Coordinated Deployment of Multiple Autonomous Agents in Area Coverage Problems with Evolving Risk

by

Mostafa M.H.Fallah

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master of Applied Science in Mechanical Engineering degree in
Mechanical Engineering

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

Abstract

Coordinated missions with platoons of autonomous agents are rapidly becoming popular because of technological advances in computing, networking, miniaturization and combination of electromechanical systems. These multi-agents networks coordinate their actions to perform challenging spatially-distributed tasks such as search, survey, exploration, and mapping. Environmental monitoring and locational optimization are among the main applications of the emerging technology of wireless sensor networks where the optimality refers to the assignment of sub-regions to each agent, in such a way that a suitable coverage metric is maximized. Usually the coverage metric encodes a distribution of risk defined on the area, and a measure of the performance of individual robots with respect to points inside the region of interest. The risk density can be used to quantify spatial distributions of risk in the domain.

The solution of the optimal control problem in which the risk measure is not time varying is well known in the literature, with the optimal configuration of the robots given by the centroids of the Voronoi regions forming a centroidal Voronoi tessellation of the area. In other words, when the set of mobile robots converge to the corresponding centroids of the Voronoi tessellation dictated by the coverage metric, the coverage itself is maximized.

In this work, it is considered a time-varying risk density evolving according to a diffusion equation with varying boundary conditions that quantify a time-varying risk on the border of the workspace. Boundary conditions model a time varying flux of external threats coming into the area, averaged over the boundary length, so that rather than considering individual kinematics of incoming threats it is considered an averaged, distributed effect. This approach is similar to the one commonly adopted in continuum physics, in which kinematic descriptors are averaged over spatial domain and suitable continuum fields are introduced to describe their evolution. By adopting a first gradient constitutive relation between the flux and the density, a simple diffusion equation is obtained. Asymptotic convergence and optimality of the non-autonomous system are studied by means of Barbalat's lemma and connections with varying boundary conditions are established. Some criteria on time-varying boundary conditions and evolution are established to guarantee the stabilities of agents' trajectories. A set of numerical simulations illustrate theoretical results.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Davide Spinello for the continuous support of my Master study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

My sincere thanks also goes to Dr. Suruz Miah for supporting me by his systematic advices and guidance. I appreciate his vast knowledge and skill in many areas.

Last but not the least, I would like to thank my family: my wife, Mrs Fatemeh Pourmasouri, my parents Mr. Nasrollah M. H. Fallah and Mrs. Tayebeh Jamei, and my mother in law Mrs. Maryam Banakhojasteh for supporting me spiritually throughout my life.

Dedication

I would like to dedicate my thesis to:

- My beloved wife, Mrs Fatemeh Pourmansouri whose supports and patience allowed me to fulfil my Master's requirements and complete this thesis.
- My family members specially my parents, Mr. Nasrollah M. H. Fallah and Mrs. Tayebeh Jamei who offered me unconditional love and support throughout the course of this thesis.
- My mother in law, Mrs. Maryam Banakhojasteh whose encouragements enthused me to continue my study with more power and hope and to the bright memory of my father in law, Mr. Mohammadreza Pourmansouri, may God bless him.

Table of Contents

List of Figures

Chapter 1

Introduction

Recent advances in micro-electromechanical systems, digital electronics, and wireless communications induced the development of low power, low cost, multifunctional sensor nodes. These sensor nodes, which measure, analyse data, and share information through communication, make the idea of sensor networks real. They are equipped with an on-board processor and instead of sending unprocessed data to stations responsible for data fusion, sensor nodes use their abilities to locally perform the processing and send only partially processed data.

The mentioned features allow innumerable applications for sensor networks. Three important applications of mobile wireless sensor networks are target tracking, task assignment, and coverage control. The deployment of large group of autonomous agents in an environment to perform challenging and spatially-distributed missions such as exploration, search, survey, and mapping is called coverage control.

1.1 Motivation

Area coverage control by deployment of mobile wireless sensor network has many applications in a vast class of problems that include perimeter surveillance, search and rescue missions, and protection of high value units located in sensitive areas. For instance, in the harbour protection scenario, it is natural to protect a high-valued unit by employing a group of mobile agents where asymmetric threats may penetrate through the boundaries of the protected areas, with time varying boundary conditions that describe, for example, the risk associated to the penetration of threats through a given portion of the boundary. A variety of other applications, such as social foraging by a group animals, may exist where the non-uniform coverage metric has a significant impact on the boundary density of resources or risks.

Relatively little work has been done for the case when the risk density function is time-varying. For instance Cortes et al.(2004) presents an algorithm for optimal coverage of time-varying density functions [7]. They designed distributed gradient descent algorithms

for a class of utility functions which encodes optimal coverage and sensing policies. Pimenta et al.(2009) introduces the problem of simultaneously covering an environment and tracking targets [25]. This problem is translated to the task of environmental coverage with evolving density functions under the locational optimization framework. Lekien and Leonard (2010), study non-uniform coverage of a region by a network of mobile agents. They focus on time-varying coverage metrics and the design of an appropriate control algorithms to cover domains with non-uniform slowly varying metrics [15]. Lee et al.(2010) presents an approach to controlling a multi-agent system by choosing a time-varying risk density function. The intention is to control a system of multiple agents to provide coverage over the workspace by employing optimal coverage ideas on general time-varying density functions [14]. Miah and Spinello (2014) investigate the non-uniform coverage of a planar region by a platoon of autonomous mobile agents when communications among them are stochastically intermittent [21]. In the real world, many phenomena can be modelled by time-varying risk density rather than time-invariant one; hence there has been an increasing attention to the cases with time-varying risk density specially in coverage control problems. Motivated by mentioned researches in this field, we take a time-varying risk density evolving as dictated by a diffusion equation with time-varying boundary conditions in the consideration.

1.2 Thesis Objectives

This work contributes to the development of a non-uniform coverage control algorithm with a group of autonomous agents, in which the evolution of the risk density obeys a conservation law in a spatial workspace where mobile agents are deployed. The following items are objectives of this thesis:

1. The risk density is considered with evolving boundary conditions which model interaction with external threats coming into the domain, averaged over the boundary length. It means that instead of considering the individual kinematics of incoming threats, their averaged and distributed effect across the boundary is assumed. This approach is analogous the one usually-adopted in continuum mechanics, in which kinematic descriptors are averaged over the spatial domain and appropriate continuum fields are introduced to describe their evolution. By adopting a first gradient constitutive relation between the risk flux and the risk, a simple diffusion equation is obtained which can be solved numerically,

2. A non-autonomous feedback law whose generated trajectories maximize the coverage metric is suggested,

3. Asymptotic stability of agents trajectories is established by using Barbalat's lemma. A criterion on time-varying boundary conditions is established to guarantee the stabilities of agents' trajectories, and

4. Several scenarios with different time-varying boundary conditions are simulated and accuracy of the proposed feedback law is verified.

1.3 Thesis Outline

In chapter 2, we review the literature pertaining wireless sensor networks and go through the sensor architecture and communication methods in wireless sensor networks. Common applications of spatial mobile wireless sensor networks are discussed and basic techniques used in area coverage maximization are introduced.

In chapter 3 we present the thesis research project, whose core contribution is the study of a non-autonomous coverage control problem in which the risk map of the environment evolve according to a conservation law. Under the assumption that there are no risk sources inside the domain, evolution of the risk is associated with time varying boundary conditions that model distributed incursions of external threats through the boundary of the region to be covered. As opposed to the approach commonly used in the literature, the risk map is not assumed a priori but it is rather characterized by a diffusive evolution that is a consequence of basic principles. Then we propose a feedback law that maximizes the coverage metric and in order to guarantee the stability of generated trajectories of agents and by utilizing the divergence theorem to transform the domain integral into a boundary integral, a constraint on the time-varying boundary conditions is established. Finally, we investigate several scenarios with different time-varying boundary conditions.

Chapter 4 is dedicated to summary and conclusions.

Chapter 2

Literature Review

2.1 Wireless Sensor Networks

Wireless sensor networks can simply be defined as sensor networks in which the nodes can communicate wireless. Mobile wireless sensor network is much more versatile than static networks as they can be deployed in scenarios with rapid network and environment topology changes. However, they have many similar applications, such as environmental surveillance or monitoring. Usually the nodes consist of a microcontroller and a wireless transceiver powered by a battery, and also some kind of smart sensors with limited processing and computing resources [1]. A variety of thermal, mechanical, chemical, biological, magnetic, and optical sensors may be attached to the sensor nodes to measure and quantify properties of the environment.

The concept of wireless sensor networks is introduced in detail in [1], among others, and several applications and sensing tasks are discussed. A review of factors influencing the design of sensor networks is performed, and communication architectures for sensor networks are described. Additionally, protocols and algorithms developed for each layer are explored.

A comprehensive literature survey on various aspects of wireless sensor networks can be found in [31]. Challenges associated to wireless sensor networks are classified into three different categories:

1. underlying operating system and internal platform,
2. communication protocol stack, and
3. network provisioning, services, and deployment.

One of the most important constraints on sensor nodes is the low power consumption requirement. Sensor nodes carry limited, generally irreplaceable, power sources [1]. Battery is the most important power source in sensor node. It may be equipped with secondary power supply which harvests energy from the environment like solar panels depending on the environment in which the sensor will be deployed.[31]

2.2 Applications of Wireless Sensor Networks

Wireless sensor networks can be used for several applications, for instance space exploration, military surveillance, rescue and search, and cooperative classification. An overview of wireless sensor networks and their applications is given in Fig. 2.1 from [31]. They classify the wireless sensor network applications into two main categories: monitoring and tracking and also enumerate some subdivisions and examples of each one.

