{\binom{n}{k}} \sum_{j=1}^{k} \int_{\mathcal{V}_\ell} \phi(\mathbf{q}, t) f(r_{\ell_j}) \mathrm{d}\Omega \\ &= \sum_{\ell=1}^{n(n-1)/2} \int_{\mathcal{V}_\ell} \phi(\mathbf{q}, t) \left(f(r_{\ell_1}) + f(r_{\ell_2}) \right) \mathrm{d}\Omega \end{aligned} \tag{3.33}$$

Alternatively, in order to write the coverage metric for higher order Voronoi partitions in a way that is structurally analogous to the order 1 case, we introduce the following notations. For each $\mathbf{p}_i \in \Omega$, define $\mathcal{P}_i$ as a set of all $\mathcal{T}_\ell$ which contain the index i:

$$\mathcal{P}_i = \{\mathcal{T}_\ell \mid \mathcal{T}_\ell \subset \mathcal{S}, i \in \mathcal{T}_\ell\} \tag{3.34}$$

Then the union of cells related with $\mathbf{p}_i$ is:

$$\mathcal{W}_i = \bigcup_{\mathcal{T}_\ell \in \mathcal{P}_i} \mathcal{V}_{\mathcal{T}_\ell} \tag{3.35}$$

Based on this, the coverage metric for order k Voronoi partition can be written as

$$\mathcal{H} = \sum_{i=1}^{n} \int_{\mathcal{W}_i} \phi(\mathbf{q}, t) f(r_i) \mathrm{d}\Omega \tag{3.36}$$

In the following, we aim to show that the optimal area coverage control developed for $k = 1$ extends to higher order Voronoi tessellations.

3.3.1 Asymptotic Stability Analysis

With the formalism just introduced, the asymptotic stability analysis for higher order Voronoi tessellations proceeds as in the case of order 1, just by replacing the ith Voronoi cell $\mathcal{V}_i$ (assigned to agent i in the case of order 1 tessellation) with the ith union of higher order Voronoi cells $\mathcal{W}_i$ that are related to agent i.

By defining the generalized risk function, the generalized mass and the generalized centroids as

$$\tilde{\phi}_i(\mathbf{q}, t) = -2\frac{\partial f(r_i)}{\partial r_i^2}\phi(\mathbf{q}, t) \tag{3.37a}$$

$$\tilde{m}(\mathcal{W}_i) = \int_{\mathcal{W}_i} \tilde{\phi}_i(\mathbf{q}, t)\mathrm{d}\Omega \tag{3.37b}$$

$$\tilde{\mathbf{c}}(\mathcal{W}_i) = \frac{1}{\tilde{m}(\mathcal{W}_i)} \int_{\mathcal{W}_i} \mathbf{q}\tilde{\phi}_i(\mathbf{q}, t)\mathrm{d}\Omega \tag{3.37c}$$

the feedback control law is simply given by

$$\mathbf{u}_i(t) = \frac{\partial \mathcal{H}/\partial \mathbf{p}_i}{\|\partial \mathcal{H}/\partial \mathbf{p}_i\|^2} \left(\kappa \|\tilde{\mathbf{c}}(\mathcal{W}_i) - \mathbf{p}_i\|^2 - \frac{\partial \mathcal{H}_i}{\partial t} \right) \tag{3.38}$$

The following Lemma generalizes the analogous one for order 1 Voronoi partitions, establishing that the coverage metric increases along the trajectories generated by the feedback law (3.38).

Lemma 5. *Suppose that the time-varying risk density ϕ is governed by the convection-diffusion equation (3.5). The velocity feedback control law (3.38) asymptotically maximizes the coverage metric defined in (3.36).*

Proof. The proof follows the same reasoning of the one from order 1 Voronoi partitions, and it repeated here with the generalized notation. The time derivative of the coverage metric is given by

$$\dot{\mathcal{H}}(\mathbf{q}, t) = \sum_{i=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} \cdot \dot{\mathbf{p}}_i + \frac{\partial \mathcal{H}_i}{\partial t} \right) \tag{3.39}$$

The partial derivative of the coverage metric is given by

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} &= \int_{\mathcal{W}_i} \phi(\mathbf{q},t)\frac{\partial f(r_i)}{\partial \mathbf{p}_i}\mathrm{d}\Omega \\ &= \int_{\mathcal{W}_i} -2\frac{\partial f(r_i)}{\partial r_i^2}(\mathbf{q}-\mathbf{p}_i)\phi(\mathbf{q},t)\mathrm{d}\Omega \\ &= \tilde{m}(\mathcal{W}_i)\left(\tilde{\mathbf{c}}(\mathcal{W}_i)-\mathbf{p}_i\right)\end{aligned} \tag{3.40}$$

Substitution from (3.40) and (3.38) into (3.39) gives

$$\dot{\mathcal{H}}(\mathbf{q},t) = \sum_{i=1}^{n} \kappa\|\tilde{\mathbf{c}}(\mathcal{W}_i)-\mathbf{p}_i\|^2 > 0 \tag{3.41}$$

which shows the coverage metric increases along trajectories generated by (3.38). □

The following Lemma establishes the asymptotic stability of the trajectories around higher order centroids.

Lemma 6. *Let $\phi(\mathbf{q},t)$ denote density in the convection-diffusion equation model governed by (3.5), and $\mathcal{H}$ be the coverage metric defined in (3.36). If the normal flux Φ across the boundary and the velocity field $\|\mathbf{U}\|$ are finite, trajectories generated by the feedback control law (3.38) asymptotically stabilize to the centroids' trajectories.*

Proof. Consider the following Lyapunov function:

$$V(\mathbf{p},t) = \frac{1}{\mathcal{H}(\mathbf{p},t)} \tag{3.42}$$

This function satisfies the following conditions:

- $V(\mathbf{p},t) > 0$.
  Since $f(r_i)$ and $\phi(\mathbf{q},t)$ are all positive functions, we must have:

  $$\int_{\mathcal{W}_i} \phi(\mathbf{q},t)f(r_i)\mathrm{d}\Omega > 0,\ \forall i \in \mathcal{S} \tag{3.43}$$

  Therefore, $V(\mathbf{p},t)$ must be positive.

- $\dot{V}(\mathbf{p},t) < 0$.
  The derivative of $V(\mathbf{p},t)$ with respect to time is given by

  $$\dot{V}(\mathbf{p},t) = -\frac{\dot{\mathcal{H}}(\mathbf{p},t)}{\mathcal{H}^2(\mathbf{p},t)} \tag{3.44}$$

From lemma 5, we have $\dot{\mathcal{H}} > 0$. Therefore, $\dot{V}$ is negative definite.

- $\ddot{V}$ is finite.
  The second derivative of candidate Lyapunov function with respect to time is given by

$$\ddot{V}(\mathbf{p},t) = \frac{-\ddot{\mathcal{H}}(\mathbf{p},t)\mathcal{H}^2(\mathbf{p},t) + 2\dot{\mathcal{H}}^2(\mathbf{p},t)}{\mathcal{H}^3(\mathbf{p},t)} \tag{3.45}$$

  where $\ddot{\mathcal{H}}$ is given by

$$\ddot{\mathcal{H}}(\mathbf{q},t) = \sum_{i=1}^{n} \frac{\partial \dot{\mathcal{H}}}{\partial \mathbf{p}_i} \cdot \dot{\mathbf{p}}_i + \frac{\partial \dot{\mathcal{H}}}{\partial t} \tag{3.46}$$

  Substitution from (3.38) into (3.46) gives

$$\begin{aligned}\ddot{\mathcal{H}}(\mathbf{p},t) = &\sum_{i=1}^{n}\sum_{j=1}^{n}\left(2\kappa(\tilde{\mathbf{c}}(\mathcal{W}_i) - \mathbf{p}_i)^{\mathrm{T}}\left(\frac{\partial \tilde{\mathbf{c}}(\mathcal{W}_i)}{\partial \mathbf{p}_j} - \mathbf{I}\delta_{ij}\right)\mathbf{u}_j\right) \\ &+ \sum_{i=1}^{n}\left(\kappa(\tilde{\mathbf{c}}(\mathcal{W}_i) - \mathbf{p}_i)^{\mathrm{T}}\frac{\partial \tilde{\mathbf{c}}(\mathcal{W}_i)}{\partial t}\right)\end{aligned} \tag{3.47}$$

  The boundedness of $\ddot{V}$ depends on $\frac{\partial \tilde{\mathbf{c}}(\mathcal{W}_i)}{\partial t}$, which is given by

$$\frac{\partial \tilde{\mathbf{c}}(\mathcal{W}_i)}{\partial t} = \frac{1}{\tilde{m}(\mathcal{W}_i)^2}\left(\tilde{m}(\mathcal{W}_i)\int_{\mathcal{W}_i}\mathbf{q}\frac{\partial \tilde{\phi}_i}{\partial t}\mathrm{d}\Omega - \int_{\mathcal{W}_i}\mathbf{q}\tilde{\phi}_i\mathrm{d}\Omega\int_{\mathcal{W}_i}\frac{\partial \tilde{\phi}_i}{\partial t}\mathrm{d}\Omega\right) \tag{3.48}$$

  Lemma 4 states that if the flux Φ across the boundary and the velocity field $\|\mathbf{U}\|$ are finite, then $\frac{\partial \phi}{\partial t}$ is bounded. Furthermore, $\frac{\partial \tilde{\phi}_i}{\partial t}$ is bounded if $\frac{\partial \phi}{\partial t}$ is bounded. Such guarantee the boundedness of $\ddot{V}$. The Lyapunov candidate function satisfies the three conditions in Barlalat's lemma, which implies that:

$$\lim_{t\to\infty}\dot{\mathcal{H}} = 0 \tag{3.49}$$

  The agents converge to the generalized centroids according to (3.41), which guarantee the asymptotic stability of the system.

□

3.4 Simulation Results

In this section, the theoretical predictions of the feedback control law (3.9) in convective-diffusive environment are demonstrated by using Matlab simulations. We consider a square domain Ω that is being contaminated by a diffusive substance (e.g., spilled oil or radioactive waste), and ϕ represent the density of this substance. The domain is given by

$$\Omega = \{(x, y) \mid 0 \leq x \leq L, 0 \leq y \leq L, L > 0\} \tag{3.50}$$

The evolution of the density is governed by the convection-diffusion equation (3.5). We assign the following boundary conditions

$$x = 0, \bar{\Phi} = 0 \tag{3.51a}$$
$$x = L, \bar{\Phi} = 0 \tag{3.51b}$$
$$y = 0, \bar{\Phi} = 0.1 \tag{3.51c}$$
$$y = L, \bar{\Phi} = 0 \tag{3.51d}$$

The boundary conditions indicate that the diffusive substance enter the domain from the bottom boundary with respect to the adopted system of coordinates. The initial condition of (3.5) is given by

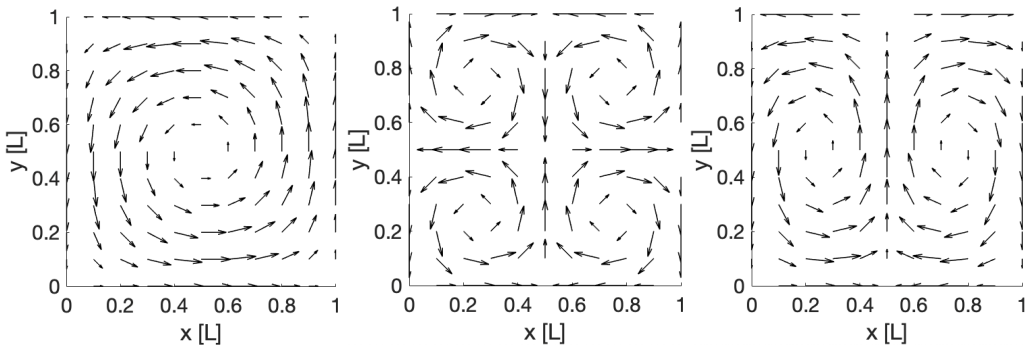
$$t = 0, \ \phi = 10^{-6} \tag{3.52}$$

Three velocity fields were designed to simulate different diffusive-convective environments, with Cartesian coordinates

$$\mathbf{U}_1 = \left(\sin\frac{\pi y}{L}\cos\frac{\pi x}{L}, -\cos\frac{\pi x}{L}\sin\frac{\pi y}{L}\right) \tag{3.53a}$$
$$\mathbf{U}_2 = \left(\sin\frac{2\pi x}{L}\cos\frac{2\pi y}{L}, -\cos\frac{2\pi x}{L}\sin\frac{2\pi y}{L}\right) \tag{3.53b}$$
$$\mathbf{U}_3 = \left(\sin(\frac{2\pi x}{L}\cos\frac{\pi y}{L}, -\cos\frac{2\pi x}{L}\sin\frac{\pi y}{L}\right) \tag{3.53c}$$

The three velocity fields are visualized in Figure 3.1. The methodology to calculate the numerical solution of (3.5) is given in Appendix B.

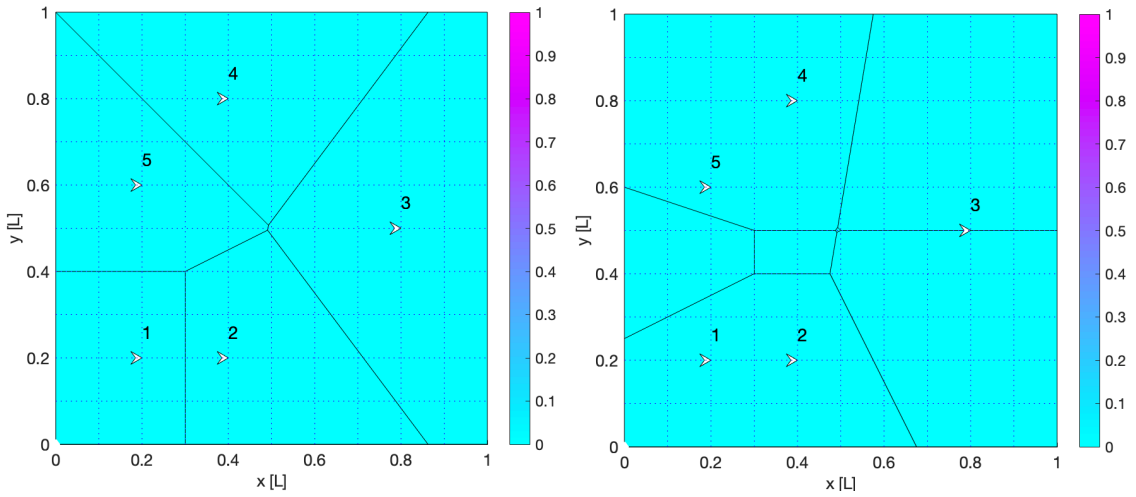
Five agents with the sensing performance function $f(r_i) = e^{-\frac{r_i^2}{2\alpha^2}}$ ($\alpha = 0.4$), are deployed to achieve the optimal coverage of the domain. The agents are placed



(a) $\mathbf{U}_1$ (b) $\mathbf{U}_2$ (c) $\mathbf{U}_3$

Figure 3.1: Three velocity fields given by (3.53).

at initial positions $(0.2, 0.2)L$, $(0.4, 0.2)L$, $(0.8, 0.5)L$, $(0.4, 0.8)L$, and $(0.2, 0.6)L$, as shown by the five white arrows in Figure 3.2. The initial Voronoi partitions are also plotted in Figure 3.2. In this work, the kinetics of agents is not taken into account, which means the agents are modelled as simple fully actuated kinematic particles.



(a) Order 1 initial Voronoi partition. (b) Order 2 initial Voronoi partition.

Figure 3.2: Initial Voronoi partitions (a) order 1 tessellation; (b) order 2 tessellation.