In our work, the applications of wireless sensor network are divided into three main categories:

1. Target Tracking,
2. Task Assignment, and
3. Coverage Control.

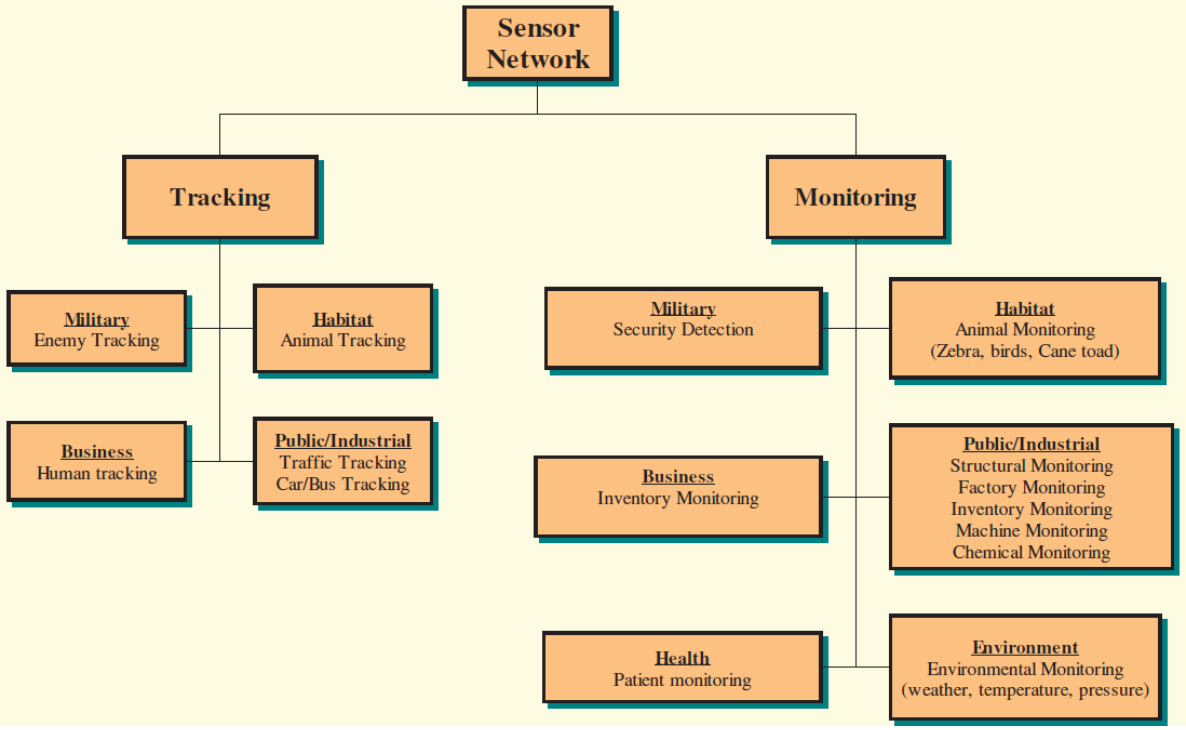


Figure 2.1: Overview of wireless sensor network applications.[31]

2.2.1 Target Tracking

Target tracking is the capacity to detect and continuously track the state of a target or a set of targets. Tracking is usually stated as an estimation problem based on a series of measurements: the main goal is to estimate the targets' state and update the estimation

with measurements. Modern surveillance systems usually utilize several sensors of different types to provide overlapping coverage on targets. In order to estimate targets' state, the received data need to be combined.

Different distributed data fusion architectures are discussed in [16], in which each node processes the data from its own sensors and communicates with others to improve and update final estimations. Fusion procedure for target tracking includes two important operations: (I) estimation and (II) association. Authors present a distributed algorithm for several arbitrary fusion architectures and demonstrate their relationship to linear\non-linear distributed estimation results. Distributed versions of two common tracking approaches (multiple hypothesis and joint probabilistic data association tracking) are presented, and some examples of the implementations are given.

In [4], authors consider the problem of dynamic sensing utilizing portable sensor nodes that together estimate the state of a mobile target as a sensor network. They suggest a decentralized gradient search-based algorithm that presents the advantages of distributed sensing and then examine the task of tracking a set of targets. A greedy algorithm is used in order to associate sensors with targets along with the gradient search-based strategy for sensor vehicle motion planning.

Martinez and Bullo in [20] study locational optimization of sensors and motion coordination strategies for portable sensor nodes. They study the determinant of the Fisher information matrix and calculate it in both 2D and 3D cases, characterizing the global minima in the 2D case, for a target-tracking applications with range sensors. Afterwards, they suggest motion coordination algorithms that guide the mobile networked sensors to an optimal distribution and that are amenable to a decentralized implementation. At the end, their numerical results illustrate how the suggested algorithms lead to enhanced performance of an extended Kalman filter in a target tracking scenarios.

In [19], authors study the problem of target tracking by a group of ground vehicles. Control laws for cooperative target tracking are presented that achieve tracking of a mobile target with known state (position, velocity, and acceleration), for both single- and double-integrator robot models. When the information of target's motion is unknown constant, vision-based assessment schemes are applied to acquire estimates of the target's state. The efficiency of the suggested control laws and their vision-based counterparts is presented by numerical simulation examples using non-holonomic robots in order to obtain desired formations.

Giving the limited processing capability and power in sensor nodes, a critical step in target tracking is how to obtain suitable data and accomplish information processing at the local level using cooperative communication and networking around the target. Therefore, the need to save energy and scalability lead to organizing the sensors hierarchically, which can represent the target's state and incorporate several statistical models for the target positioning. In [3], authors try to develop a fully distributed technique for cooperative target tracking in wireless sensor networks from two viewpoints: (I) energy-balanced tracking and (II) improved estimation accuracy.

The first viewpoint is to build up an energy-balanced architecture for tracking network. They conduct the idea of leader-based information processing to obtain cooperative sensor

scheduling with several tasking sensors in a cluster-based network topology designed based on target information, estimation quality, and sensor residual energy level. It is necessary to mention that the cluster members and the cluster-head refer to the original network topology, whereas the sub-cluster members and the leader of sub-cluster refer to the sensor group assigned for the tracking task. Local criteria and random timers are used to specify the tracking responsibility of the existing clusters. Afterwards, a leader which can be a cluster-head or a cluster member in the original network, forms a sub-cluster of the corresponding cluster for the tracking procedure.

The second viewpoint is to explore the characteristics and behaviours of a target so that supplementary information can be used to enhance estimation accuracy. First, the sensor nodes provide their measurements to the leader within the sub-cluster. After receiving the measurements, the leader fuses the obtained estimates and reports it to the cluster-head. If the tracked target leaves the region of the active sub-cluster, leader hand-off procedure needs to be triggered.

Authors in [3] present the fully distributed cooperative target tracking system: Two-level clustering approach via timer in a cluster-based network topology that organizes the tracking mission in four phases: (I) choosing the leader, (II) sub-cluster members selection, (III) estimating target's position, and (IV) leader hand-off and reselection of sub-cluster members. Hence, the suggested tracking approach organizes a sub-cluster to perform target tracking task, allows each member of sub-cluster to locally estimate the target's position and uses cooperation to acquire the fused estimation in the leader node. The described tracking architecture is shown in Fig. 2.2.

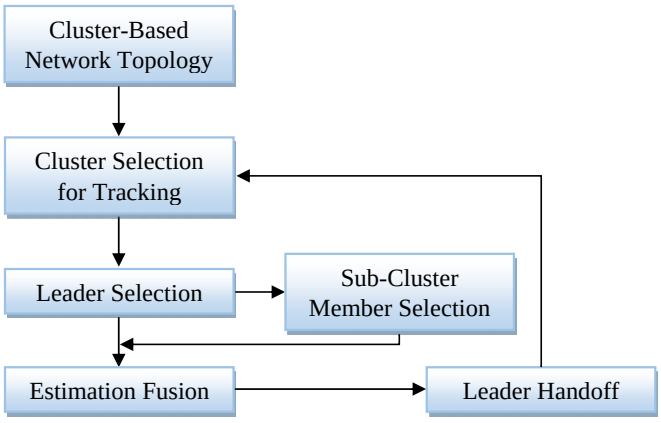


Figure 2.2: Block diagram for the two-level clustering approach via timer method.[3]

Authors in [29] concentrate on the problem of mobile-target tracking with a group of networked robots. Each robot in the network has its own pan/tilt camera to detect the target and has restricted communication capability to communicate with its neighbour robots. Authors solve mentioned problem by separating it into two different parts. One part is the target's position estimation and another is the control of networked robots moving toward the detected target. In the first part, they propose to use a novel distributed Kalman filter to estimate the target's position. The distributed Kalman filter is deduced based on a

standard Kalman filter by modelling the neighbour's information as one of measurements. They develop a distributed flocking algorithm to track the estimated position and avoid collision in the motion control part. In both sections, only local communication between neighbour robots is necessary. At the end, the tracking algorithms are presented with 2D and 3D robots to confirm their performance. A set of real ground robots is used to test the suggested algorithm. The experimental results demonstrate that multiple robots are able to work cooperatively to track the target under the suggested algorithms and the tracking results outperform the result produced by individual robots with no cooperation.

Because of the sensors limited battery resources, there is a trade-off between the tracking accuracy and energy consumption. To solve this problem, authors in [32] propose an energy efficient tracking algorithm. Sensors in the workspace are scheduled to switch their working mode to track the target based on the cooperation of dispatchers. Because energy consumption in active mode is more than that in sleeping or monitoring mode, for each sampling period, a minimum set of sensors is woken up based on the select mechanism and other sensors keep in sleeping mode. Simulation results show that the suggested algorithm provides a better performance than other existing approaches.