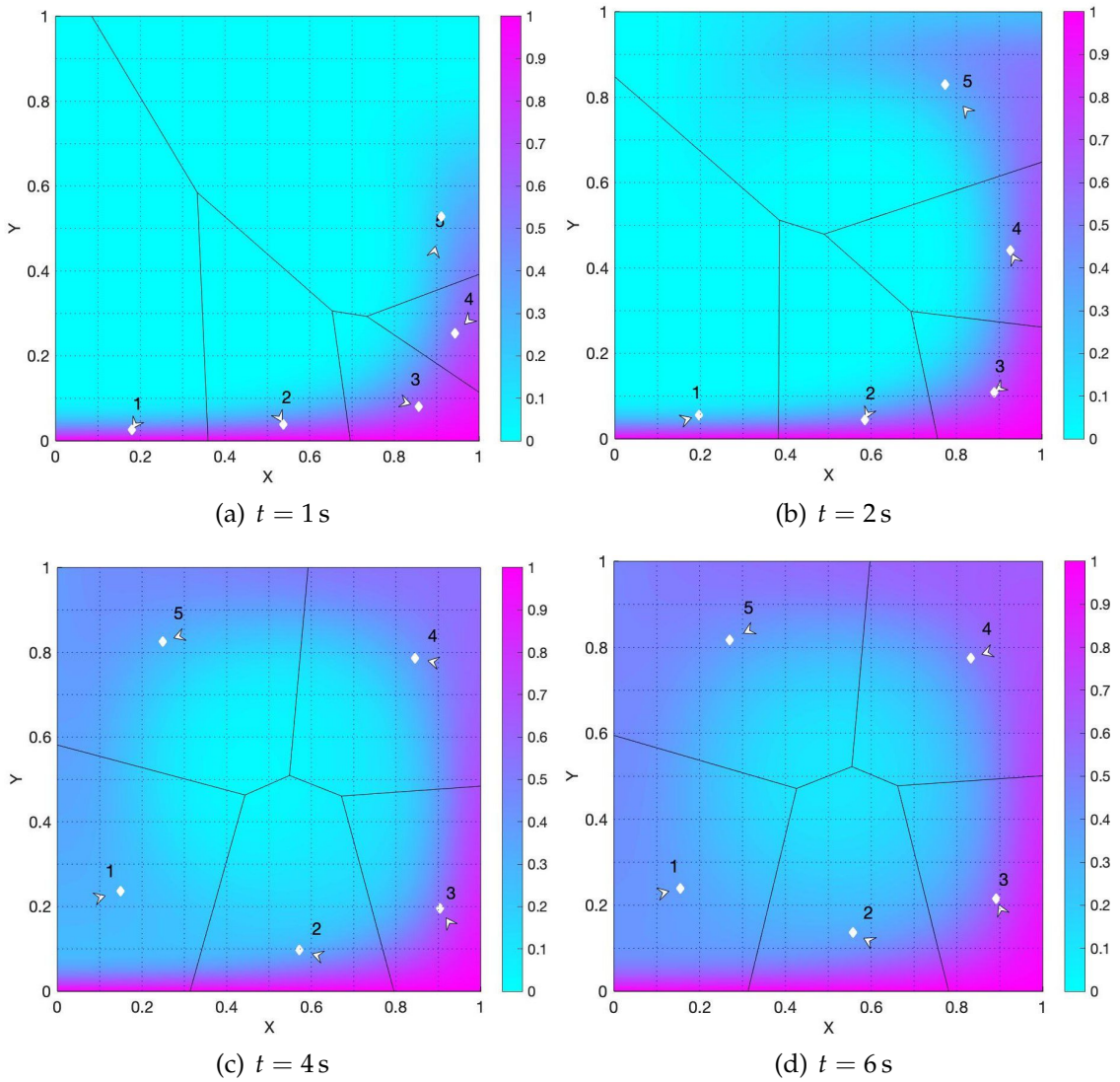
3.4.1 Coverage Control with Order 1 Voronoi Tessellations

In this subsection, the simulation results of three scenarios with order 1 coverage are presented, where the three scenarios are defined respectively by the three environmental convective velocity fields in (3.53). The agents' velocities are determined by the feedback control law (3.9) and the agents' positions are updated

according to the kinematic model (3.6). For the case with velocity field $\mathbf{U}_1$, simulation results are summarized in Figures 3.3 and 3.4 for two different feedback gains (respectively $\kappa = 10$ and 20), with white dots representing the generalized centroids, and with X and Y representing nondimensional coordinates x/L and y/L respectively. The time histories of the errors are presented in Figure 3.5, where the error is defined by

$$\mathbf{e}_i = \|\mathbf{p}_i - \tilde{\mathbf{c}}(\mathcal{V}_i)\| \tag{3.54}$$

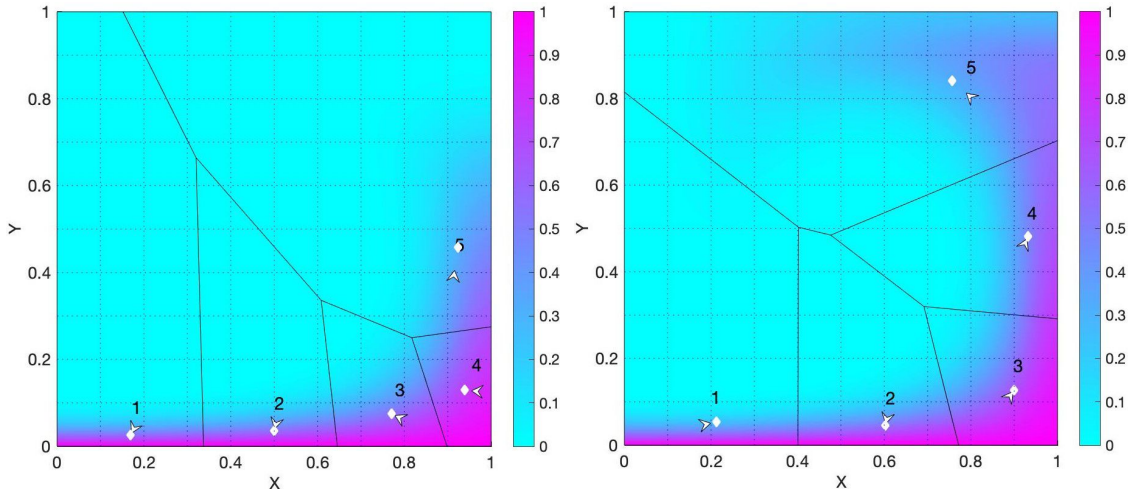
and the evolution of the coverage metric calculated for the two gains is plotted in Figure 3.6.



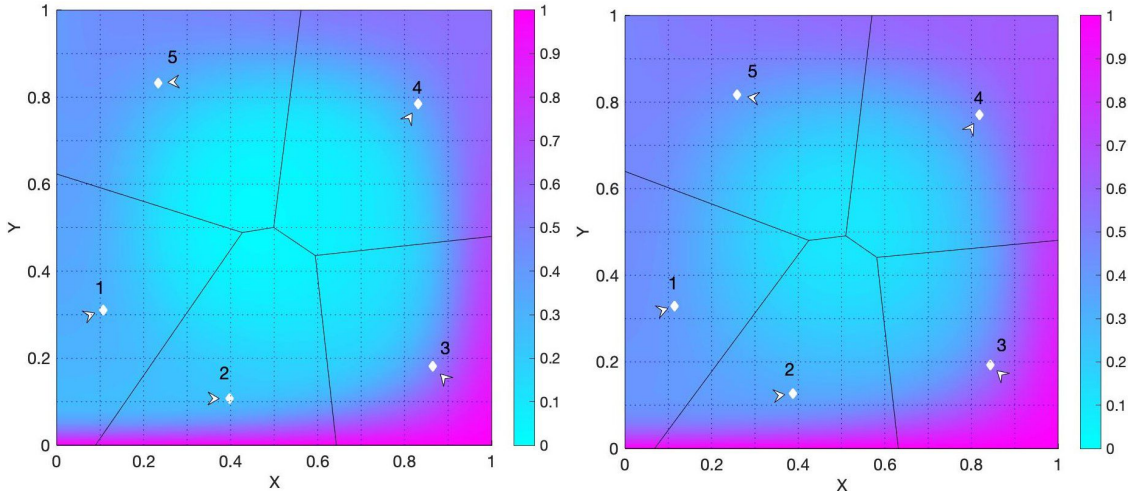
(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.3: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_1$ and feedback gain $\kappa = 10$.

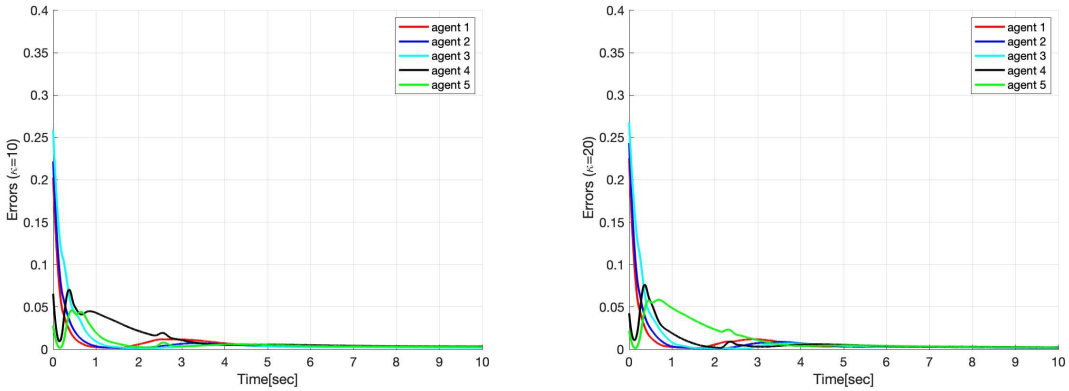


(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$



(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.4: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_1$ and feedback gain $\kappa = 20$.



(a) Errors ($\kappa = 10$). (b) Errors ($\kappa = 20$).

Figure 3.5: Errors with environmental convective velocity $\mathbf{U}_1$.

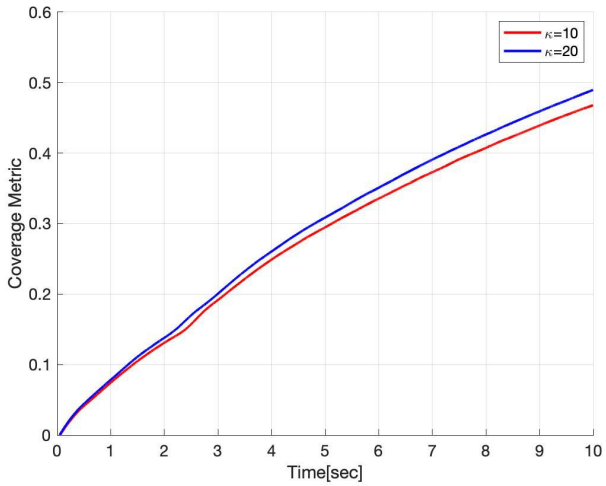
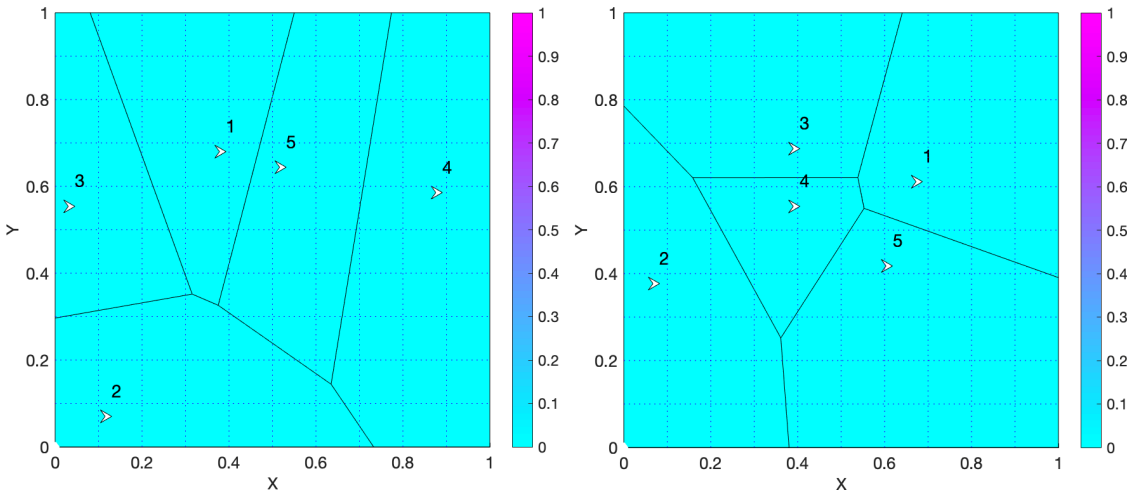


Figure 3.6: Coverage metrics for two different control gains and environmental convective velocity $\mathbf{U}_1$.



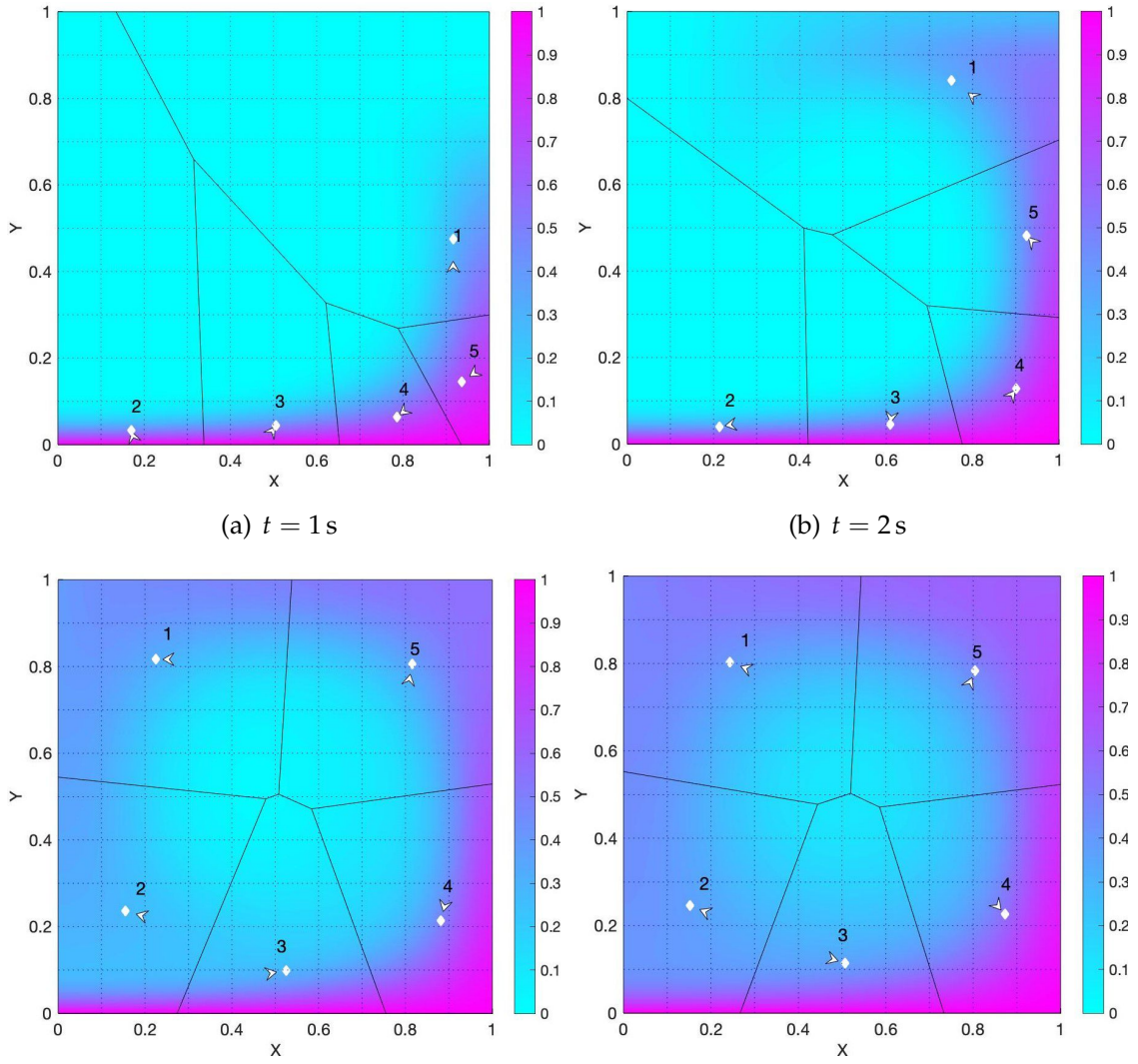
(a) Randomly generated initial positions: set 1 (b) Randomly generated initial positions: set 2

Figure 3.7: Two sets of randomly generated initial positions to simulate the case with convective environmental velocity $\mathbf{U}_1$.