2.2.2 Task Assignment

In the current publications, there are also some examples of assigning miscellaneous tasks to wireless sensor networks. For instance, Belta and Kumar in [2] address the general problem of controlling a set of robots required to move together as a group. They propose an abstraction based on the definition of a map from the configuration space $\mathcal{Q}$ of the robots to a manifold $\mathcal{A}$ with lower dimension which is completely independent of the number of existing robots. In this paper, authors concentrate on planar fully actuated robots. They require that the considered manifold has a product structure $\mathcal{A} = \mathcal{S} \times \mathcal{G}$, in which $\mathcal{S}$ is a shape and form manifold, which is an inherent characterization of the group describing the "shape" formed by distributed mobile robots and $\mathcal{G}$ is a Lie group, that captures the orientation and position of the group in the selected world-coordinate frame. They propose decoupled controllers for the mentioned shape and group variables and obtain controllers for individual robots that guarantee the desired behaviour of them on manifold. Each controller can be realized by feedback which depends on the state of manifold $\mathcal{A}$ and the current state of the robot. This has the practical advantage of limiting the complexity of each robot controller and reducing the required sensing and communication even for large group of robots. Some snapshots from the produced motion with different control laws and different manifolds are shown in Fig. 2.3 and Fig. 2.4.

Motion can be used in wireless sensor networks to change the configuration and improve the sensing performance. Authors in [13] assume the problem of distributed controlling motion for a mobile wireless sensor network for a specific form of motion capability. An architecture that allows each sensor node in the network to learn the phenomenon and medium characteristics is presented. Since the sensing performance affects most sensor network applications directly, it is important to collect data at the highest fidelity possible within the given resource restrictions. Motion can help us to achieve this goal in at least following three ways:

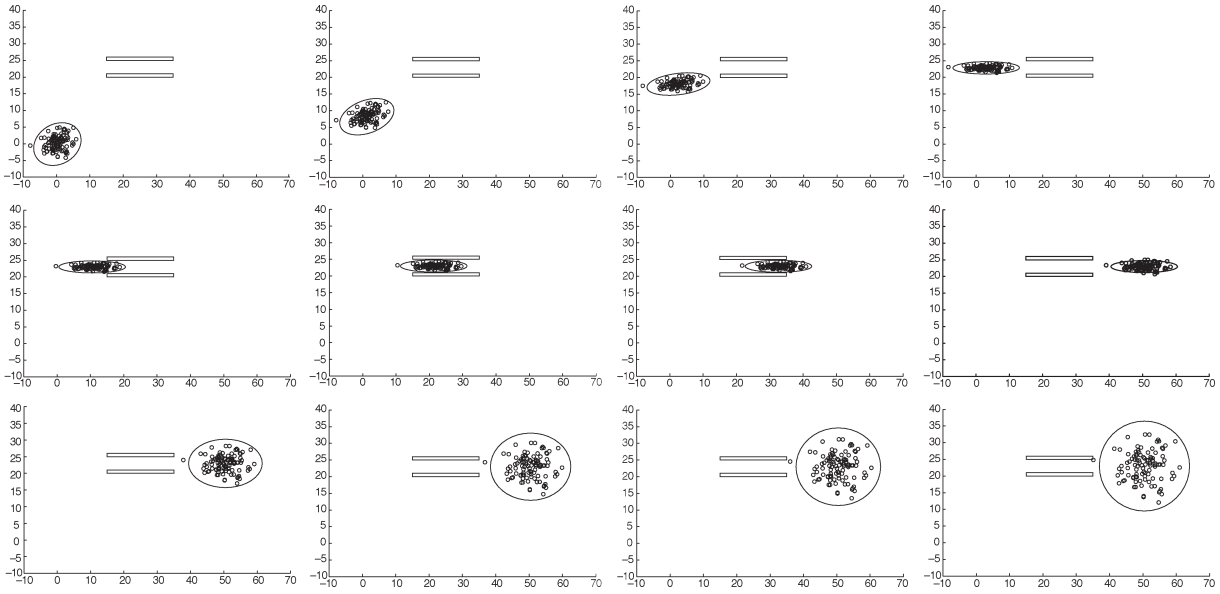


Figure 2.3: Ninety-nine of 100 normally distributed planar robots travel through a considered tunnel by designing 5D controls for the corresponding equiprobability ellipse.[2]

1. **Phenomenon Adaptivity:** Mobile sensor nodes can converge sample the phenomenon accurately where coverage with highest resolution is necessary.

2. **Medium Adaptivity:** Mobile sensor nodes can reconfiguration themselves to overcome existing anisotropies and obstacles.

3. **Increased Sensor Range:** Mobile sensor nodes can cover a larger space, depending on the acceptable motion delay.

The sensing advantages are more important than the cost of motion for suitable system design choices. Authors in [13] describe a quantitative metric for measuring sensors' efficiency that is concretely tied to real medium and sensor specifications, rather than considering an abstract model. In order to determine the optimal network configuration, they try to optimize this metric. Authors first propose a distributed optimization algorithm that determines an appropriate network configuration, and then adapt it to environmental evolutions. Afterwards, the relationship of the presented algorithm to incremental sub-gradient descent based methods and simulated annealing is investigated. A key characteristic of their algorithm is that although no global coordination is included, convergence to a desirable configuration can be proved.

Another example of task assignment is presented in [30] in which a framework for design of collective behaviours for groups of identical mobile robots is presented. The approach is established upon decentralized estimation and control at the same time, where each robot communicates with its neighbours and estimates the global performance specifications of the swarm required to make a local control decision. Challenges of mentioned approach include designing an estimator which allows each robot to make correct estimates of the

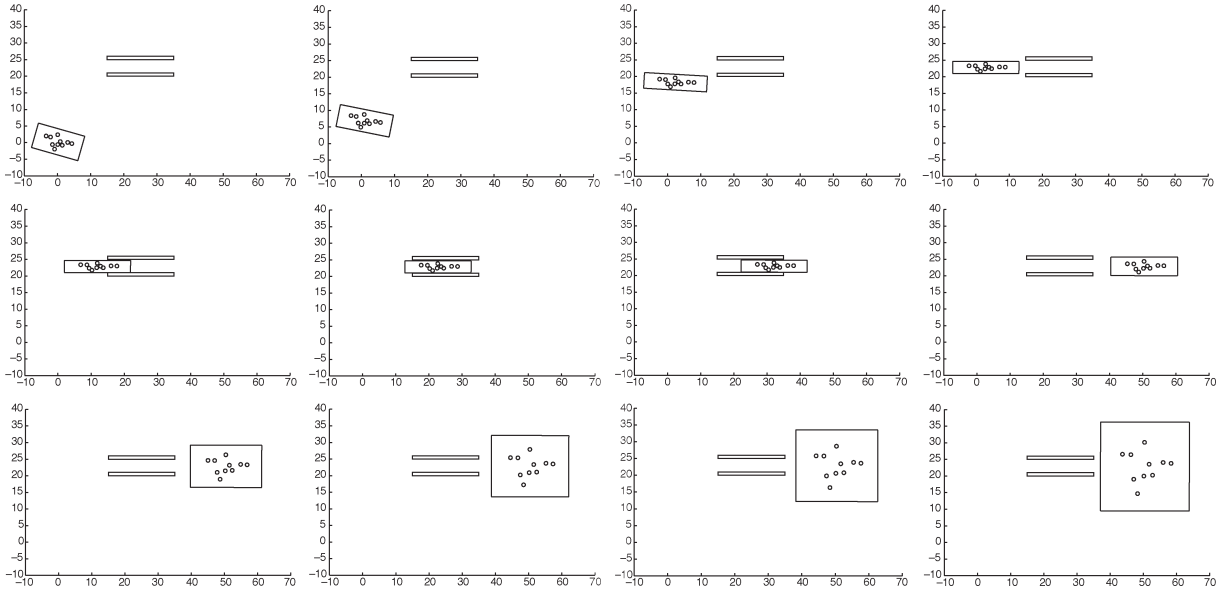


Figure 2.4: 10 planar robots pass through a considered tunnel by designing 5D controls for the corresponding rectangle.[2]

global specifications required to implement the controller, designing a control law with appropriate convergence properties, considering each robot has perfect global knowledge, and possibly improving the controller to recover desired convergence specifications when using the estimates of global performance. They utilize mentioned framework to the problem of controlling the moment statistics describing the shape and location of a swarm and specify conditions which guarantee that the formation statistics are driven to ideal values, even in the presence of a evolving network topology. Authors also illustrate their work with some examples. In the first example presented in Fig. 2.5, the left picture shows the initial configuration of a swarm, a uniform-density ellipse with the same mass and first- and second-order moments as the swarm, and the goal formation of the swarm represented as a light grey uniform-density ellipse. The right picture represents the swarm which converges to a configuration having the desired first- and second-order moment statistics.

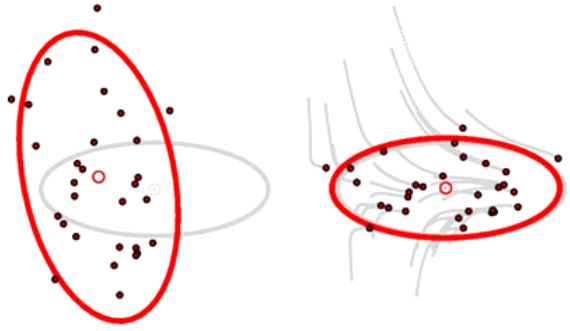


Figure 2.5: The swarm's configurations before and after convergence.[30]

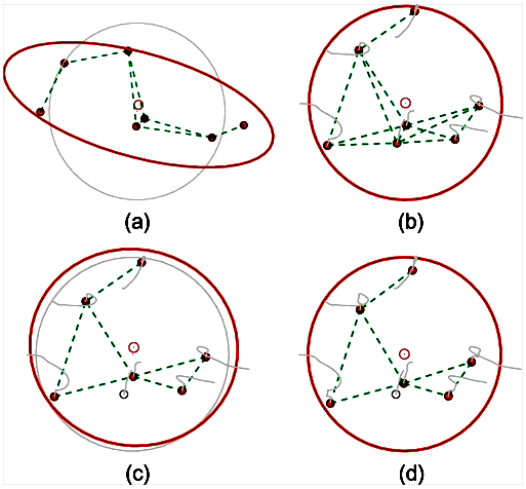


Figure 2.6: Snapshots of the robots' movement.[30]

Fig. 2.6 which belongs to second example, shows snapshots of the robots' movement under the Proportional-Integral scheme: (a) $t = 0$ initial condition, (b) $t = 25$ robots satisfy the goal, (c) $t = 25$ one robot dies, (d) $t = 45$ robots move to re-satisfy the goal.