For the same environmental convective velocity $\mathbf{U}_1$, we consider a different set of initial conditions by simulating the system with the randomly generated initial states in Figures 3.7(a) and 3.7(b). The agents' configurations and Voronoi partition at different time instants are presented in Figure 3.8 and Figure 3.9, and the time histories of the coverage metrics and errors are presented in Figure 3.10.

For the case of environmental convective velocity $\mathbf{U}_2$ in (3.53), simulation results are presented in Figures 3.12 and 3.13 for two feedback control gains. The time histories of the coverage metrics and errors are presented in Figures 3.14.

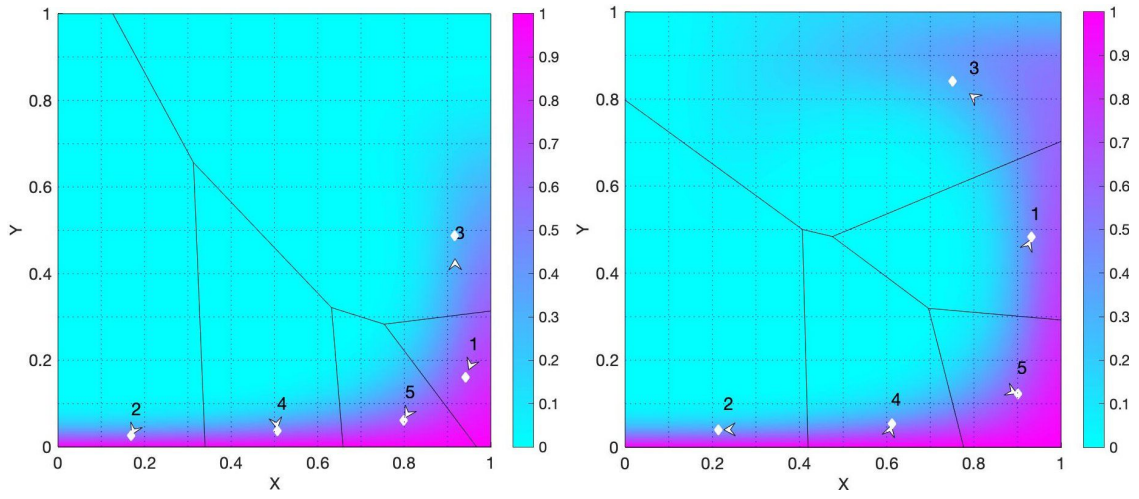
When simulating the case with convective environmental velocity field $\mathbf{U}_3$ in (3.53), time histories of the coverage metrics and collective errors are presented in Figure 3.18, and snapshots of agents' configurations and related Voronoi tessellations of the environment are shown in Figure 3.16 and Figure 3.17.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

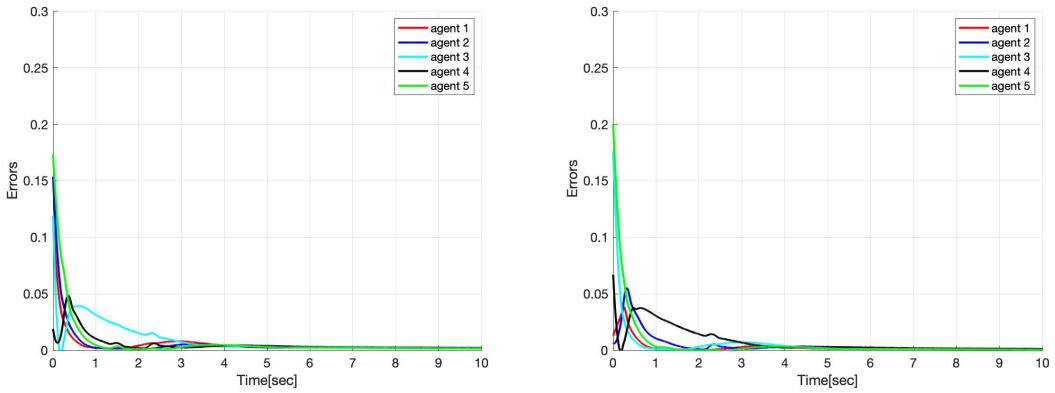
Figure 3.8: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_1$ and feedback gain $\kappa = 20$. The agents' initial position are the randomly generated ones in in Figure 3.7(a).



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.9: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_1$ and feedback gain $\kappa = 20$. The agents' initial position are the randomly generated ones in in Figure 3.7(b).



(a) Errors (Randomly generated initial positions: set 1) .

(b) Errors (Randomly generated initial positions: set 2).

Figure 3.10: Errors with environmental convective velocity $\mathbf{U}_1$ and randomly generated initial positions in Figure 3.7.

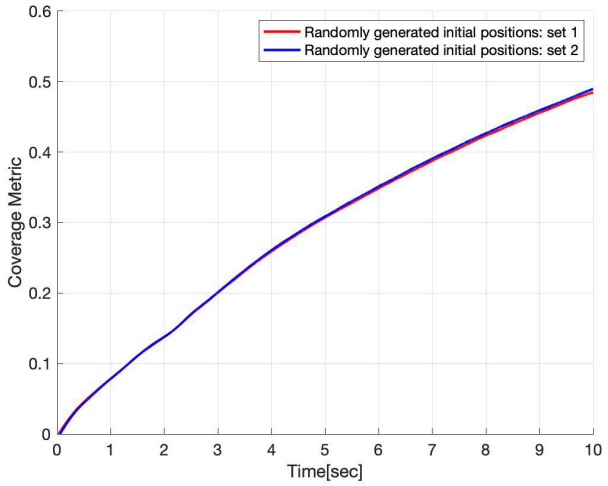
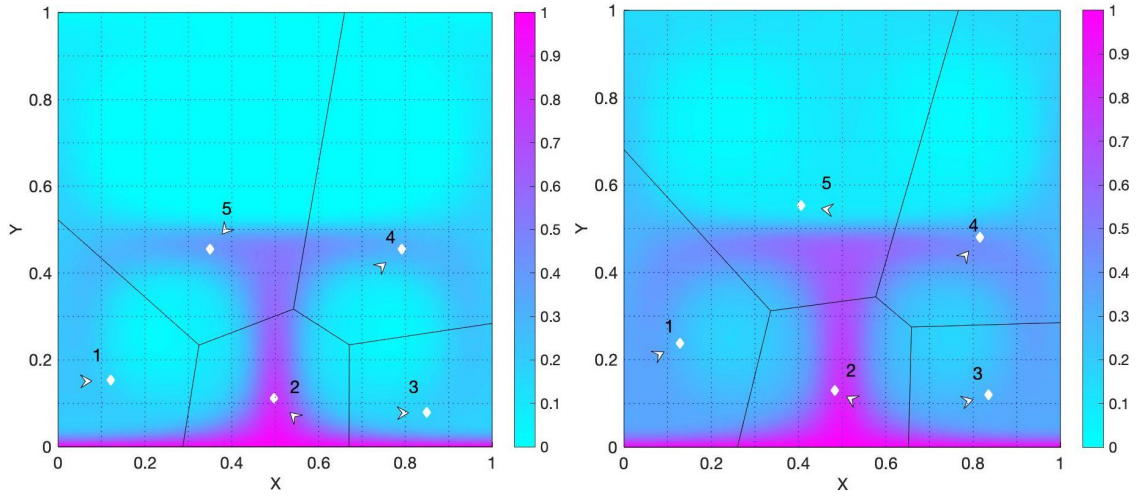
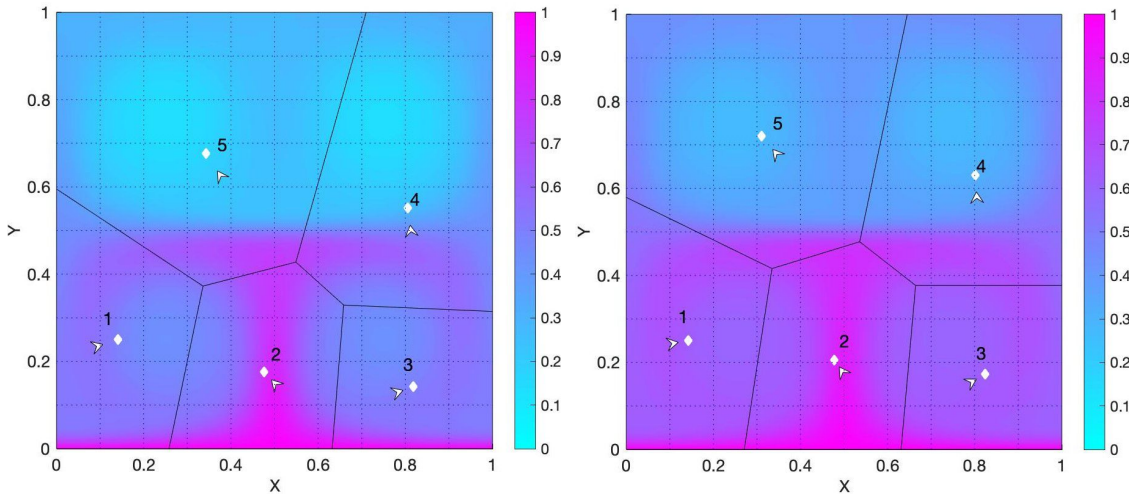


Figure 3.11: Coverage metrics for two randomly generated initial positions in Figure 3.7 and environmental convective velocity $\mathbf{U}_1$.

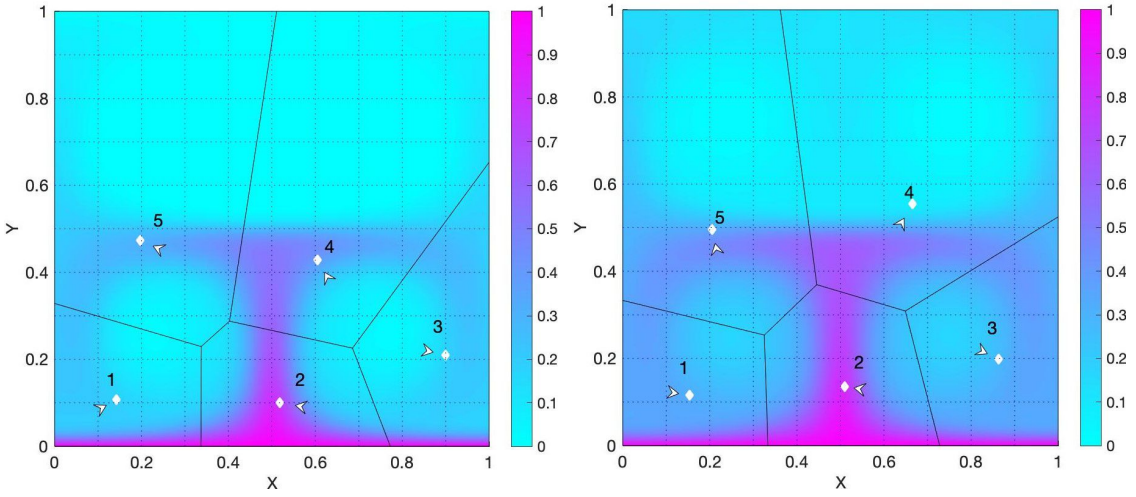


(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$



(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

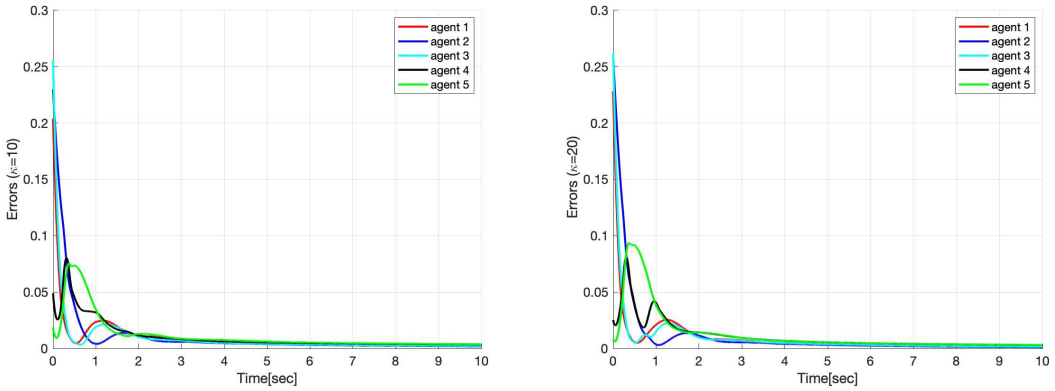
Figure 3.12: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_2$ and feedback gain $\kappa = 10$.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.13: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_2$ and feedback gain $\kappa = 20$.



(a) Errors ($\kappa = 10$) (b) Errors ($\kappa = 20$)

Figure 3.14: Errors with environmental convective velocity $\mathbf{U}_2$.

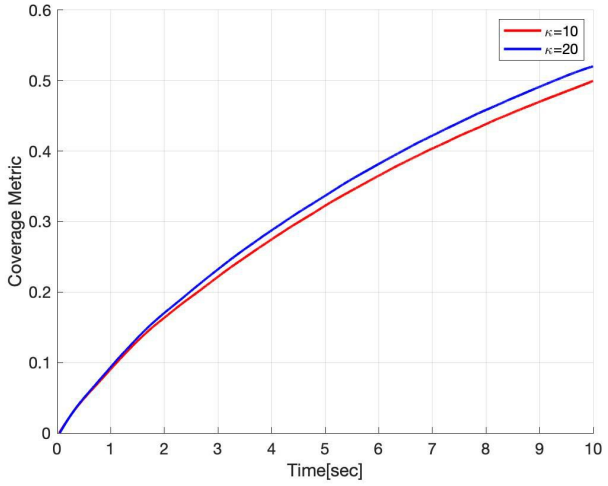
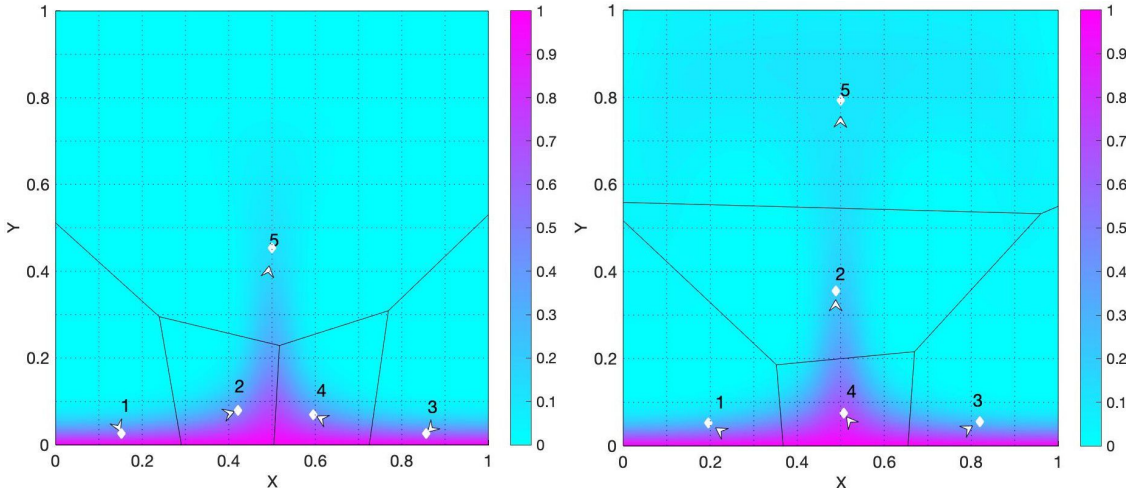


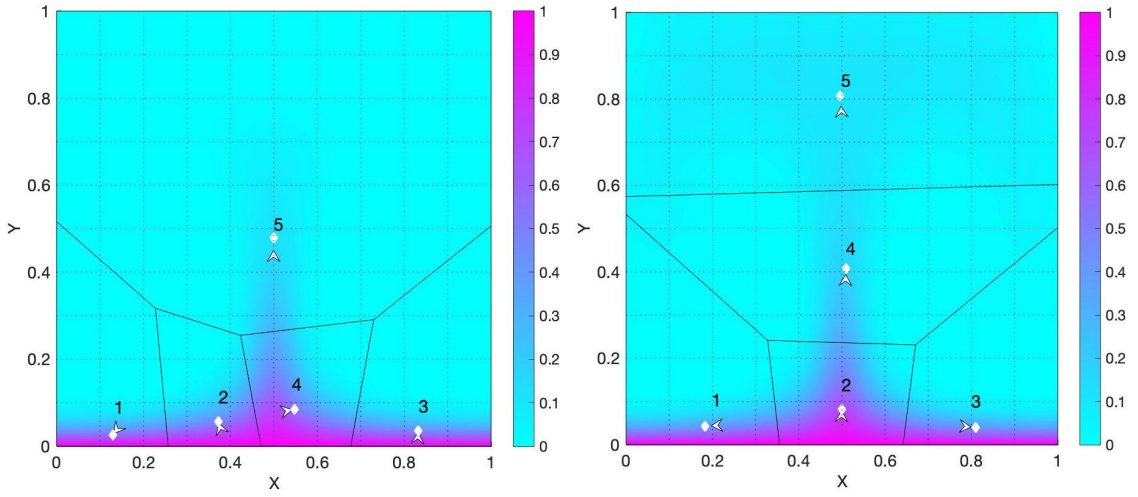
Figure 3.15: Coverage metrics for two different control gains and environmental convective velocity $\mathbf{U}_2$.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

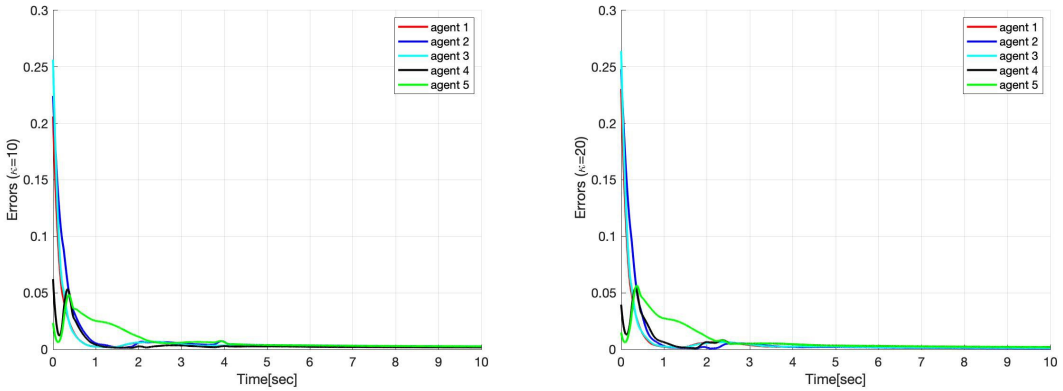
Figure 3.16: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_3$ and feedback gain $\kappa = 10$.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.17: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_3$ and feedback gain $\kappa = 20$.



(a) Errors ($\kappa = 10$) (b) Errors ($\kappa = 20$)

Figure 3.18: Errors with environmental convective velocity $\mathbf{U}_3$

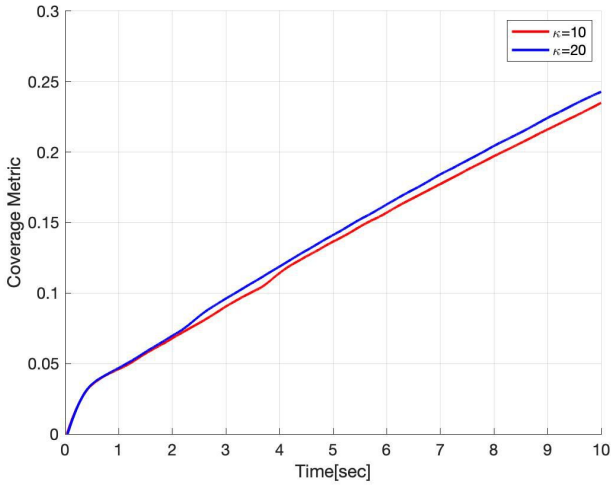


Figure 3.19: Coverage metrics for two different control gains and environmental convective velocity $\mathbf{U}_3$.

According to the analysis in Subsection 3.2.1, the non-autonomous feedback law induces agents' trajectories that track the related generalized centroids with decreasing error. However, the error between agents' states and generalized centroids do not monotonically decrease, as shown by the Figures 3.5, 3.10, 3.14 and 3.18, due to the fact that the density ϕ is time varying. The evolution of the density acts as a disturbance with respect to the control law, which reacts by adjusting the agents' configuration accordingly. When an agent moves from a low density/risks area towards a high risk area, the related generalized centroid will move towards the boundary of the cell, suddenly increasing the tracking error.

For the simulated cases, variations of the initial conditions do not seem to qualitatively affect the systems' performance, although quantitative differences are expected due to the fact that coverage metric equilibria are in general local. Similarities are mainly due to the fact that the evolution of the density is the same, and it asymptotically dictates the trajectories of the agents.

The feedback gain κ in the control law (3.9) clearly influences the results. As (3.9) shows, the velocity of agents depends on the feedback gain and its distance from related generalized centroid. Therefore, the agents move faster with a larger κ. The agents can track the generalized centroids more effectively when κ is larger, as shown by the comparison between Figures 3.5(a) and 3.5(b), 3.14(a) and 3.14(b), 3.18(a) and 3.18(b). In actual hardware implementation, the magnitude of the gain is typically limited by the physics of the system, and therefore it has to be taken into account.

3.4.2 Coverage Control with Order 2 Voronoi Tessellations

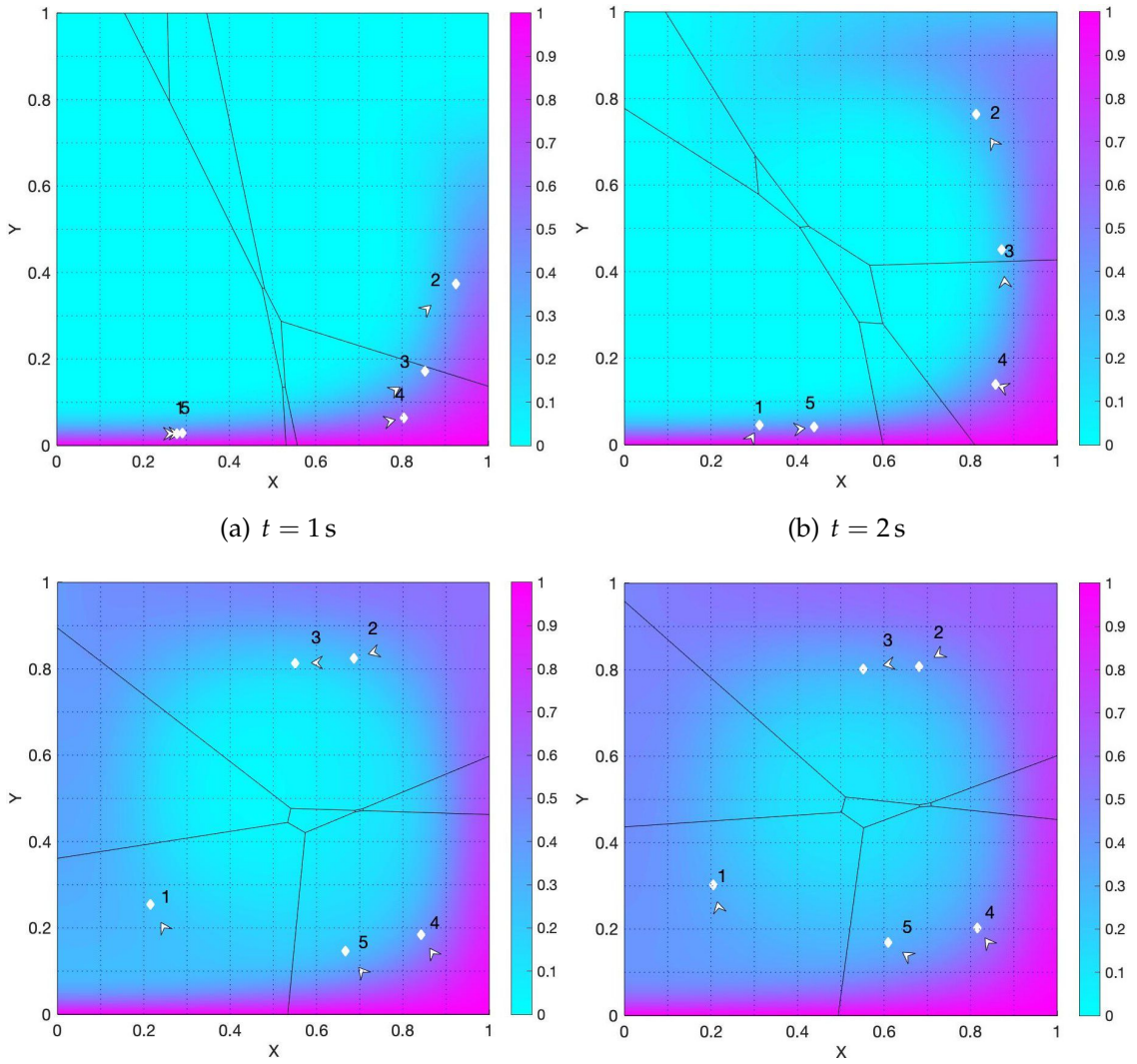
In this subsection, coverage control simulation results with order 2 Voronoi tesselations are presented. Same with the last subsection, three scenarios defined by the three environmental convective velocity fields $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$ in (3.53) are considered. The boundary conditions and the initial condition of the convection-diffusion equation (3.5) are the same as Subsection 3.4.1, and the methodology to calculate the numerical solution of time-varying density function (3.5) is presented in Appendix B.

For the scenario with convective velocity field $\mathbf{U}_1$, time histories of the coverage metrics and errors are presented in Figure 3.22, and snapshots of agents configurations are presented in Figure 3.20 and Figure 3.21. Note that for higher order Voronoi tessellations, there is no one to one correspondence between agents and cells, as in general the number of cells is different than the number of agents. Therefore, agent i no longer tracks the trajectories of the Voronoi centroid of cell $\mathcal{V}_i$ as in the case of order 1, but rather tracks the trajectory of the Voronoi centroid of $\mathcal{W}_i$, see (3.35) and (3.37), which is the union of higher order cells that are related to agent i. Therefore, for this set of simulation results, white dots are no longer directly representing cell's centroids.

For the case with environmental convective velocities $\mathbf{U}_2$ and $\mathbf{U}_3$ in (3.53), the coverage metrics and errors are presented in Figure 3.26 and 3.30 respectively, and systems' configurations for different control gains are presented in Figures 3.24

and 3.25, and in Figures 3.28 and 3.29 respectively.

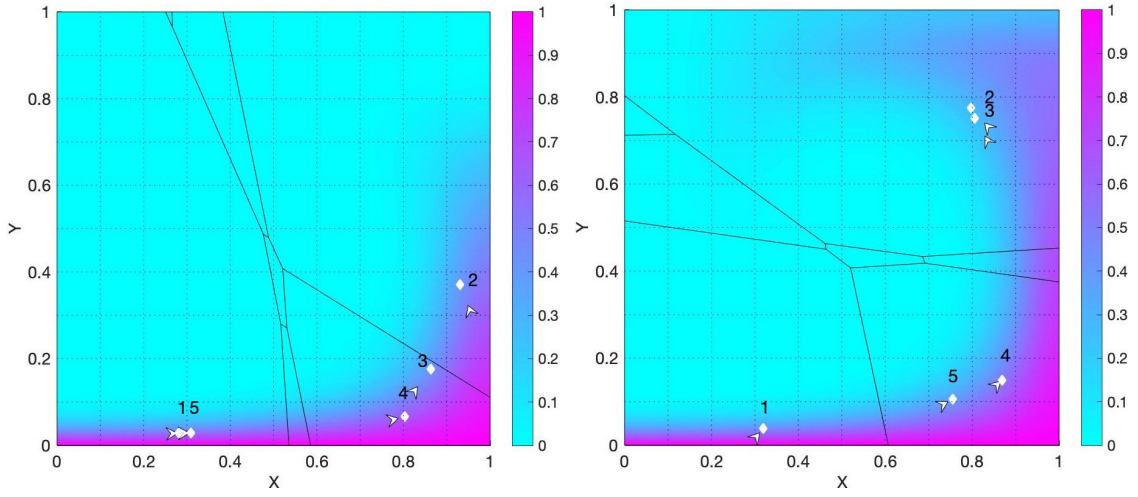
Coverage control based on higher order Voronoi tessellations is characterized by augmented detection areas, following from the definitions of coverage metrics in (3.7) and (3.36). This predicts higher coverage values when everything else is equal, as confirmed by the comparison between the analogous Figures 3.6 and 3.22, 3.14 and 3.27, and 3.18 and 3.31.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

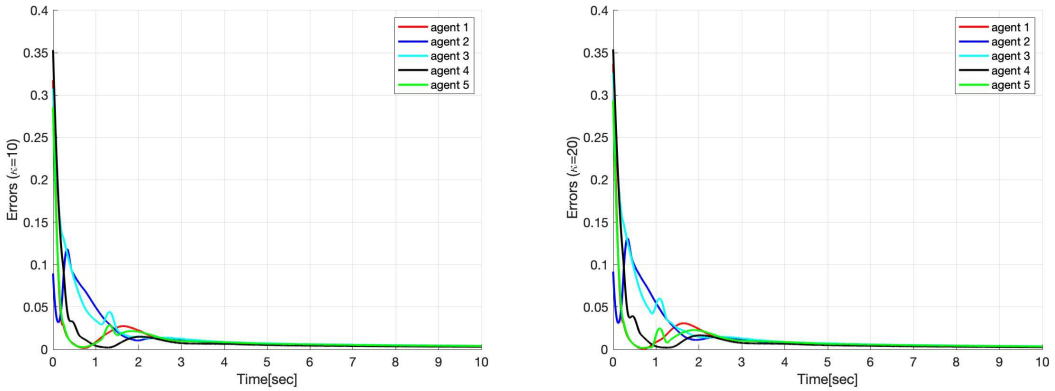
Figure 3.20: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_1$ and feedback gain $\kappa = 10$.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.21: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_1$ and feedback gain $\kappa = 20$.



(a) Errors ($\kappa = 10$) (b) Errors ($\kappa = 20$)

Figure 3.22: Errors with environmental convective velocity $\mathbf{U}_1$.

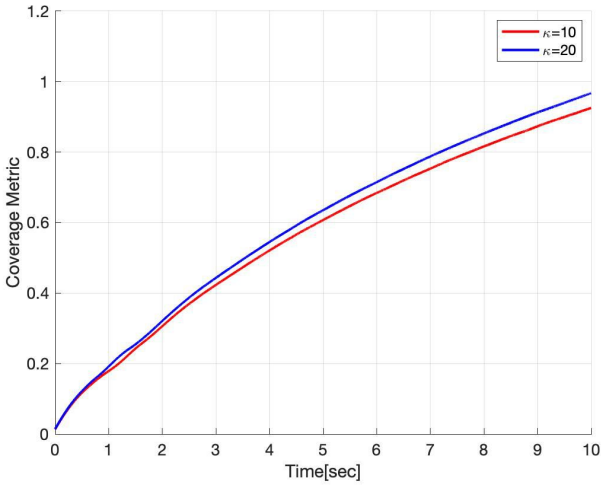
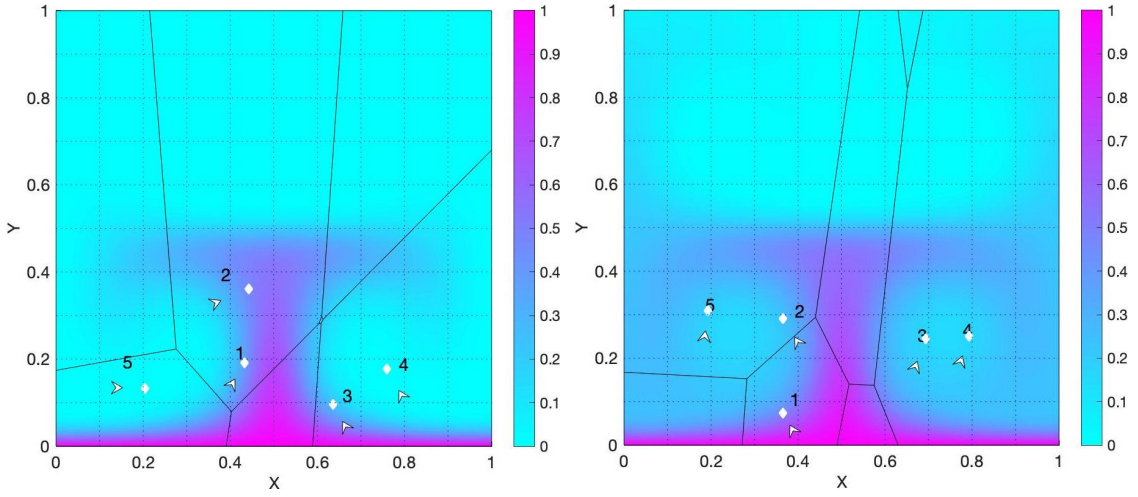


Figure 3.23: Coverage metrics for different control gains and environmental convective velocity $\mathbf{U}_1$.