In [28], authors present an algorithm to estimate and reconstruct the border of a domain. The goal is for a group of mobile robots to optimally place several interpolation points on the border of a connected planar domain. The border is then rebuilt by linear interpolation of the nominated points. They consider that

(i) at the beginning the robots have an estimate of the domain's border,

(ii) each robot is equipped with a limited-range camera-like sensor and with algorithms to estimate the curvature and tangent of the border, and

(iii) the robots exchange information and interact through a ring-topology communication network.

Authors reach to their goals in two steps. First and foremost, they suggest a criterion to optimally find interpolation points to rebuild a planar border. The criterion needs that the mentioned interpolation points are uniformly distributed on the border according to a curvature-weighted distance function assumed along the border. Second, they demonstrate a provably convergent algorithm to rebuild a planar border by combining a data structure created by the mobile robots with the optimal distribution for the interpolation points. The proposed algorithm has the following two specifications:

(i) it is efficient for slowly-moving borders and provably convergent for fixed one,

(ii) it leads the interpolation points to a optimal distribution along the border.

As some examples of perimeter surveillance with group of robots, we can mention [24] and [33]. In [24], it is presented a decentralized controller to lead a group of flying robots to converge to a simple closed trajectory and to move along it specified in a 3D space. This trajectory may be assumed as a perimeter to be followed and surveilled by the robots. The solution proposed in [24] is based on an artificial vector field relies on local sensing and modulated by a collision avoidance scheme. Authors devise proofs of asymptotic stability of the presented controller for a group of kinematically controlled rotorcrafts. In [33], a hybrid system of finite states is presented for an autonomous multi-robot system in order to detect and track hazardous spill perimeter. In the system, each agent is considered to be in one of following states: searching, pursuing, and tracking. The robots are prioritised based on their states, and in each state, a potential field is build for agents. For each robot in the tracking state, its own state as well as states of its nearest leading and trailing robots are used to control its motion. The convergence of the final tracking algorithm is investigated for several spills under different conditions.

In this work, the main focus is on the area coverage control by employing a mobile wireless sensor network. A comprehensive review of the publications with coverage control theme follows.

2.2.3 Coverage Control

The deployment of large groups of autonomous agents in an environment is rapidly becoming possible due to technological advances in sensing, networking, and miniaturization of electromechanical systems. Environmental coverage is one of the most popular applications of the emerging technology of wireless sensor networks.

Authors in [27], present a distributed algorithm to estimate the evolution of a scalar field (such as atmospheric pressure, temperature, intensity of light, etc.) employing a random wireless sensor network. They mention some potential applications of their algorithm in preventing forest fires, energy conservation, oceanography, and building science.

In [12], authors describe a potential-field-based approach, in which sensor nodes are considered as virtual particles, subject to virtual forces. Virtual forces repel the sensor nodes from obstacles and from each other, and ensure that an initial, compact configuration of sensor nodes will rapidly spread out to maximize the area covered by the network (it is necessary to mention that nowhere do they reason about coverage clearly; rather, coverage is an emergent specification of the algorithm). On the other side, sensor nodes are also subject to viscous friction forces. These forces are used to ensure that the sensor nodes will eventually reach static equilibrium; i.e., all nodes will finally come to zero velocities and a complete stop. These viscous forces do not prevent the network from reacting to evolutions in the environment. The sensor nodes will automatically reconfigure themselves for the modified environment before return once again to zero velocities and a static equilibrium if something changes. Therefore, sensor nodes move only when it is essential to do so, saving a great amount of energy. The potential-field-based approach described in this paper relies on unique consideration: that each node is equipped with a sensor that enables it to determine the bearing and range of both nearby obstacles and nodes (appropriate sensors

can be built using omni-camera or scanning laser range-finder). Using this information, the sensor node can calculate the virtual forces acting it, and convert this data into a control feedback law to be sent to its actuators. No other information is necessary. It should be mentioned that this approach does not need models of the existing environment, localization, or communication between sensor nodes.

Ogren and his colleagues in [22] present a stable control strategy for sets of networked agents to move and reconfigure cooperatively in response to a sensed environment. Each agent serves as a mobile sensor node and the whole network as a reconfigurable and portable sensor array. Their control strategy separates, in part, the network manoeuvres from the cooperative management of the network formation. The underlying coordination framework used virtual bodies and artificial potentials. They concentrate on gradient climbing tasks in which the mobile wireless sensor network looks for local minima or maxima in the workspace. The networked agents can adapt their configurations in response to the sensed environment and in order to optimize their gradient climb.

In [6], Cortes, Martinez and Bullo present coordination algorithms for sets of mobile vehicles performing deployment and coverage missions. As an important modelling criterion, they consider that each mobile vehicle has a limited sensing and communication range. Based on proximity graphs and the geometry of Voronoi partitions, they analyse a class of aggregate objective functions and suggest coverage algorithms in discrete and continuous time. It can be guaranteed that these algorithms have convergence and they are spatially distributed regarding appropriate proximity graphs. In their previous work, they investigated distributed algorithms for optimal coverage problem using tools from computational geometry, non-smooth analysis and geometric optimization [5]. They considered n generators $(p_1, \cdots, p_n)$ changing within a convex area Q according to one of the following interaction laws:

(i) each generator moves away from the closest other generator or region boundary,

(ii) each generator moves toward the farthest vertex of its own Voronoi partition, or

(iii) each generator moves toward a geometric centre (circumcentre, incentre, centroid, etc.) of its own Voronoi partition.

The resulting dynamical systems promised to be of use in coordination problems for networked robots; in that setting the distributed control laws correspond to local interactions between the robots.

Simultaneous tracking and formation control is addressed in [26], for a group of autonomous robots which evolve dynamically inside a domain containing a measurable vector field. Each robot measures the local amount of the field along its trajectory and sometimes shares related information with other robots, in order to obtain the spatial average achieved from averaging measurements across all robots. Using shared information, each robot control its trajectory in a cooperative manner with other robots, with the dual goals of maintaining a desired arrangement about the average and driving the mean field measurement to a specified amount. Two different approaches to virtual leader estimation are

assumed. The first includes the synthesis of a unique virtual leader state, whereas the second includes decentralized estimation of the virtual leader by each agent. Under the second one, control is posed as a two-level problem, where robots reach agreement on the leader state at first level and reach formation about the leader at the second level. The decentralized approach is efficient even when communication among robots is restricted, which means that the associated network graph may be disconnected in frozen time.

Non-uniform coverage of an area by a networked mobile autonomous agents has been studied in [15]. Authors propose centralized non-uniform coverage control feedback laws from uniform coverage algorithms utilizing cartograms[1]. They study time-varying coverage metrics and also the design of control algorithms to cover domains with slowly varying, non-uniform metrics. Their final results can be applied to the design of mobile wireless sensor networks, especially when the coverage metric evolves as data is gathered.[2]

2.3 An Introduction to Coverage Control Models

Voronoi partitions emerge as the solution of the coverage problem and in many publications in the field of coverage control, authors employed a Voronoi diagram to partition the workspace optimally. Therefore, it sounds necessary to explain the concept of Voronoi diagram comprehensively.

Voronoi Diagram

a Voronoi diagram is a partitioning of a domain into regions based on distance to points in a specific subset of the domain. The set of points (called seeds, sites, generators or in this thesis agents) is specified beforehand, and for each agent there is a corresponding section consisting of a set of points closer to that agent than to any other. The mentioned regions are called Voronoi cells or Voronoi partitions. It is named after Georgy Voronoy- an Ukrainian mathematician, and is also called a Voronoi decomposition, or a Voronoi tessellation. Voronoi tessellation has theoretical and practical applications to a large number of fields of science and technology and even visual art and archaeology.

Formal Definition: Let Ω be a space (a non-empty set) endowed with a distance function d. Let K be a set of indices and let $(\mathbf{p}_k)_{k\in K}$ be a tuple (ordered collection) of non-empty subsets (the sites or generators) in the space Ω. The Voronoi region, or Voronoi cell, R_k, associated with the generator $\mathbf{p}_k$ is the set of points in Ω whose distances to $\mathbf{p}_k$ are not greater than their distances to the other generators $\mathbf{p}_j$, where j is any index different from k. In other words, if $d(\mathbf{x}, A)$ denotes the distance between the subset A and the point $\mathbf{x}$, then

$$R_k = \{\mathbf{x} \in \Omega \mid d(\mathbf{x}, \mathbf{p}_k) \leq d(\mathbf{x}, \mathbf{p}_j), \forall j \neq k\} \tag{2.1}$$

[1] Transformation that maps a non-uniform metric to a near Euclidean metric.

[2] We will investigate their work in section 2.3.2 with more detail.

The Voronoi tessellation is simply the tuple of cells $(R_k)_{k\in K}$. In principle some of the generators can intersect and even coincide, but normally they are considered to be disjoint. In addition, infinitely many agents are allowed in the definition (which has applications in crystallography and geometry of numbers), but again, in many cases only finite generators are assumed.

In the simplest and most familiar case which is the usual Euclidean space, we are given a finite set of points $\{\mathbf{p}_1, \cdots, \mathbf{p}_n\}$. In this case each generator is a point and its Voronoi partition R_k consist of all points whose distances to $\mathbf{p}_k$ are less than or equal to their distances to any other generator. Each partition is convex polygon because it is obtained from the intersection of half-spaces. The common border of two adjacent Voronoi cells is the bisector of the line-segment whose ends are their generators and is the locus of all the points in the workspace that are equidistant to the two closest generators. The Voronoi vertices are also the points equidistant to three (or more) agents. (Fig.2.7)

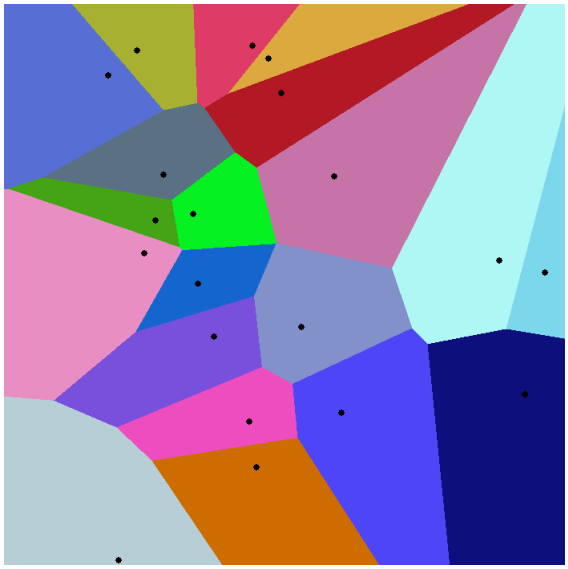


Figure 2.7: A Voronoi diagram based on Euclidean distance.