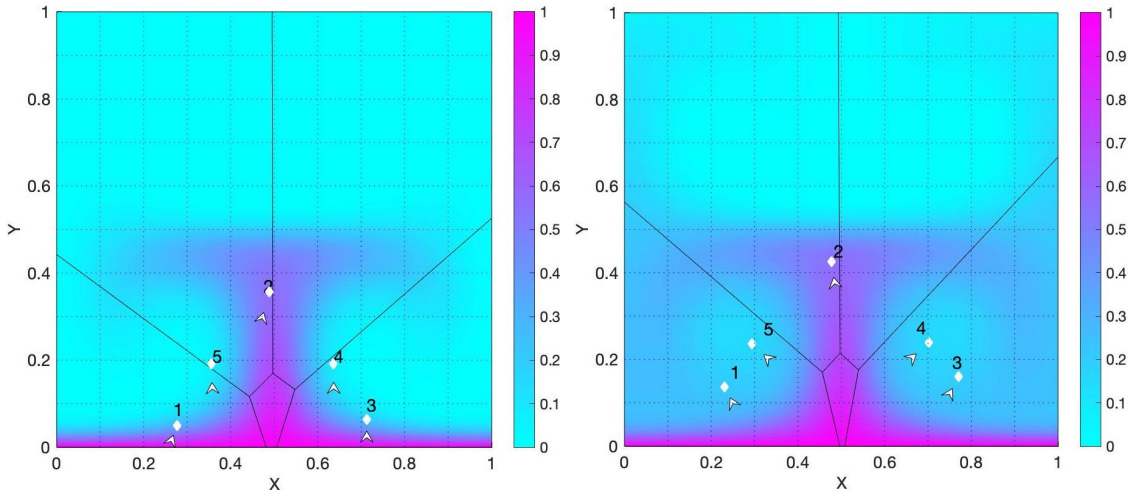
Analogously to the case of order 1 coverage control, the agents approach the optimal configuration faster when a larger feedback gain is given, as shown by a comparison of the blue and red curves in Figures 3.22, 3.27 and 3.31. The initial conditions considered do not seem to have a substantial effect on the errors, that in general show non monotonic behaviour due to changes in the environment related to the evolution of the density $\phi(\mathbf{q}, t)$, which in turn causes the centroids to evolve, and the control law to react by driving the agents to track the centroids trajectories.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

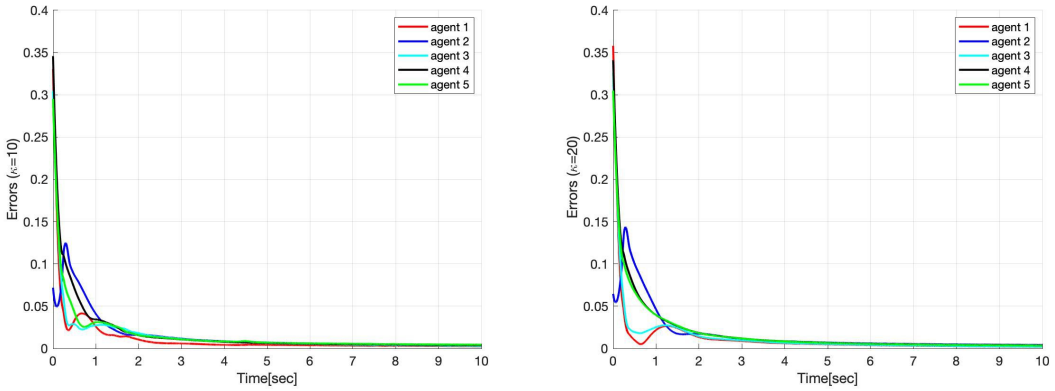
Figure 3.24: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_2$ and feedback gain $\kappa = 10$.



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.25: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_2$ and feedback gain $\kappa = 20$.



(a) Errors ($\kappa = 10$) (b) Errors ($\kappa = 20$)

Figure 3.26: Errors with environmental convective velocity $\mathbf{U}_2$.

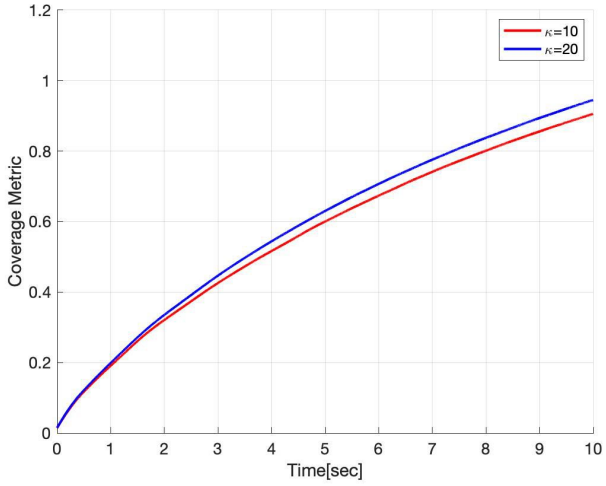
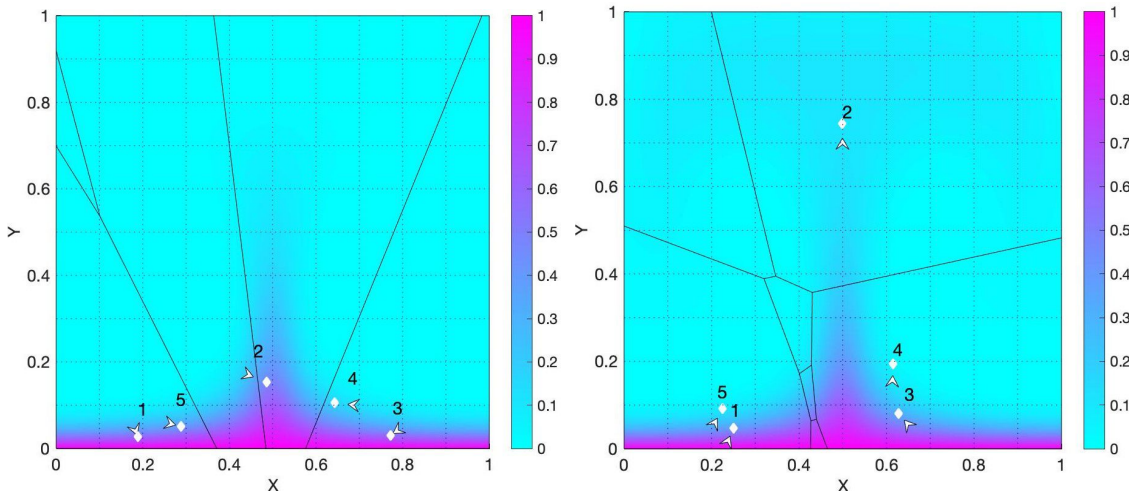
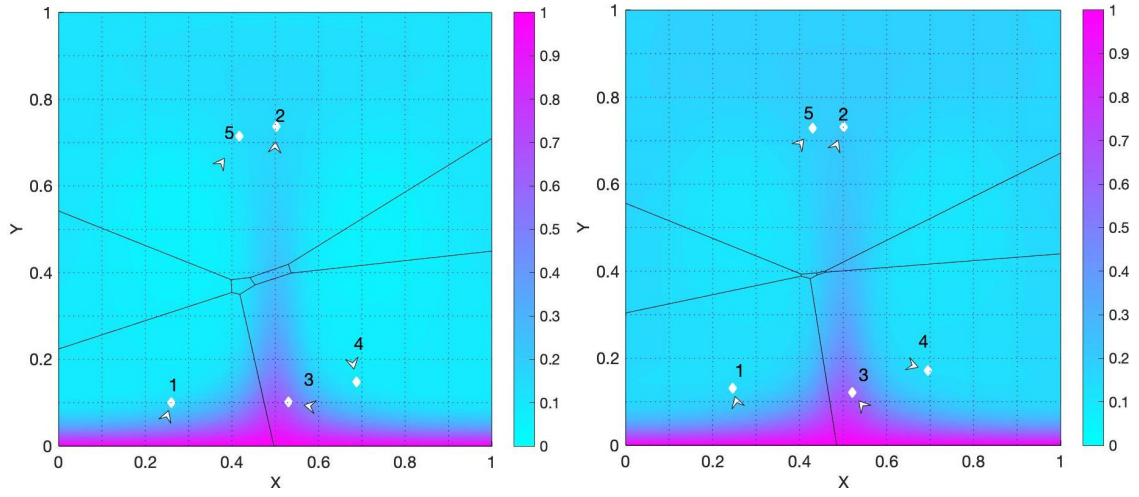


Figure 3.27: Coverage metrics for two different control gains and environmental convective velocity $\mathbf{U}_2$.

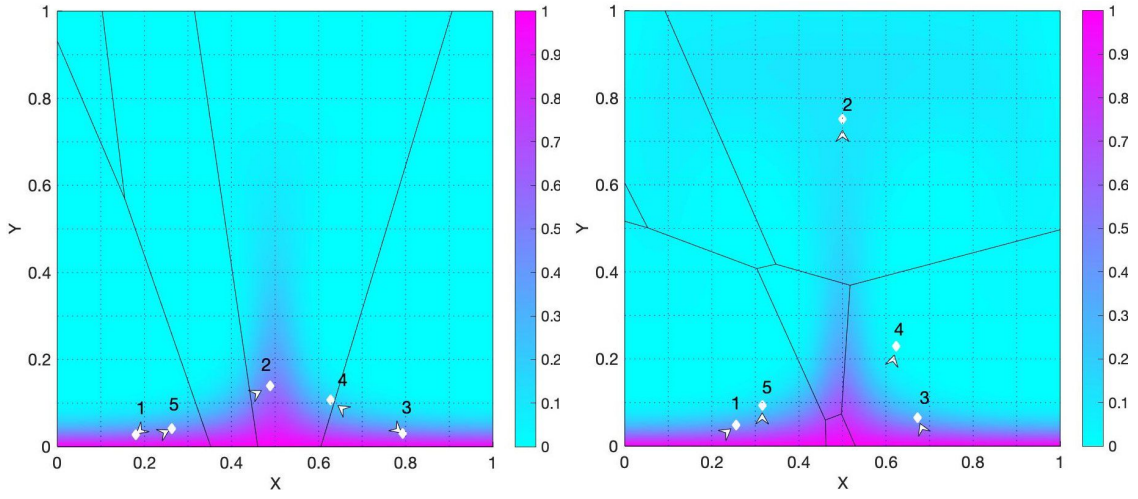


(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

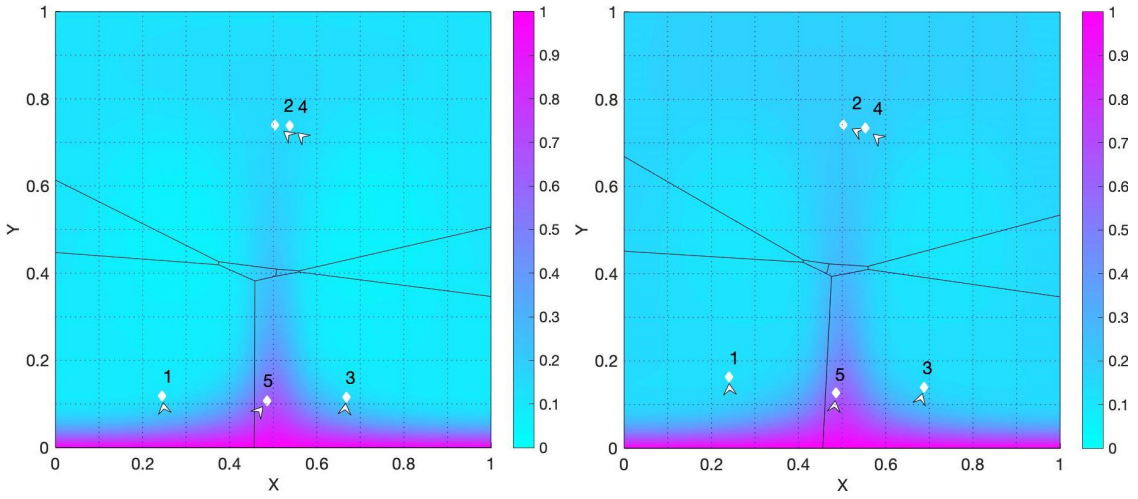


(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.28: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_3$ and feedback gain $\kappa = 10$.

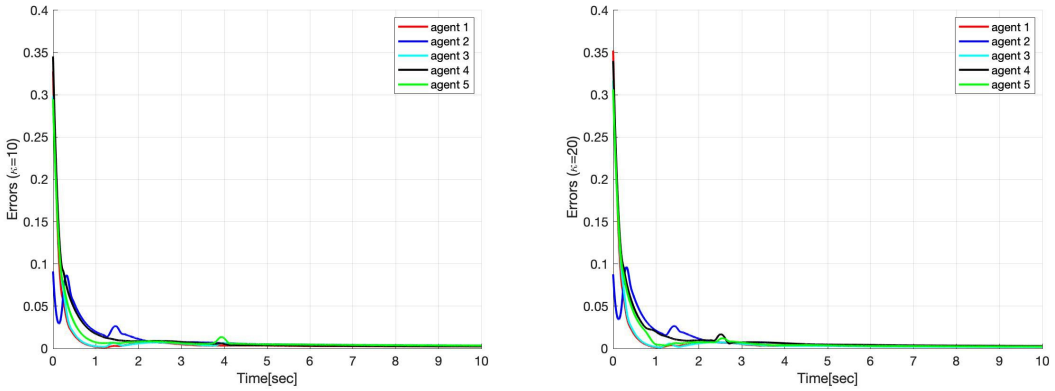


(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$



(c) $t = 4\,\mathrm{s}$ (d) $t = 6\,\mathrm{s}$

Figure 3.29: Snapshots of the agents' distribution in Ω and related Voronoi partition with environmental convective velocity $\mathbf{U}_3$ and feedback gain $\kappa = 20$.



(a) Errors ($\kappa = 10$) (b) Errors ($\kappa = 20$)

Figure 3.30: Errors with environmental convective velocity $\mathbf{U}_3$.

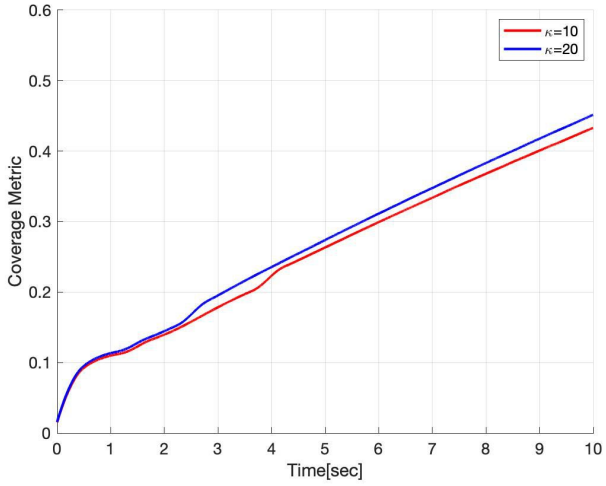


Figure 3.31: Coverage metrics for different control gains and environmental convective velocity $\mathbf{U}_3$.