Illustration: As an illustration, assume a group of restaurants in a flat city. Suppose that we want to estimate the approximate number of customers of a given restaurant. With all other factors being equal (menu, price, quality of service, etc.), it sounds reasonable to consider that customers choose their preferred restaurant simply by distance assumptions: they will select the restaurant located closest to them. In this case the Voronoi partition of a given restaurant can be employed for obtaining a rough estimate on the number of customers going to mentioned restaurant modelled by a point in our flat city. So far it was considered that the distance between two different points in this city is measured using the familiar Euclidean distance:

$$l_1 = d[(a_1, a_2), (b_1, b_2)] = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \tag{2.2}$$

However, if we consider the case in which customers only go to the restaurants using a vehicle and the traffic paths are parallel to the reference axes (x and y), as in Manhattan, then a more practical and realistic distance function will be

$$l_2 = d[(a_1, a_2), (b_1, b_2)] = |a_1 - b_1| + |a_2 - b_2| \tag{2.3}$$

There are different algorithms for drawing Voronoi diagrams, e.g. Discretization Method, Incremental Method, Divide and Conquer Method and Fortune's Method which can be chosen with respect to the required accuracy and also available time and processor.

Centroidal Voronoi Tessellations

A centroidal Voronoi tessellation is a particular type of Voronoi diagram in which the generating point of each Voronoi partition is also its centroid (centre of mass). It can be considered as an optimal partition corresponding to an optimal distribution of cells' generators. A centroidal Voronoi tessellation in a spatial domain is not unique and completely depends on the initial configuration of generators and distance metric, d. For example, ten different centroidal Voronoi tessellations of a square are shown in Fig. 2.8.

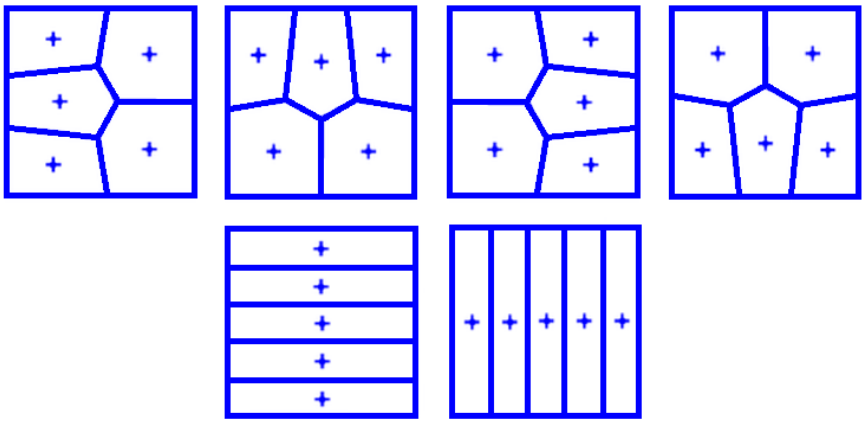
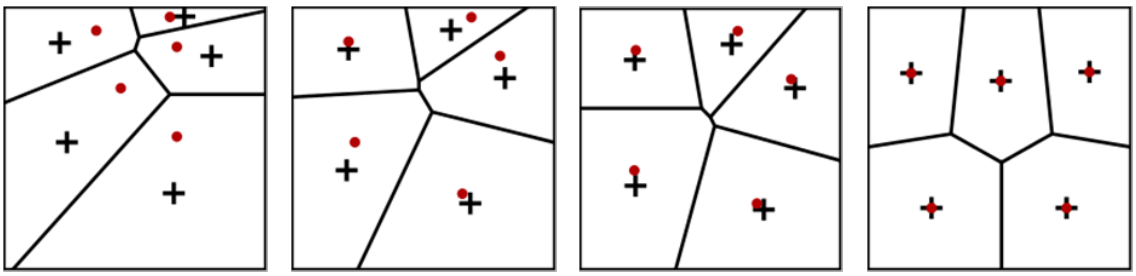


Figure 2.8: Ten different centroidal Voronoi tessellations of a square.

A number of methods including Lloyd's algorithm can be utilized to generate centroidal Voronoi tessellations [18]. This algorithm named after Stuart P. Lloyd starts by an arbitrary placement of several generators in the workspace. It then repeatedly perform the following relaxation step:

- The Voronoi tessellation of the existing sites is calculated.
- Each partition of the Voronoi tessellation is integrated and the corresponding centroid is obtained.
- Each generator is moved to the centroid of its Voronoi partition.



(a) First iteration (b) Second iteration (c) Third iteration (d) Fifth iteration

Figure 2.9: An example of Lloyd's algorithm.

An example of Lloyd's algorithm is shown in Fig 2.9. After each relaxation step, the points are located in a more even distribution: widely spaced generators move closer together, and closely spaced generators move farther apart. It has been shown that this algorithm converges to a centroidal Voronoi tessellation in one dimension [9]. However, slightly weaker convergence results are known in higher dimensions [10]. This algorithm converges slowly or, because of limitations in numerical precision, may not converge at all. So, real-world implementations of Lloyd's algorithm typically stop once the distribution of generators is "good enough". One usual termination criterion is to stop procedure when the maximum distance moved by any generator in each iteration falls below a pre-set threshold. Convergence procedure can be accelerated by over-relaxing the generators, which is done by moving each generator, κ times the distance to the centroid (typically less than 2).

In [9], several new analytical results on the global and local convergence of the Lloyd algorithm are demonstrated. Mentioned results are obtained through careful employment of the optimization properties shared by centroidal Voronoi tessellations.

Authors in [8] assume a direct application of the Newton method and suggest a coupled Lloyd-Newton scheme which can be employed to accelerate the convergence procedure of the original method. Both computational simulations and theoretical analysis are conducted. Rigorous convergence results are demonstrated and significant speed-up in calculation is presented through the comparison with previous methods.

In [17], Liu and his co-authors develop new efficient methods for centroidal Voronoi tessellation computation and present the accelerated convergence of these methods. They propose a quasi-Newton method to obtain centroidal Voronoi tessellation and present its faster convergence than traditional Lloyds method with various numerical examples. Proposed method is also significantly faster and more robust in comparison with the Lloyd-Newton method, represented in [8].

consider n mobile agents equipped with multiple sensors in a bounded domain represented by $\Omega \subset \Re^N$, and the location of the ith agent is given by $\mathbf{x}_i \in \Omega$. Then consider $\mathbf{x} = \{\mathbf{x}_1^\mathsf{T}, \cdots, \mathbf{x}_n^\mathsf{T}\}^\mathsf{T} \in \Omega$ being the vector representing the configuration of the mobile-

agents network. Then we can define the following coverage metric [25]:

$$\mathcal{H}(\mathbf{x}) = \sum_{i=1}^{n} \mathcal{H}_i(\mathbf{x}_i) = \sum_{i=1}^{n} \int_{\Omega_i} \max_i f(r_i)\phi(\mathbf{q})d\mathbf{q}, \tag{2.4}$$

where $r_i : \Re^N \times \Re^N \to \Re^+$ represents the distance between $\mathbf{x}_i$ and any point in Ω_i (where $\Omega = \cup_{i=1}^{n}\Omega_i$) or $r_i = \| \mathbf{x}_i - \mathbf{q} \|$, the function $f(r_i) : \Re \to \Re$ is a continuously differentiable, strictly decreasing function over the range of r_i that measures the degradation of sensing ability and performance with distance, and the function $\phi : \Omega \to \Re^+$ is a risk density function, which defines a risk for each point in Ω. The density function reflects the probability of occurrence of events in different points, or a measure of relative importance of each point in the domain. Therefore, points with greater risk values should be better covered by agents than points with smaller risk values. This function can be time-invariant or time varying and consequently the coverage metric can be autonomous or non-autonomous. Therefore, the coverage metric encodes the performance of agents in environment Ω, and a hierarchy (risk) associated to each point in Ω, so that one can model different levels of risk or importance associated to different regions in the area.

The maximum coverage metric reflects the fact that each point in the workspace should be the responsibility of the agent with the best sensing performance at that point. Optimal covering of the domain Ω will take place when the objective function (2.4) is maximized. Hence, the necessary conditions to maximize the coverage metric are presented, that will give the basis for a control strategy.

Consider the maximization of the coverage metric $\mathcal{H}(\mathbf{x})$

$$\max_{\mathbf{x}} \mathcal{H}(\mathbf{x}) = \max_{\mathbf{x}} \sum_{i=1}^{n} \mathcal{H}_i(\mathbf{x}_i) = \max_{\mathbf{x}} \sum_{i=1}^{n} \int_{\Omega_i} \max_i f(r_i)\phi(\mathbf{q})d\mathbf{q}, \tag{2.5}$$

According to the definition of Voronoi diagram, the maximum inside the integral induces a partition of Ω into non-overlapping cells, $\mathcal{V}_i$, to give

$$\max_{\mathbf{x}} \mathcal{H}(\mathbf{x}) = \max_{\mathbf{x}} \sum_{i=1}^{n} \mathcal{H}_i(\mathbf{x}_i) = \max_{\mathbf{x}} \sum_{i=1}^{n} \int_{\mathcal{V}_i(\mathbf{x})} f(r_i)\phi(\mathbf{q})d\mathbf{q}, \text{[25]} \tag{2.6}$$

where

$$\mathcal{V}_i(\mathbf{x}) = \{\mathbf{q} \in \Omega : f(r_i) \geq f(r_j), \forall j \neq i\}. \tag{2.7}$$

Since $f(r_i)$ is strictly decreasing, (2.7) is equivalent to

$$\mathcal{V}_i(\mathbf{x}) = \{\mathbf{q} \in \Omega : r_i \leq r_j, \forall j \neq i\}, \tag{2.8}$$

where the region $\mathcal{V}_i$ is the Voronoi partition of $\mathbf{x}_i$. The collection of Voronoi partitions is called the Voronoi diagram or Voronoi tessellation of Ω.