Chapter 4

Summary and Conclusion

This thesis has focused on optimal area coverage control problems, in which groups of autonomous mobile agents are deployed in a time varying environment. A risk density field is conserved in the environment, with the flux having a diffusive component and a convective component. This generalizes previous results in which the same class of systems had been studied in evolving environments with diffusive risk density. A time varying environment of this type allows to model important classes of problems, for example in environmental monitoring and intervention of submarine and generally aquatic ecosystems susceptible of contamination from diffusive substances that can be also transported by the flow.

The main contribution of the thesis is the extension to convective-diffusive time varying environments of results previously established with the restriction to diffusive environments, specifically working out the technical details of a the proof that a class of non-autonomous feedback laws generates agents' trajectories that are optimal in the sense that they maximize a non-autonomous coverage metric encoding environment and sensors' performances. These trajectories asymptotically track the centroids of a time-varying Voronoi partition of the environment, therefore assigning a sub-region to each agent deployed in the workspace. The proof is also extended to higher order Voronoi partitions, demonstrating the generality of the approach.Simulation results illustrate theoretical prediction, and show comparisons between the order 1 Voronoi partition, and control based on order 2 Voronoi partitions.

This work is based on several idealizing assumptions:

1) The agents are purely kinematic and fully actuated.

2) The information sharing in the group, necessary to compute feedback laws, is perfect so that each agent always know the state of the other agents.

Limited by these assumptions, the kinetics of agents is ignored. However, this model may be unrealistic. More complex modeling must be considered in order to implement coverage control in more realistic convective-diffusive scenarios. Generally, the fluid and/or environmental field motion is expected to be coupled with the agents' motion. Moreover, in real life scenarios, the agents are not allowed for instantaneous steering due to the rotational degrees of freedom.

This work provide a preliminary kinematics analysis for implementation of coverage control in real life convective-diffusive environments. Immediate extension could be directed towards the study of stability of convergence in the presence of time and spatially varying information network topologies, and towards the inclusion of the agents' kinetics.

Appendix A

Matlab Code

Listing A.1: Deployment Based on the Classical Voronoi Partition

```
clear
clc
Δ_x=pi/100;
Δ_y=pi/100;
Δ_t=0.001;      %define the time-step
x=0:Δ_x:pi;
y=0:Δ_y:pi;
t=0;
[xx,yy]=meshgrid(x,y);
N=102;
c_new=zeros(N,N);
c_old=zeros(N,N);
%----------------------------initialzation
   ---------------------------------

n=5;       %number of agents
pt=pi*(rands(2,n)/2+0.5);
workspace=Polyhedron([pi pi;pi 0;0 0;0 pi]);

position=pt;    %time warient
center=zeros(2,n);    %record the centroid of each state
orientation=zeros(2,n);  %record the orientation of next step
step=zeros(2,n);       %record the next step
[Pn]=mpt_voronoi(pt,'bound',workspace);
T=5;

H=zeros(T/0.001,n);
```

```
27  error=zeros(T/0.001,n);
28  u=zeros(T/0.001,n);
29  mm=zeros(n,1);
30
31  %----------numerical solution of diffusion-convection equation
       ------------
32  k=1;
33 while t≤T
34 for i=2:N-1
35     for j=2:N-1
36         div_x=(c_old(i,j+1)-c_old(i,j-1))/2/▵_x;
37         div_y=(c_old(i+1,j)-c_old(i-1,j))/2/▵_y;
38         diffu=(c_old(i,j+1)+c_old(i,j-1)-2*c_old(i,j))/power(▵_x,2)
              +...
39              (c_old(i+1,j)+c_old(i-1,j)-2*c_old(i,j))/power(▵_y,2);
40         temp=0.2*diffu+pi^2*cos(x(j))*sin(y(i))*div_y-...
41             pi^2*sin(x(j))*cos(y(i))*div_x;
42         c_new(i,j)= c_old(i,j)+▵_t*temp;
43     end
44 end
45 for j=1:N
46    c_new(1,j)=1;
47    c_new(N,j)=0;
48 end
49 for i=2:N-1
50    c_new(i,1)=1/3*(4*c_new(i,2)-c_new(i,3));
51    c_new(i,N)=1/3*(4*c_new(i,N-1)-c_new(i,N-2));
52 end
53
54 c_old=c_new;
55 t=t+▵_t;
56
57 for i=2:N-1
58       for j=2:N-1
59       c(i,j)=1/4*(c_new(i+1,j)+c_new(i-1,j)+c_new(i,j-1)+c_new(i,j+1)
            );
60       end
61 end
62
63   c(1,:)=1;
64   c(2:N-1,1)=1/3*(4*c(2:N-1,2)-c(2:N-1,3));
65 %---------------calculate the modified centroid for EACH state
      -------------
```

```
66    c_cal=c+0.0000001*ones(101,101); %regularize the concentration
67    H_Vi=0;                          %initialize the coverage metric
68    error_Vi=0;
69    points=getpoints(x,y,c_cal);
70
71  for i = 1:n
72
73     pts=instate(points,Pn.Set(i).V);
74     mm(i)=modified_mass(pts,Pn.Set(i).Data.voronoi.seed);
75     center(:,i)=modifiedcentriod(pts,Pn.Set(i).Data.voronoi.seed);
76     orientation(:,i)=(center(:,i)-position(:,i))/(norm(center(:,i)-
          position(:,i)));
77     H(k,i)=coveragemetric(pts,position(:,i));
78     u(k,i)=agent_velocity(mm(i),position(:,i),center(:,i),H,k,i);
79     %step(:,i)=pi*0.003*orientation(:,i);
80
81     step(:,i)=u(k,i)*orientation(:,i);
82     position(:,i)=position(:,i)+step(:,i);
83     H(k,i)=coveragemetric(pts,position(:,i));
84     error(k,i)=norm(center(:,i)-position(:,i));
85
86  end
87       k=k+1;
88      [Pn]=mpt_voronoi(position,'bound',workspace);
89   end
```

Listing A.2: Deployment Based on the Order 2 Voronoi Partition

```
1  clear
2  clc
3  ∆_x=pi/100;
4  ∆_y=pi/100;
5  ∆_t=0.001;      %define the time-step
6  x=0:∆_x:pi;
7  y=0:∆_y:pi;
8  t=0;
9  [xx,yy]=meshgrid(x,y);
10 N=102;
11 c_new=zeros(N,N);
12 c_old=zeros(N,N);
13  %-----------------------initislzation
       ------------------------------------
```

```
14
15  n=5;      %number of agents
16  pt=pi*(rands(2,n)/2+0.5);
17  workspace=Polyhedron([pi pi;pi 0;0 0;0 pi]);
18
19  position=pt;              %time warient
20  center=zeros(2,n);        %record the centroid of each state
21  orientation=zeros(2,n);   %record the orientation of next step
22  step=zeros(2,n);          %record the next step
23  [Pn,Pairs]=voronoi2(pt,'bound',workspace);
24  T=5;
25
26   H=zeros(T/0.001,n);
27   error=zeros(T/0.001,n);
28   u=zeros(T/0.001,n);
29   mm=zeros(n,1);
30
31 %--------------------calculate the density-----------------------
32   k=1;
33  while t≤T
34
35 for i=2:N-1
36     for j=2:N-1
37         div_x=(c_old(i,j+1)-c_old(i,j-1))/2/▵_x;
38         div_y=(c_old(i+1,j)-c_old(i-1,j))/2/▵_y;
39         diffu=(c_old(i,j+1)+c_old(i,j-1)-2*c_old(i,j))/power(▵_x,2)
               +...
40              (c_old(i+1,j)+c_old(i-1,j)-2*c_old(i,j))/power(▵_y,2);
41         temp=0.2*diffu+pi^2*cos(x(j))*sin(y(i))*div_y-...
42             pi^2*sin(x(j))*cos(y(i))*div_x;
43         c_new(i,j)= c_old(i,j)+▵_t*temp;
44     end
45 end
46 for j=1:N
47    c_new(1,j)=1;
48    c_new(N,j)=0;
49  end
50 for i=2:N-1
51    c_new(i,1)=1/3*(4*c_new(i,2)-c_new(i,3));
52    c_new(i,N)=1/3*(4*c_new(i,N-1)-c_new(i,N-2));
53 end
54
55     c_old=c_new;
```

```
56      t=t+Δ_t;
57
58  for i=2:N-1
59        for j=2:N-1
60        c(i,j)=1/4*(c_new(i+1,j)+c_new(i-1,j)+c_new(i,j-1)+c_new(i,j+1)
              );
61        end
62  end
63     c(1,:)=1;
64     c(2:N-1,1)=1/3*(4*c(2:N-1,2)-c(2:N-1,3));
65  %------------calculate the modified centroid for EACH state
       -----------------
66     c_cal=c+0.0000001*ones(101,101);  %regularize the concentration
67     H_Vi=0;                           %initialize the coverage metric
68     error_Vi=0;
69
70
71    points=getpoints(x,y,c_cal);
72  for i = 1:n
73      agentzone=findagent(Pairs,i);
74      pts=[];
75      for az=agentzone
76         pts = [pts;instate(points,Pn.Set(az).V)];
77      end
78
79     center(:,i)=centroidofcell(pts);
80     orientation(:,i)=(center(:,i)-position(:,i))/(norm(center(:,i)-
          position(:,i)));
81     %u(k,i)=agent_velocity(mm(i),position(:,i),center(:,i),H,k,i);
82     u(k,i)=pi*0.001;
83
84     step(:,i)=u(k,i)*orientation(:,i);
85     position(:,i)=position(:,i)+step(:,i);
86     H(k,i)=coveragemetric(pts,position(:,i));
87     error(k,i)=norm(center(:,i)-position(:,i));
88
89  end
90      k=k+1;
91      [Pn,Pairs]=voronoi2(position,'bound',workspace);
92   end
```

Listing A.3: Calculation of the Velocity of the Agents

```
function u=agent_velocity(mass,position,center,d,kappa)


        u=1/mass*(kappa*norm(position-center)-d/norm(position-center)
            );


end
```

Listing A.4: Calculation of the Generalized Centroid of the Voronoi Cell

```
function [centroid_of_mass]=modifiedcentriod(points,generator,param)

   %extract the points in the polygon




   %calculate the

   pts_Vi = points(:,1:2);   %coordinates of all n points in the i
      Polygon

   ptrp=repmat(generator',size(points,1),1);   %n x 2 matrix of
      coordinates of generator
   mtrx=pts_Vi-ptrp;
   ri = arrayfun(@(idx) norm(mtrx(idx,:)), 1:size(mtrx,1));  %n x 1
      array ,distance of all n points to generator
   rho_Vi=points(:,3);
   dfp_drisq = -1/(param^3*2*sqrt(2*pi))*exp(-ri.^2/(2*param^2)); %
      using signal strength performance function



   rho_Vi_tilde = -2*(rho_Vi').*dfp_drisq;  %modified density
    rho_Vi = rho_Vi_tilde';
    % compute centroid
        risk_sum_Vi = sum(rho_Vi); % sum of risk (density) of polygon
            Vi
        x_rho_Vi = (1/risk_sum_Vi)*(rho_Vi'*pts_Vi(:,1));
        y_rho_Vi = (1/risk_sum_Vi)*(rho_Vi'*pts_Vi(:,2));
        cm = [x_rho_Vi ; y_rho_Vi];
```

```
27          centroid_of_mass = cm;
28
29
30  end
```

Listing A.5: Calculation of the Generalized Mass of the Voronoi Cell

```
1
2     function m=modified_mass(points,generator,param)
3
4
5     %calculate the
6
7     pts_Vi = points(:,1:2);   %coordinates of all n points in the i
          Polygon
8
9     ptrp=repmat(generator',size(points,1),1);   %n x 2 matrix of
          coordinates of generator
10    mtrx=pts_Vi-ptrp;
11    ri = arrayfun(@(idx) norm(mtrx(idx,:)), 1:size(mtrx,1));  %n x 1
          array ,distance of all n points to generator
12    rho_Vi=points(:,3);
13    dfp_drisq = -1/(param^3*2*sqrt(2*pi))*exp(-ri.^2/(2*param^2)); %
          using signal strength performance function
14
15
16
17    rho_Vi_tilde = -2*(rho_Vi').*dfp_drisq;  %modified density
18     rho_Vi = rho_Vi_tilde';
19     % compute centroid
20     m = sum(rho_Vi)*(pi/300)^2; % sum of risk (density) of polygon Vi
21
22    end
23    end
```

Listing A.6: Calculation of the Coverage Metric

```
1      function h=coveragemetric(points,seed,param)
2       pts_Vi = points(:,1:2);   %coordinates of all n points in the i
           Polygon
3       rho_Vi=points(:,3);
```

```
4       ptrp=repmat(seed',size(points,1),1);    %n x 2 matrix of
            coordinates of generator
5       mtrx=pts_Vi-ptrp;
6       ri = arrayfun(@(idx) norm(mtrx(idx,:)), 1:size(mtrx,1));
7       fs=@(x) exp(-x^2/2/param^2);
8       f=arrayfun(fs,ri);
9       h=f*rho_Vi;
10     end
```

Listing A.7: Plotting of the Agent

```
1   function plot_arrow(amount,position,direction)
2   %position is a 1x2 coordinates of agents
3   %direction is a 2x1 vector
4   for i=1:amount
5       theta=atan2(direction(2,i),direction(1,i));
6       l=pi*0.03;
7       x=position(1,i);
8       y=position(2,i);
9       X=[x,x+l*cos(theta-5*pi/6),x-0.5*l*cos(theta),x+l*cos(theta+5*
            pi/6),x];
10      Y=[y,y+l*sin(theta-5*pi/6),y-0.5*l*sin(theta),y+l*sin(theta+5*
            pi/6),y];
11      Poly_arrow=polyshape(X,Y);
12      plot(Poly_arrow,'FaceColor','w','EdgeColor','w','FaceAlpha',1)
            ;
13      hold on
14  end
15  end
```

Listing A.8: Plotting of the Voronoi Partition

```
1    function plot_voronoi(PolyUnion_Pn)
2     n=PolyUnion_Pn.Num;
3     for i=1:n
4         A=polyshape(PolyUnion_Pn.Set(i).V);
5         F=convhull(A);
6         plot(F,'FaceColor','w','FaceAlpha',0,'LineWidth',0.5);
7         hold on
8      end
9    end
```

Listing A.9: Get The Points Inside The Polyhedron

```
function pts = instate(points,vertices)

   Poly=polyshape(vertices(:,1),vertices(:,2));
   PolyConvHull=convhull(Poly);
   vx=PolyConvHull.Vertices(:,1);
   vy=PolyConvHull.Vertices(:,2);
   in=inpolygon(points(:,1),points(:,2),vx,vy);
   pts=points(in,:);

   end
```

Listing A.10: Discretization of the Workspace

```
function [pts]=getpoints(x,y,concentration)

      %this function aims to generate a array of all discrete points
         in
      %polygon
      %p is the n x 2 matrix specifying n VERTICES of the polygon
 c=concentration;
 pt=zeros(101*101,3);
 for i=1:101
    for j=1:101
        pt((i-1)*101+j,:)=[x(i),y(j),c(j,i)];
    end
 end
 %scatter(pts(:,1),pts(:,2),5,pts(:,3))
N=300;
[X,Y,C]=griddata(pt(:,1),pt(:,2),pt(:,3),linspace(0,pi,N)',linspace
   (0,pi,N),'cubic');
pts=zeros(N*N,3);
for i=1:N

    pts((i-1)*N+1:N*i,1)=X(:,i);
    pts((i-1)*N+1:N*i,2)=Y(:,i);
    pts((i-1)*N+1:N*i,3)=C(:,i);
end
end
```