Let $\partial\mathcal{V}_i$ and $\partial\Omega$ be the boundaries of $\mathcal{V}_i$ and Ω respectively. By $\mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})$ we mean a point

$\mathbf{q} \in \partial\mathcal{V}_i$, and $\mathbf{n}_{\partial\mathcal{V}_i}$ represents the outward unit normal of $\partial\mathcal{V}_i$. Given an agent i, we define $\mathcal{N}_i$ as the index set of agents that share Voronoi boundaries with $\mathcal{V}_i$, $\mathcal{N}_i = \{j | \mathcal{V}_i \cap \mathcal{V}_j \neq 0\}$. The set of points on the Voronoi boundary shared by agents i and j are denoted as $l_{ij} = \mathcal{V}_i \cap \mathcal{V}_j$. Then $\mathbf{q}_{l_{ij}}(\mathbf{x}_i, \mathbf{x}_j)$ is a point $\mathbf{q} \in l_{ij}$, and $\mathbf{n}_{l_{ij}}$ is the unit normal of l_{ij} from $\mathbf{x}_i$ to $\mathbf{x}_j$. By the definition of the Voronoi partition (2.8), the following facts are known:

$$\begin{aligned} \partial\mathcal{V}_i &= (\cup_{j\in\mathcal{N}_i} l_{ij}) \cup (\partial\mathcal{V}_i \cap \partial\Omega), \\ l_{ij} &= l_{ji}, \\ \mathbf{n}_{l_{ij}} &= -\mathbf{n}_{l_{ji}}. \end{aligned} \tag{2.9}$$

When d is the Euclidean distance, the common boundaries l_{ij} are hyperplanes, and the Voronoi partitions are convex.[25]

The following lemma expresses an important fact about the objective function (2.6):

Lemma 1 [25]. The gradient of the coverage metric $\mathcal{H}$ with respect to $\mathbf{x}_i$ is given by

$$\frac{\partial\mathcal{H}}{\partial\mathbf{x}_i} = \int_{\mathcal{V}_i(\mathbf{x})} \frac{\partial}{\partial\mathbf{x}_i} f(r_i)\phi(\mathbf{q})d\mathbf{q} \ . \tag{2.10}$$

Proof. By differentiating under the integral sign [11], we have

$$\begin{aligned} \frac{\partial\mathcal{H}}{\partial\mathbf{x}_i} &= \int_{\mathcal{V}_i(\mathbf{x})} \frac{\partial}{\partial\mathbf{x}_i} f(r_i)\phi(\mathbf{q})d\mathbf{q} \\ &+ \int_{\partial\mathcal{V}_i\cap\partial\Omega} f(r_i)\phi(\mathbf{q}) \frac{\partial\mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})}{\partial\mathbf{x}_i} \mathbf{n}_{\partial\mathcal{V}_i} d\mathbf{q} \\ &+ \sum_{j\in\mathcal{N}_i} \int_{l_{ij}} (f(r_i) - f(r_j))\phi(\mathbf{q}) \frac{\partial\mathbf{q}_{l_{ij}}(\mathbf{x}_i, \mathbf{x}_j)}{\partial\mathbf{x}_i} \mathbf{n}_{l_{ij}} d\mathbf{q} \end{aligned} \tag{2.11}$$

where $\frac{\partial\mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})}{\partial\mathbf{x}_i}$ and $\frac{\partial\mathbf{q}_{l_{ij}}(\mathbf{x}_i,\mathbf{x}_j)}{\partial\mathbf{x}_i}$ are $N \times N$ matrices. By definition of l_{ij}, $\| \mathbf{x}_i - \mathbf{q} \| = \| \mathbf{x}_j - \mathbf{q} \|$ $\forall \mathbf{q} \in l_{ij}$, therefore the last term vanishes. On the other hand, since points on the boundary of the main domain do not change their position as a function of $\mathbf{x}_i$, we have

$$\frac{\partial\mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})}{\partial\mathbf{x}_i} = 0, \quad \forall \mathbf{q} \in \partial\mathcal{V}_i \cap \partial\Omega \tag{2.12}$$

and so the second term vanishes, too, and the lemma is proved. It means that an agent i can calculate its own gradient component, $\frac{\partial\mathcal{H}}{\partial\mathbf{x}_i}$, using only data relevant to its Voronoi cell.

In this work we consider planar region ($N = 2$), r (the distance metric) is the Euclidean distance, $f(r_i) = \alpha e^{-\beta r_i^2}$ where $\alpha, \beta > 0$ [21], and the environment Ω is convex. In this restricted setting, the objective function (coverage metric) becomes

$$\mathcal{H}(\mathbf{x}) = \sum_{i=1}^{n} \mathcal{H}_i(\mathbf{x}_i) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} \alpha e^{-\beta r_i^2} \phi(\mathbf{q})d\mathbf{q}. \tag{2.13}$$

Define the quantities

$$\begin{aligned}
&\tilde{m}_\phi(\mathcal{V}_i(t)) = \int_{\mathcal{V}_i} \tilde{\phi}(\mathbf{q}, \mathbf{x}_i(t))d\mathbf{q}, \quad \text{as generalized mass} \\
&\tilde{\mathbf{c}}_\phi(\mathcal{V}_i(t)) = \frac{1}{\tilde{m}_\phi(\mathcal{V}_i(t))} \int_{\mathcal{V}_i} \mathbf{q}\tilde{\phi}(\mathbf{q}, \mathbf{x}_i(t))d\mathbf{q}, \quad \text{as generalized centroid, and} \\
&\tilde{\phi}(\mathbf{q}, \mathbf{x}_i) = -2\phi(\mathbf{q})(\frac{\partial f(r_i)}{\partial(r_i^2)}) = 2\alpha\beta e^{-\beta r_i^2}\phi(\mathbf{q}) \quad \text{as generalized risk density.}
\end{aligned} \tag{2.14}$$

Then the gradient of coverage metric can be written as [25]:

$$\begin{aligned}
\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} = &\int_{\mathcal{V}_i(\mathbf{x})} \frac{\partial}{\partial \mathbf{x}_i} f(r_i)\phi(\mathbf{q})d\mathbf{q} = \\
&\int_{\mathcal{V}_i(\mathbf{x})} \frac{\partial(\alpha e^{-\beta\|\mathbf{x}_i-\mathbf{q}\|^2})}{\partial \mathbf{x}_i}\phi(\mathbf{q})d\mathbf{q} = \\
&\int_{\mathcal{V}_i(\mathbf{x})} -2(\mathbf{x}_i-\mathbf{q})\alpha\beta e^{-\beta\|\mathbf{x}_i-\mathbf{q}\|^2}\phi(\mathbf{q})d\mathbf{q} = \\
&\int_{\mathcal{V}_i(\mathbf{x})} \mathbf{q}\tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q} - \int_{\mathcal{V}_i(\mathbf{x})} \mathbf{x}_i\tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q} = \\
&\tilde{m}_\phi(\mathcal{V}_i(t))(\tilde{\mathbf{c}}_\phi(\mathcal{V}_i(t)) - \mathbf{x}_i).
\end{aligned} \tag{2.15}$$

In the two following sections, coverage control with time-invariant and time-varying risk density functions will be discussed respectively.

2.3.1 Coverage Control with Time-Invariant Density Function

For time-invariant risk density functions, the optimal configuration of agents are centroidal Voronoi tessellations. This fact, known as Lloyd algorithm, is established by the following lemma, which defines also the optimal trajectories to maximize the coverage metric.

Lemma 2 [18]**.** Given the autonomous risk density $\phi(\mathbf{q}), \mathbf{q} \in \Omega, \kappa > 0$, consider agents trajectories defined by the first order kinematics state of agent i, $i = 1, \cdots, n$.

$$\dot{\mathbf{x}}_i = \mathbf{u}_i, \tag{2.16}$$

where $\mathbf{u}_i(t) \in \Re^2$ is the velocity of the ith agent at time t. And consider each agent is a particle in two dimensional region. The state trajectories of a system of n agents can be prescribed by using the model (2.16) as,

$$\dot{\mathbf{x}}(t) = \mathbf{u}(t), \ \mathbf{x}^0 = \mathbf{x}(0), \tag{2.17}$$

where $(\mathbf{u}_1(t), \cdots, \mathbf{u}_n(t))^\mathsf{T} \equiv \mathbf{u}(t) \in \Re^{2n}$ and $\mathbf{x}(0)$ is the set of initial states of agents. The following feedback law maximizes the coverage metric defined in (2.13) with asymptotic

convergence of agent i to its generalized Voronoi centroid $\tilde{\mathbf{c}}_\phi(\mathcal{V}_i)$:

$$\mathbf{u}_i = \kappa \frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} = \kappa \tilde{m}_\phi(\mathcal{V}_i)(\tilde{\mathbf{c}}_\phi(\mathcal{V}_i) - \mathbf{x}_i), \tag{2.18}$$

Proof [25]. To prove this lemma, one may consider the negative definiteness of the Hessian of $\mathcal{H}$ at such points $\mathbf{x} = \tilde{\mathbf{c}}_\phi(\mathcal{V})$. Therefore, it seems necessary to explain Hessian matrix briefly.

Hessian Matrix

In mathematics, the Hessian (matrix) is a square matrix contains second partial derivatives of a function of many variables which describes the local curvature of mentioned function.