Listing A.11: Calculation of the order 2 Voronoi Partition

```
function [V,Pairs] = voronoi2(S, varargin)
% Computes the 2-order Voronoi diagram of a set of points
% Syntax:
%   V = voronoi2(S)
%   V = voronoi2(S, 'bound', B)
%   [V, Pairs] = voronoi2(S, 'bound', B)
%
% Inputs:
%   S: seed points stored column-wise
%   B: artificial bounding of the voronoi cells as a Polyhedron
    object
%
% Outputs:
%   V: all Voronoi cells as a PolyUnion (only non-empty cells)
%   Pairs: all combanation of seeds have non-empty voronoi cell
%
% at least one input
narginchk(1, Inf);
if ¬isa(S, 'double') || isempty(S)
    error('First input must be a non-empty set of points.');
end
[nx, n_points] = size(S);
if n_points<2
    error('More than one point please.');
end

% parse and validate options
ip = inputParser;
ip.addParamValue('bound', [], @validate_polyhedron);
ip.parse(varargin{:});
options = ip.Results;
if ¬isempty(options.bound)
    if options.bound.Dim≠nx
        error('The bound must be a polyhedron in R^%d.', nx);
    elseif options.bound.isEmptySet
        error('The bound must not be an empty set.');
    end
end

% create cells
cells(1:n_points) = Polyhedron;
%calculate the 1-order voronoi diagram,and storage in cells
```

```
for i = 1:n_points
    A = zeros(n_points-1, nx);
    b = zeros(n_points-1, 1);
    idx = 1;
    for j = setdiff(1:n_points, i)
        A(idx, :) = 2*(S(:, j)-S(:, i))';
        b(idx) = S(:, j)'*S(:, j) - S(:, i)'*S(:, i);
        idx = idx + 1;
    end
    C = Polyhedron(A, b);
    if ¬isempty(options.bound)
       C=C.intersect(options.bound);
    end
    C.Data.voronoi.seed = S(:, i);
    cells(i) = C;
end

Pairs=combnk(1:n_points,2);
n_face=size(Pairs,1);
cellisempty(1:n_face)=false;
Cells2(1:n_face)=Polyhedron;% cells2 storage 2-order voronoi cell
for k=1:n_face
    i_=Pairs(k,1);
    j_=Pairs(k,2);
    %remove the seed i_,then calculate the 1-order voronoi cell for
       seed
    % j_C
    n_i=setdiff(1:n_points,i_);  %S_x:denote index of set of seed {S-
       i_C}

    A_=zeros(n_points-2,nx);
    b_=zeros(n_points-2,1);
    idx2=1;
    for l=setdiff(n_i,j_)  %calculate the cell j_
       A_(idx2,:)=2*(S(:,l)-S(:,j_))';
       b_(idx2)=S(:,l)'*S(:,l)-S(:,j_)'*S(:,j_);
       idx2=idx2+1;
    end
    C_=Polyhedron(A_,b_);
    %calculate the intersection of Cell{S}_i_ and Cell{S-i_}_j_,if
       not intersect
    %break the loop,if intersect continue
    if ¬(C_.doesIntersect(cells(i_)))
```

```
82
83          cellisempty(k)=true;
84          continue
85      end
86      H1=C_.and(cells(i_));
87      %remove seed j_ and calculate the 1-order voronoi cell for seed
            i_
88      A_=zeros(n_points-2,nx);
89      b_=zeros(n_points-2,1);
90      idx3=1;
91      n_j=setdiff(1:n_points,j_);
92      for l=setdiff(n_j,i_)
93         A_(idx3,:)=2*(S(:,l)-S(:,i_))';
94         b_(idx3,:)=S(:,l)'*S(:,l)-S(:,i_)'*S(:,i_);
95         idx3 = idx3+1;
96      end
97        C_ = Polyhedron(A_,b_);
98        %calculate the intersection of Cell{S}_j_ and Cell{S-j_}_i_
99        H2 = C_.and(cells(j_));
100       %calculate the union of Polyhedron H1 and H2
101       H=H1.or(H2);
102       H.Data.voronoi.seed_1 = S(:,i_);
103       H.Data.voronoi.seed_2 = S(:,j_);
104       %remove the set outside the bound of workspace
105       Cells2(k)=H;
106
107 end
108 %remove the empty cell from Cells(2)
109 Cells2=Cells2(¬cellisempty);
110 Pairs=Pairs(¬cellisempty,:);
111
112 V = PolyUnion(Cells2);
113 % by definition the cells do not overlap
114 V.setInternal('Overlaps', false);
115 % by definition the union of cells is convex
116 V.setInternal('Convex', true);
117 % by definition the cells are connected
118 V.setInternal('Connected', true);
119
120 end
```

Appendix B

Finite difference method for two dimensional convection - diffusion equation

In this part, the numerical soulution of the convection-diffusion equation (3.5) is presented. The solution was used to describe the time-varying risk density function. The following simplifications are valid if the convection coefficient is constant and the velocity field describes an incompressible flow:

$$\nabla \cdot (D\nabla\phi) = D\nabla^2\phi \tag{B.1}$$

$$\nabla \cdot (\mathbf{U}\phi) = \mathbf{U} \cdot \nabla\phi \tag{B.2}$$

The following equations can be established in 2-D domains:

$$D\nabla^2\phi = D\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right) \tag{B.3}$$

$$\mathbf{U} \cdot \nabla\phi = u\frac{\partial\phi}{\partial x} + v\frac{\partial\phi}{\partial y} \tag{B.4}$$

where u and v are components of convection flow in x and y direction. Thereafter, the convection-diffusion equation can be simplified to

$$\frac{\partial\phi}{\partial t} = D\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right) - u\frac{\partial\phi}{\partial x} - v\frac{\partial\phi}{\partial y} \tag{B.5}$$

B.0.1 Finite Difference Scheme of Convection-Diffusion Equation

To construct the difference scheme, Patanker employs a discretization method in [55]. This method was used to solve the heat transfer problem, but it also provides a reference to the problem under discussion. Through this method, Patanker builds a two-dimensional grid on the workspace. As shown in Figure. B.1, two families of lines along the coordinated x and y direction are given by:

$$x = i\Delta x,\ i = 0, \pm 1, \pm 2, \cdots$$

$$y = j\Delta y,\ j = 0, \pm 1, \pm 2, \cdots$$

The intersection points of two families of lines are called grid points or nodes. Two nodes are neighborhoods if the distance between them is one unit size. Node W and E are x-direction neighborhoods while node N and node S are y direction neighborhoods.

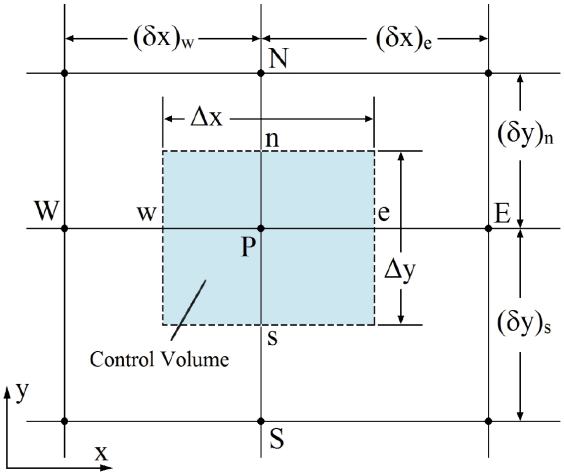


Figure B.1: Neighborhoods of node P

A node is interior when all neighborhoods belong to $\Omega - \partial\Omega$ or when the node is frontier — whereby least one neighborhood belongs to $\partial\Omega$. The control volume can be denoted by a blue rectangle, surrounded by dashed lines. Δx, Δy are the length and width of the control volume respectively; while δx and δy are the distance between two nodes along the coordinated x-direction and y-direction, respectively. Further, the following approximations can be adopted to calculate the

coverage metric defined by (3.7) and (**??**):

$$\int_{\mathcal{V}_i} \mathbf{q}\tilde{\phi}(\mathbf{q},t)\mathrm{d}\mathbf{q} \approx \sum_{\hat{\mathbf{q}}\in\mathcal{V}_i} h^2\hat{\mathbf{q}}\tilde{\phi}(\hat{\mathbf{q}},t) \tag{B.6a}$$

$$\int_{\mathcal{V}_i} \tilde{\phi}(\mathbf{q},t)\mathrm{d}\mathbf{q} \approx \sum_{\hat{\mathbf{q}}\in\mathcal{V}_i} h^2\tilde{\phi}(\hat{\mathbf{q}},t) \tag{B.6b}$$

where $\hat{\mathbf{q}}$ are grid points.

Let $\Delta x = \delta x = \Delta y = \delta y = h$, the discretized workspace is defined by N_x and N_y nodes in x and y direction respectively. The node locates at $(i\delta x, j\delta y)$ can be denoted as (i,j) in the grid, for $i = 0,1,\cdots,N_x-1$; $j = 0,1,\cdots,N_y-1$. For simplicity, we are introducing the notation:

$$\phi^n_{i,j} = \phi \mid x = i\delta x, y = \delta y, t = n\delta t \tag{B.7}$$

where δt denotes the sampling time. In the two-dimensional domain, four continuous interior nodes around $\phi^n_{i,j}$ at time $t = n\delta t$ can be denoted as:

$$\phi^n_{i,j+1} = \phi \mid x = i\delta x,\ y = (j+1)\delta y,\ t = n\delta t \tag{B.8a}$$

$$\phi^n_{i,j-1} = \phi \mid x = i\delta x,\ y = (j-1)\delta y,\ t = n\delta t \tag{B.8b}$$

$$\phi^n_{i+1,j} = \phi \mid x = (i+1)\delta x,\ y = j\delta y,\ t = n\delta t \tag{B.8c}$$

$$\phi^n_{i-1,j} = \phi \mid x = (i-1)\delta x,\ y = j\delta y,\ t = n\delta t \tag{B.8d}$$

The following finite difference scheme can be established by Taylor series expansion:

$$\frac{\phi^{n+1}_{i,j} - \phi^n_{i,j}}{\delta t} = \left[\frac{\partial\phi}{\partial t}\right]^n_{i,j} + O(\delta t) \tag{B.9a}$$

$$\frac{\phi^n_{i+1,j} - \phi^n_{i-1,j}}{2h} = \left[\frac{\partial\phi}{\partial x}\right]^n_{i,j} + O(h) \tag{B.9b}$$

$$\frac{\phi^n_{i,j+1} - \phi^n_{i,j-1}}{2h} = \left[\frac{\partial\phi}{\partial y}\right]^n_{i,j} + O(h) \tag{B.9c}$$

$$\frac{\phi^n_{i+1,j} - 2\phi^n_{i,j} + \phi^n_{i-1,j}}{h^2} = \left[\frac{\partial^2\phi}{\partial x^2}\right]^n_{i,j} + O(h^2) \tag{B.9d}$$

$$\frac{\phi^n_{i,j+1} - 2\phi^n_{i,j} + \phi^n_{i,j-1}}{h^2} = \left[\frac{\partial^2\phi}{\partial y^2}\right]^n_{i,j} + O(h^2) \tag{B.9e}$$

Substituting (B.9a-B.9e) into (B.5), we establish a approximate difference equation of differential equation:

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\delta t} = D\left(\frac{\phi_{i+1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i-1,j}^{n}}{h^2} + \frac{\phi_{i,j+1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j-1}^{n}}{h^2}\right) - u\frac{\phi_{i+1,j}^{n} - \phi_{i-1,j}^{n}}{2h} - v\frac{\phi_{i,j+1}^{n} - \phi_{i,j-1}^{n}}{2h} \tag{B.10}$$

which can be written in recursive format:

$$\phi_{i,j}^{n+1} = (1 - 4\alpha)\phi_{i,j}^{n} + (\alpha - \beta)\phi_{i+1,j}^{n} + (\alpha - \gamma)\phi_{i,j+1}^{n} + (\alpha + \beta)\phi_{i-1,j}^{n} + (\alpha + \gamma)\phi_{i,j-1}^{n} \tag{B.11}$$

where

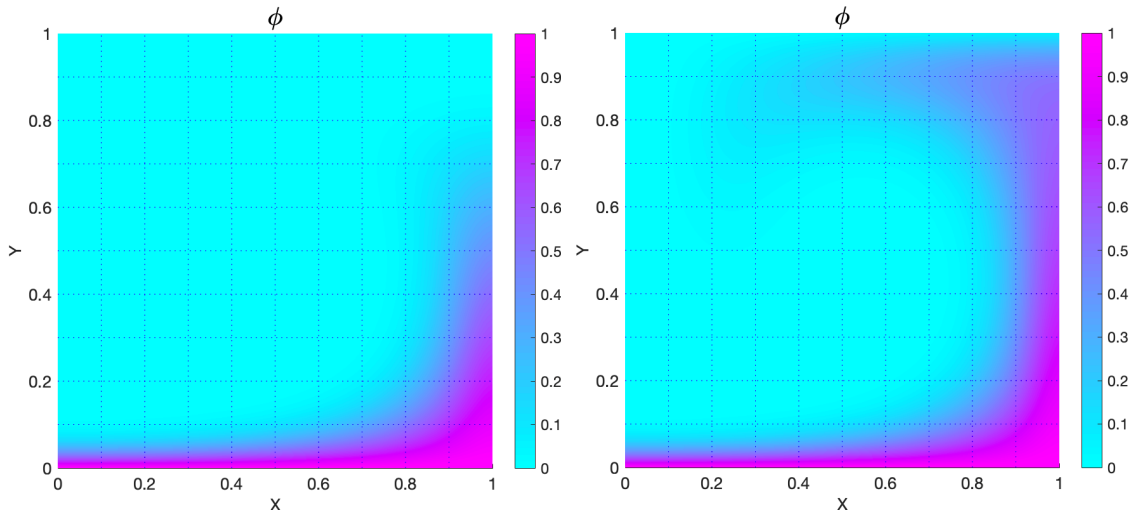
$$\alpha = \frac{D\delta t}{h^2},\ \beta = \frac{u\delta t}{2h},\ \gamma = \frac{v\delta t}{2h}$$

Note that α is constant; however, β and γ are variables due to the varying velocity in x and y directions.

B.0.2 Numerical Solution

In this section, the numerical solutions of the convection-diffusion equation (B.5) with velocity field $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$ are illustrated. The three velocity fields are given by (3.53). In this work, The sampling time δt in the differential scheme (B.11) should be very small to reduce truncation error. Moreover, the sampling time δt and the step size of space h should be matched, so that the oscillation of the solution can be reduced. The differential scheme performs well in simulation when $\delta t = 0.001s$ and $h = 0.01$. The diffusion coefficient D in (B.5) was set to 0.006.