Given the real-valued function $f(x_1, x_2, \cdots, x_n)$, if all second-order partial derivatives of function exist and are continuous over the domain of f, then the Hessian of f is

$$H(f)_{ij}(\mathbf{x}) = \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \tag{2.19}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$. Thus

$$H(f) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \, \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \, \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}. \tag{2.20}$$

Because f is often clear from context, $H(f)(\mathbf{x})$ is usually abbreviated to $H(\mathbf{x})$. Sometimes, the determinant of the mentioned matrix is also considered as the Hessian.

Second Derivative Test: In mathematics, the second-order partial derivative test is a method used to specify if a critical point of a multi-variable function is a local maximum, minimum or saddle point.

There is a generalization of the mentioned rule for a function of more than two variables. In this case, instead of analysing the determinant of the Hessian, one can apply the following test at any critical point for which the Hessian is invertible:

- If the Hessian matrix is positive definite (all eigenvalues are positive) at critical point, then the function attains a local minimum at that point.

- If the Hessian matrix is negative definite (all eigenvalues are negative) at critical point, then the function attains a local maximum at that point.
- If the Hessian matrix has both negative and positive eigenvalues then critical point is a saddle point of f.

Now, we can obtain the Hessian of $\mathcal{H}$ and prove that $\mathcal{H}$ has maximum at $\mathbf{x} = \mathbf{c}_\phi(\mathcal{V}(t))$.

$$
\begin{aligned}
H(\mathcal{H}) &= \begin{bmatrix}
\dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_1^2} & \dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} & \cdots & \dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_1 \partial \mathbf{x}_n} \\
\dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_2 \partial \mathbf{x}_1} & \dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_2^2} & \cdots & \dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_2 \partial \mathbf{x}_n} \\
\vdots & \vdots & \ddots & \vdots \\
\dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_n \partial \mathbf{x}_1} & \dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_n \partial \mathbf{x}_2} & \cdots & \dfrac{\partial^2 \mathcal{H}}{\partial \mathbf{x}_n^2}
\end{bmatrix} \\
&= \begin{bmatrix}
-\kappa \tilde{m}_\phi(\mathcal{V}_i) & 0 & \cdots & 0 \\
0 & -\kappa \tilde{m}_\phi(\mathcal{V}_i) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & -\kappa \tilde{m}_\phi(\mathcal{V}_i)
\end{bmatrix}
\end{aligned}
\tag{2.21}
$$

since all eigenvalues of $H(\mathcal{H})$ are negative and equal to $-\kappa \tilde{m}_\phi(\mathcal{V}_i)$, $\mathcal{H}$ has local maximum at $\mathbf{x} = \tilde{\mathbf{c}}_\phi(\mathcal{V})$ and lemma 2 is proved.

2.3.2 Coverage Control with Time-Varying Density Function

Due to this fact that in the real world, many phenomena can be modelled by time-varying risk density rather than time-invariant one, there has been an increasing attention to the cases with the time-varying density functions. Cortes, Martinez, Karatas, and Bullo in 2004, presented control and coordination algorithms for networks of autonomous vehicles. They focused on groups of agents performing distributed sensing missions where each agent plays the role of a mobile sensor. They designed several gradient descent algorithms for a class of functions which encodes sensing policies and optimal coverage [7]. In particular, they presented coverage algorithms which are asynchronous, distributed, adaptive, and verifiably asymptotically correct.

Pimenta et al. (2009) addressed the problem of simultaneously covering an environment and tracking Targets. The main task of networked agents is covering environments with time-varying risk density functions under the locational optimization framework. This allows for combining the basic subtasks: environmental coverage, target tracking, and task assignment. A decentralized controller with guaranteed convergence is also devised.[25]

Lekien and Leonard in 2010, investigated non-uniform coverage of a planar domain by a network of mobile agents. They also focused on time-varying and non-autonomous coverage metrics and the design of an appropriate control algorithms to cover domains with slowly evolving, non-uniform metrics. In this paper, the authors define optimal in a slightly different sense that agents must have equal magnitude of resources in their generalized Voronoi partitions. Nevertheless, the risk density function is necessary to evolve slowly enough with time to guarantee the stability of the algorithm presented in the paper.[15]

Lee et al. (2013) presented an approach to control a multi-agent system by choosing a time-varying risk density function. A control law which causes the agents to track the risk density function while providing optimal coverage of the environment has been proposed. They employed Voronoi tessellations of the domain in their design of control feedback law to provide optimal coverage over the domain. The agents first converge to centroidal Voronoi tessellation under time-invariant risk density function, and then the risk density function is switched to a time-varying one. Under the proposed algorithm, the agents track Voronoi centroids provided that they are on Voronoi centroids initially. Authors used the proposed algorithm to achieve desired behaviour of the system of agents while employing the time-varying risk density function as an input. The suggested algorithm placed no additional assumptions on the evolution of the risk density function, at the expense of becoming a centralized control law.[14]

In 2014, Miah and Spinello investigate the non-uniform coverage of a planar region by a platoon of autonomous mobile agents when communications among them are stochastically intermittent. Given a set of generating points and an appropriate metric, the solution of the optimal coverage problem is the centroidal Voronoi tessellation, with a set of mobile agents converging to the corresponding centroids of the Voronoi partitions. In the framework of decentralized motion control, this implementation requires that all agents have knowledge of the state of the other agents in the platoon. They generalize this scenario by considering the optimal area coverage when a group of agents share information in a time varying, stochastically intermittent way. They embed on board of each agent a full state estimator that relies on local estimates and on information received by others, when available and show that under appropriate conditions on the communication network, all agents' estimates asymptotically converge to true states while maximizing the coverage metric despite intermittent communications.[21]

There are different kinds of time-varying risk density functions considered by researchers. For instance in [7], Cortes et al. introduces different geometric patterns of risk density function. They suggested the use of decentralized coverage algorithms as formation control algorithms, and presented various risk density functions which lead the multi-agent system to predetermined geometric patterns. In particular, they present simple risk density functions which lead to segments, polygons, ellipses, or uniform distributions inside convex domains.

They assume a planar domain Ω, k as a large positive gain and $\mathbf{q} = (x, y) \in \Omega \subset \Re^2$. Then they denoted α and β as real numbers. Respectively, they consider an ellipse equation as $\alpha(x - x_c)^2 + b(y - y_c)^2 = r^2$, and introduced the density function

$$\phi(\mathbf{q}) = exp(-k(\alpha(x - x_c)^2 + \beta(y - y_c)^2 - r^2)^2), \tag{2.22}$$

where (x_c, y_c) is the coordination of target's centre. One can consider the equivalent time-varying risk density function as

$$\phi(\mathbf{q}, t) = exp(-k(\alpha(x - x_c(t))^2 + \beta(y - y_c(t))^2 - r^2)^2), \tag{2.23}$$

when there is a mobile target inside the planar workspace. Pimenta et al. also model the task assignment problem by considering radial basis risk density functions, ϕ_j, which represents each target. They defined the time-varying risk density function

$$\phi(\mathbf{q}, t) = \sum_{j=1}^{L} \alpha_j \phi_j(\mathbf{q}, t) + \beta \tag{2.24}$$

where α_j and β are positive constants that tune the importance of an assigned mission. The parameter α_j defines the importance of tracking target j, while the parameter β defines the importance of uniform coverage. They considered radial basis functions that are centred at the target position, reach a maximum amount at this position and suggested the Gaussian (bell-shaped) given by

$$\phi_j(\mathbf{q}, t) = Aexp(-\frac{(x - x_j(t))^2}{2\sigma} - \frac{(x - x_j(t))^2}{2\sigma}), \tag{2.25}$$

where (x_j, y_j) is the coordination of target's centre.[25]

Authors in [7] also define the smooth ramp function $SR_k(x) = x(\arctan(kx/\pi + (1/2)))$, and assumed

$$\phi(\mathbf{q}) = exp(-kSR_k(\alpha(x - x_c)^2 + \beta(y - y_c)^2 - r^2)^2), \tag{2.26}$$

as the time-invariant risk density function or equivalently

$$\phi(\mathbf{q}, t) = exp(-kSR_k(\alpha(x - x_c(t))^2 + \beta(y - y_c(t))^2 - r^2)^2), \tag{2.27}$$

as the time-varying one.

In this work, instead of considering a specific pre-assigned function we consider a time-varying risk density evolving according to a diffusion equation with varying boundary conditions that quantify a time-varying risk on the border of the workspace. Boundary conditions model a time varying flux of external targets coming into the area, averaged over the boundary length, so that we do not consider the individual kinematics of incoming targets but rather their averaged, distributed effect.

Chapter 3

Coordinated Deployment of Multiple Autonomous Agents in Area Coverage Problems with Evolving Risk

3.1 Problem Definition

We consider a domain in which a group of agents have the mission of covering an area optimally, and simultaneously react to external threats entering the area, which are quantified by time varying boundary condition for the risk function defined inside the area to be protected. In the described situation, considering the evolving boundary values, the coverage metric $\mathcal{H}$ is non-autonomous and trajectories defined by the feedback law (2.18) have to be restated to ensure optimality.

In this chapter, a discussion about the risk density function and its modelling procedure is presented as the first step. Then we propose a non-autonomous feedback law that generates asymptotically optimal trajectories with respect to the coverage metric. Theoretical predictions are illustrated and discussed through different simulation scenarios.