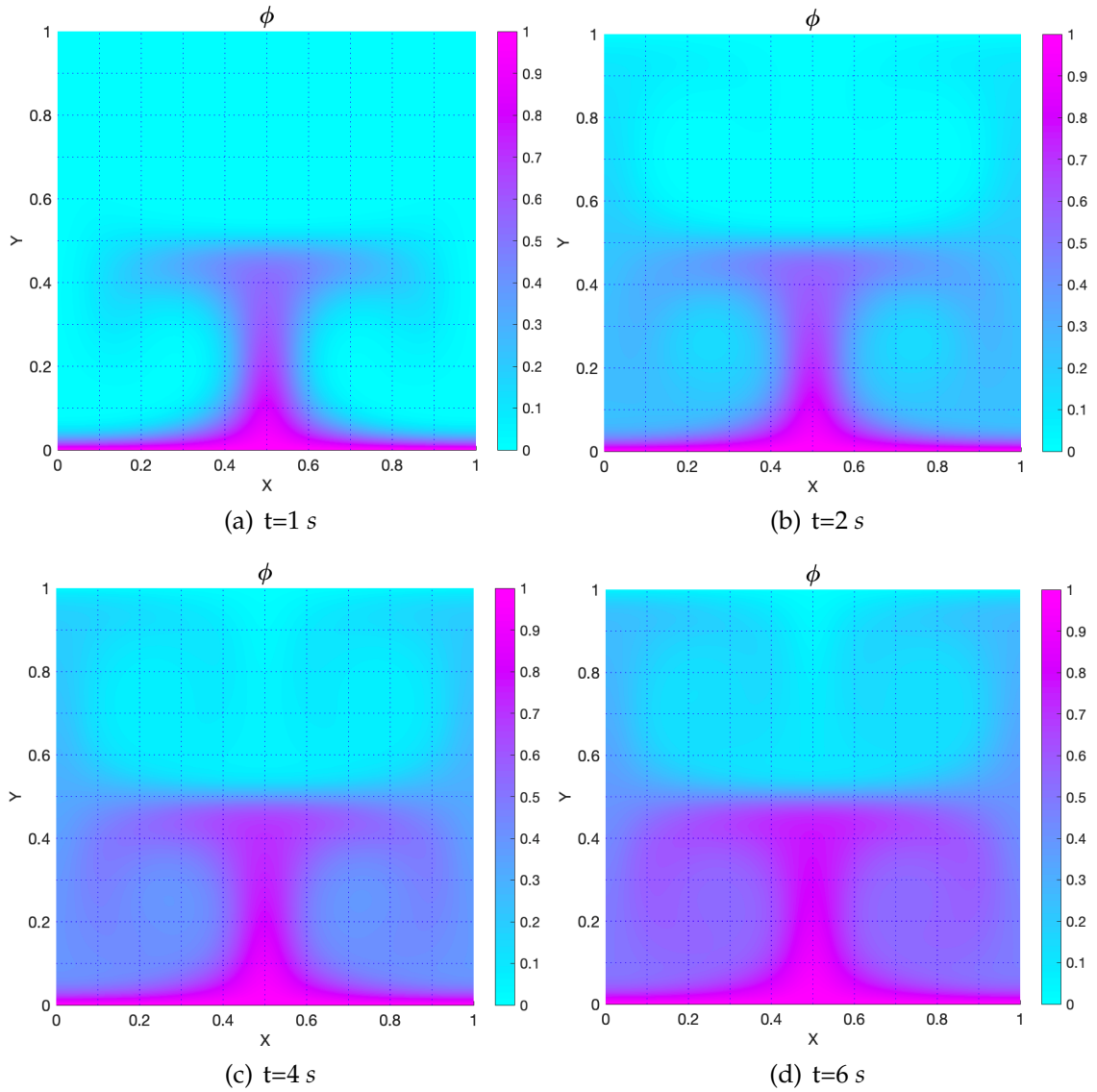
The numerical solutions of (B.5) with velocity field $\mathbf{U}_1$, $\mathbf{U}_1$ and $\mathbf{U}_1$ are represented in Figures B.2, B.3 and B.4 respectively; the mapping of density to colormap was indicated in the colorbar, where pink (blue) indicating high (low) risk density.



(a) t=1 s (b) t=2 s

(c) t=4 s (d) t=6 s

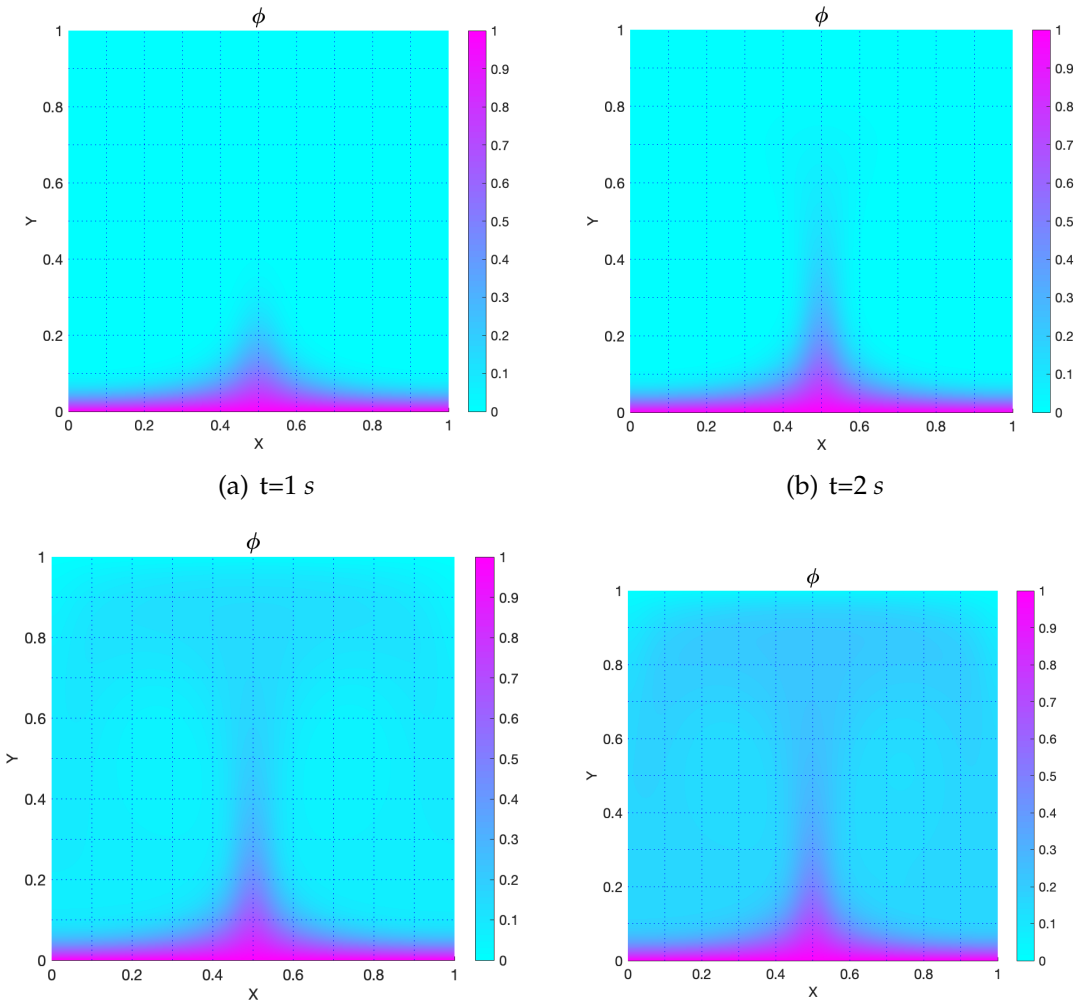
Figure B.2: The evolution of the time-varying risk density governed by the convection-diffusion equation (B.5) with velocity field $\mathbf{U}_1$.



(a) t=1 s (b) t=2 s

(c) t=4 s (d) t=6 s

Figure B.3: The evolution of the time-varying risk density governed by the convection-diffusion equation (B.5) with velocity field $\mathbf{U}_2$.



(a) t=1 s (b) t=2 s

(c) t=4 s (d) t=6 s

Figure B.4: The evolution of the time-varying risk density governed by the convection-diffusion equation (B.5) with velocity field $\mathbf{U}_3$.

Bibliography

[1] Luciano CA Pimenta et al. "Simultaneous coverage and tracking (SCAT) of moving targets with robot networks". In: *Algorithmic foundation of robotics VIII*. Springer, 2009, pp. 85–99.

[2] Maneesha V Ramesh. "Real-time wireless sensor network for landslide detection". In: *2009 Third International Conference on Sensor Technologies and Applications*. IEEE. 2009, pp. 405–409.

[3] Sotiris Papatheodorou et al. "Distributed area coverage control with imprecise robot localization: Simulation and experimental studies". In: *International Journal of Advanced Robotic Systems* 15.5 (2018), p. 1729881418797494.

[4] Minyi Zhong and Christos G Cassandras. "Distributed coverage control and data collection with mobile sensor networks". In: *IEEE Transactions on Automatic Control* 56.10 (2011), pp. 2445–2455.

[5] Paul F Hudak. "Application of facility location theory to groundwater remediation". In: *Applied Geography* 14.3 (1994), pp. 232–244.

[6] Xu Luo and Jun Yang. "Water Pollution Detection Based on Hypothesis Testing in Sensor Networks". In: *Journal of Sensors* 2017 (2017).

[7] NS Vidhya Shree and KN Shreenath. "Location Based Detection Algorithm for Clone Attack in Wireless Sensor Network". In: (2013).

[8] Gabriel Martins Dias et al. "A self-managed architecture for sensor networks based on real time data analysis". In: *2016 Future Technologies Conference (FTC)*. IEEE. 2016, pp. 1297–1299.

[9] Hailin Zhu et al. "Complex networks-based energy-efficient evolution model for wireless sensor networks". In: *Chaos, Solitons & Fractals* 41.4 (2009), pp. 1828–1835.

[10] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[11] Vehbi C Gungor and Gerhard P Hancke. "Industrial wireless sensor networks: Challenges, design principles, and technical approaches". In: *IEEE Transactions on industrial electronics* 56.10 (2009), pp. 4258–4265.

[12] Chee-Yee Chong and Srikanta P Kumar. "Sensor networks: evolution, opportunities, and challenges". In: *Proceedings of the IEEE* 91.8 (2003), pp. 1247–1256.

[13] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. "Wireless sensor network survey". In: *Computer networks* 52.12 (2008), pp. 2292–2330.

[14] Th Arampatzis, John Lygeros, and Stamatis Manesis. "A survey of applications of wireless sensors and wireless sensor networks". In: *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*. IEEE. 2005, pp. 719–724.

[15] Michael Winkler et al. "Theoretical and practical aspects of military wireless sensor networks". In: *Journal of Telecommunications and Information Technology* (2008), pp. 37–45.

[16] Sang Hyuk Lee et al. "Wireless sensor network design for tactical military applications: Remote large-scale environments". In: *MILCOM 2009-2009 IEEE Military communications conference*. IEEE. 2009, pp. 1–7.

[17] Daesung Kim et al. "Performance evaluation of routing protocols for wireless sensor networks in military scenarios". In: *2011 Third International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE. 2011, pp. 101–106.

[18] Robert Hartwell. "Wireless Sensor Network Energy Use While Tracking Secure Area Intrusions". In: *MILCOM 2013-2013 IEEE Military Communications Conference*. IEEE. 2013, pp. 1696–1701.

[19] Gene W Eidson et al. "The south carolina digital watershed: End-to-end support for real-time management of water resources". In: *International Journal of Distributed Sensor Networks* 6.1 (2010), p. 970868.

[20] Hong-bo Xia, Jiang Peng, and Kai-hua Wu. "Design of water environment data monitoring node based on ZigBee technology". In: *2009 International Conference on Computational Intelligence and Software Engineering*. IEEE. 2009, pp. 1–4.

[21] GJ Bishop-Hurley et al. "Virtual fencing applications: Implementing and testing an automated cattle control system". In: *Computers and Electronics in Agriculture* 56.1 (2007), pp. 14–22.

[22] Manijeh Keshtgary and Amene Deljoo. "An efficient wireless sensor network for precision agriculture". In: *Canadian Journal on Multimedia and Wireless Networks* 3.1 (2012), pp. 1–5.

[23] Tian He et al. "Energy-efficient surveillance system using wireless sensor networks". In: *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM. 2004, pp. 270–283.

[24] Alan Mainwaring et al. "Wireless sensor networks for habitat monitoring". In: *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. Acm. 2002, pp. 88–97.

[25] Hande Alemdar and Cem Ersoy. "Wireless sensor networks for healthcare: A survey". In: *Computer networks* 54.15 (2010), pp. 2688–2710.

[26] Rabaey Arens Federspiel et al. "Smart energy distribution and consumption: Information technology as an enabling force". In: *Center for*. Citeseer. 2001.

[27] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem". In: *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[28] Franz Aurenhammer. "Voronoi diagrams—a survey of a fundamental geometric data structure". In: *ACM Computing Surveys (CSUR)* 23.3 (1991), pp. 345–405.

[29] Chih-Hung Liu. "A nearly optimal algorithm for the geodesic voronoi diagram of points in a simple polygon". In: *34th International Symposium on Computational Geometry (SoCG 2018)*. Vol. 99. Schloss Dagstuhl, Leibniz-Zentrum für Informatik. 2018, p. 58.

[30] Adam Dobrin. "A review of properties and variations of Voronoi diagrams". In: *Whitman College* (2005), pp. 1949–3053.

[31] Cecilia Bohler et al. "A randomized divide and conquer algorithm for higher-order abstract Voronoi diagrams". In: *Computational Geometry* 59 (2016), pp. 26–38.

[32] Bomin Jiang, Zhiyong Sun, and Brian DO Anderson. "Higher order Voronoi based mobile coverage control". In: *American Control Conference (ACC), 2015*. IEEE. 2015, pp. 1457–1462.

[33] Der-Tsai Lee. "On k-nearest neighbor Voronoi diagrams in the plane". In: *IEEE transactions on computers* 100.6 (1982), pp. 478–487.

[34] Bernard Chazelle and Herbert Edelsbrunner. "An improved algorithm for constructing kth-order Voronoi diagrams". In: *IEEE Transactions on Computers* 100.11 (1987), pp. 1349–1354.

[35] Henning Meyerhenke. "Constructing higher-order Voronoi diagrams in parallel." In: *EuroCG*. 2005, pp. 123–126.

[36] Pankaj K Agarwal et al. "Constructing levels in arrangements and higher order Voronoi diagrams". In: *SIAM journal on computing* 27.3 (1998), pp. 654–667.

[37] Qiang Du and Desheng Wang. "The optimal centroidal Voronoi tessellations and the gersho's conjecture in the three-dimensional space". In: *Computers & Mathematics with Applications* 49.9-10 (2005), pp. 1355–1373.

[38] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[39] Xiao Xiao. "Over-relaxation Lloyd method for computing centroidal Voronoi tessellations". In: (2010).

[40] Jorge Cortes et al. "Coverage control for mobile sensing networks". In: *IEEE Transactions on robotics and Automation* 20.2 (2004), pp. 243–255.

[41] Cheng Song et al. "Coverage control for heterogeneous mobile sensor networks on a circle". In: *Automatica* 63 (2016), pp. 349–358.

[42] Suruz Miah, Mostafa MH Fallah, and Davide Spinello. "Non-Autonomous Coverage Control With Diffusive Evolving Density". In: *IEEE Transactions on Automatic Control* 62.10 (2017), pp. 5262–5268.

[43] Sung G Lee and Magnus Egerstedt. "Controlled coverage using time-varying density functions". In: *IFAC Proceedings Volumes* 46.27 (2013), pp. 220–226.

[44] Sung G Lee, Yancy Diaz-Mercado, and Magnus Egerstedt. "Multirobot control using time-varying density functions". In: *IEEE Transactions on Robotics* 31.2 (2015), pp. 489–493.

[45] Mostafa Mohammad Hossein Fallah. "Coordinated Deployment of Multiple Autonomous Agents in Area Coverage Problems with Evolving Risk". PhD thesis. Université d'Ottawa/University of Ottawa, 2015.

[46] Donald F Boesch and Nancy N Rabalais. *Long-term environmental effects of offshore oil and gas development*. CRC Press, 1987.

[47] Masamichi Chino et al. "Preliminary estimation of release amounts of 131I and 137Cs accidentally discharged from the Fukushima Daiichi nuclear power plant into the atmosphere". In: *Journal of nuclear science and technology* 48.7 (2011), pp. 1129–1134.

[48] Bernard D Goldstein, Howard J Osofsky, and Maureen Y Lichtveld. "The Gulf oil spill". In: *New England Journal of Medicine* 364.14 (2011), pp. 1334–1348.

[49] Sabri Ghazi, Tarek Khadir, and Julie Dugdale. "Multi-agent based simulation of environmental pollution issues: a review". In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2014, pp. 13–21.

[50] James R Payne. "Weathering of petroleum in the marine environment". In: *Marine Technology Society Journal* 18 (1984), pp. 24–42.

[51] Juan Manuel Corchado, Aitor Mata, and Sara Rodriguez. "Osm: A multi-agent system for modeling and monitoring the evolution of oil slicks in open oceans". In: *Advanced Agent-Based Environmental Management Systems*. Springer, 2009, pp. 91–117.

[52] KR Guruprasad and Debasish Ghose. "Heterogeneous locational optimisation using a generalised Voronoi partition". In: *International Journal of Control* 86.6 (2013), pp. 977–993.

[53] Suruz Miah et al. "Nonuniform coverage control with stochastic intermittent communication". In: *IEEE Transactions on Automatic Control* 60.7 (2015), pp. 1981–1986.

[54] Bomin Jiang et al. "Higher order mobile coverage control with application to localization". In: *arXiv preprint arXiv:1703.02424* (2017).

[55] Suhas Patankar. *Numerical heat transfer and fluid flow*. CRC press, 1980.