3.2 Calculation of The Time-Varying Density Function

Assume a rectangular region, Ω, and the time-varying density $\phi(\mathbf{q}, t)$ to obey a conservation law in Ω, with evolution dictated by a diffusion equation that in absence of sources in Ω is driven by time-varying boundary conditions. Time-varying boundary conditions model an evolving flux of external threats penetrating into the area, averaged over the boundary length. It means that we assume distributed effect of incoming threats instead of considering their individual kinematics. This approach is similar to the one commonly

used in continuum physics, in which kinematic descriptors are averaged over spatial region and appropriate continuum fields are introduced to describe their evolution.
The initial boundary-values problem for ϕ is posed as

$$\frac{\partial \phi}{\partial t} + \text{div}\mathbf{\Phi} = 0 \text{ in } \Omega, \tag{3.1a}$$
$$\phi(\mathbf{q}, 0) = \phi_0(\mathbf{q}) \text{ in } \Omega, \tag{3.1b}$$
$$\phi(\mathbf{q}, t) = \bar{\phi}(\mathbf{q}, t) \text{ on } \partial\Omega_\phi, \tag{3.1c}$$
$$\mathbf{\Phi} \cdot \mathbf{n} = \bar{\Phi} \text{ on } \partial\Omega_{\mathbf{\Phi}}, \tag{3.1d}$$

where $\mathbf{\Phi}$ is the flux of ϕ, "div" is the divergence operator, " $\cdot$" is the inner product between vectors, ϕ_0 is a prescribed initial condition that reflects the a-priori risk distribution in the domain, and $\bar{\phi}$ and $\bar{\Phi}$ are boundary conditions for ϕ and for the normal derivative of $\mathbf{\Phi}$ ($\mathbf{n}$ is the unit outward normal of the boundary), prescribed respectively on the boundaries $\partial\Omega_\phi$ and $\partial\Omega_{\mathbf{\Phi}}$. In order to close the system, a constitutive relation linking the flux $\mathbf{\Phi}$ to its density ϕ is needed. Here, we adopt a simple first gradient constitutive law

$$\mathbf{\Phi} = -k\frac{\partial \phi}{\partial \mathbf{q}} \tag{3.2}$$

according to which the flux is determined by the spatial variation of the risk, where k is risk conduction constant. Substitution into (3.1) gives

$$\frac{\partial \phi}{\partial t} = k\triangle\phi \text{ in } \Omega, \tag{3.3a}$$
$$\phi(\mathbf{q}, 0) = \phi_0(\mathbf{q}) \text{ in } \Omega, \tag{3.3b}$$
$$\phi(\mathbf{q}, t) = \bar{\phi}(\mathbf{q}, t) \text{ on } \partial\Omega_\phi, \tag{3.3c}$$
$$-k\frac{\partial \phi}{\partial \mathbf{q}} \cdot \mathbf{n} = \bar{\Phi} \text{ on } \partial\Omega_{\mathbf{\Phi}}, \tag{3.3d}$$

where $\triangle$ is the Laplacian operator.

As mentioned, the workspace Ω is considered to be a rectangular region, and referred to Cartesian rectangular coordinates $\{x, y\}$ with axes parallel to the sides of Ω and the origin on the bottom left corner (Fig. 3.2). The Laplacian operator in this system of coordinates assumes the simple form $\triangle = \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y}$.

Patankar in [23] suggested a discretization method for solving a two dimensional heat transfer problem. He considered a two-dimensional grid as shown in Fig. 3.1. For a grid point P, E and W (denoting east and west) are x-direction neighbours, while N and S (denoting north and south) are y-direction neighbours. The control volume around P is shown by dashed lines. Distances Δx, $(\delta x)_e$, $(\delta x)_w$, Δy, $(\delta y)_n$ and $(\delta y)_s$ are as shown in the figure.

By considering Patankar's discretization method, a system with N_x and N_y nodes, respectively along the coordinated directions x and y is defined. Let Δx and Δy be the space discretization steps and $\Delta x = (\delta x)_e = (\delta x)_w = \Delta y = (\delta y)_n = (\delta y)_s = dl$, a

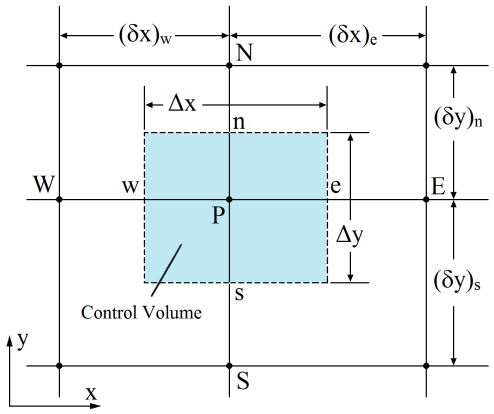


Figure 3.1: Control volume for the two-dimensional situation.[23]

spatial grid with nodes located at $(j_x\Delta x, j_y\Delta y)$, for $j_x = 0, 1, \cdots, (N_x - 1)$ and $j_y = 0, 1, \cdots, (N_y - 1)$ is considered. By denoting the $2D$ locations of five contiguous interior nodes as

$$\begin{aligned}
\mathbf{q}_{j_x+1,j_y} &= ((j_x + 1)\Delta x, (j_y)\Delta y),\\
\mathbf{q}_{j_x,j_y} &= ((j_x)\Delta x, (j_y)\Delta y),\\
\mathbf{q}_{j_x-1,j_y} &= ((j_x - 1)\Delta x, (j_y)\Delta y),\\
\mathbf{q}_{j_x,j_y-1} &= ((j_x)\Delta x, (j_y - 1)\Delta y),\\
\mathbf{q}_{j_x,j_y+1} &= ((j_x)\Delta x, (j_y + 1)\Delta y),
\end{aligned} \tag{3.4}$$

The following finite difference scheme is established

$$\begin{aligned}
a_p\phi(\mathbf{q}_{j_x,j_y}, t_j) = a_e\phi(\mathbf{q}_{j_x+1,j_y}, t_j)\\
+ a_w\phi(\mathbf{q}_{j_x-1,j_y}, t_j)\\
+ a_n\phi(\mathbf{q}_{j_x,j_y+1}, t_j)\\
+ a_s\phi(\mathbf{q}_{j_x,j_y-1}, t_j) + b,
\end{aligned} \tag{3.5}$$

where

$$a_e = k_e, \tag{3.6a}$$
$$a_w = k_w, \tag{3.6b}$$
$$a_n = k_n, \tag{3.6c}$$
$$a_s = k_s, \tag{3.6d}$$
$$a_p^0 = \frac{(dl^2)}{t_j - t_{j-1}}, \tag{3.6e}$$
$$b = \frac{(dl^2)\phi(\mathbf{q}_{j_x,j_y}, t_{j-1})}{t_j - t_{j-1}}, and \tag{3.6f}$$
$$a_p = a_e + a_w + a_n + a_s + a_p^0. \tag{3.6g}$$

Equation (3.5) generates a linear system of $N_xN_y-(2N_x+2N_y-4)$ equations for the same number of unknowns that are the nodal values of the density inside the domain at time t_j. The number of forcing terms equals $2N_x+2N_y-4$, with entries determined by the boundary conditions at the same time.

3.3 Feedback law for time-varying risk density

Lemma 3. Let $\phi(\mathbf{q},t)$ be the density governed by the conservation law (3.3), $\tilde{\mathbf{c}}_\phi(\mathcal{V}) \equiv (\tilde{\mathbf{c}}_\phi(\mathcal{V}_1),\cdots,\tilde{\mathbf{c}}_\phi(\mathcal{V}_n))^\mathsf{T}$, and $\mathcal{H}$ be the non-autonomous coverage metric defined in (3.7),

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

The trajectories generated by the simple integrator kinematics (2.16) with the feedback law

$$\mathbf{u}_i=\frac{\partial\mathcal{H}/\partial\mathbf{x}_i}{\parallel\partial\mathcal{H}/\partial\mathbf{x}_i\parallel^2}\left(\kappa\parallel\tilde{\mathbf{c}}_\phi(\mathcal{V}_i)-\mathbf{x}_i\parallel^2-\frac{\partial\mathcal{H}_i}{\partial t}\right), \tag{3.8}$$

for $\kappa>0$, asymptotically maximize the coverage metric $\mathcal{H}$ with convergence of $\mathbf{x}_i$ to its generalized Voronoi centroid $\tilde{\mathbf{c}}_\phi(\mathcal{V}_i(t))$ for $i=1,\cdots,n$, where

$$\frac{\partial\mathcal{H}_i}{\partial t}=\int_{\mathcal{V}_i} f(r_i)\frac{\partial\phi}{\partial t}d\mathbf{q}. \tag{3.9}$$

Proof. In order to prove that the generated trajectories asymptotically maximize the coverage metric and guarantee the stability of the agents' trajectories considering a time-varying system and non-autonomous coverage metric, Barbalat's lemma can be employed. Note that in the cases with time-varying risk density functions, we do not have a set of points (centroids) which should be followed by agents, rather we have a set of trajectories.

Barbalat's Lemma

Consider the scalar function (candidate Lyapunov function) $V(\mathbf{x},t)$, satisfying following conditions:

1. $V(\mathbf{x},t)$ is lower bounded
2. $\dot{V}(\mathbf{x},t)$ is negative semi-definite
3. $\dot{V}(\mathbf{x},t)$ is uniformly continuous in time

then $\dot{V}(\mathbf{x},t)\longrightarrow 0$ as $t\to\infty$. It is necessary to mention that the third condition will be satisfied if $\ddot{V}$ is finite.

We characterize the Lyapunov function

$$V(\mathbf{x},t) = \frac{1}{\mathcal{H}(\mathbf{x},t)}, \tag{3.10}$$

Since $f(r_i)$ and $\phi(\mathbf{q},t)$ were defined as positive functions, V is lower bounded and positive ($V > 0$). Therefore, the first condition is satisfied.

The time derivative of Lyapunov function V is given by

$$\dot{V}(\mathbf{x},t) = \frac{-\dot{\mathcal{H}}(\mathbf{x},t)}{\mathcal{H}^2(\mathbf{x},t)}, \tag{3.11}$$

where

$$\dot{\mathcal{H}}(\mathbf{x},t) = \sum_{i=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i}^{\mathsf{T}} \dot{\mathbf{x}}_i + \frac{\partial \mathcal{H}_i}{\partial t} \right). \tag{3.12}$$

By substituting the feedback law (3.8) in equation (3.12) we obtain

$$\dot{\mathcal{H}}(\mathbf{x},t) = \sum_{i=1}^{n} \kappa \parallel \tilde{\mathbf{c}}_\phi(\mathcal{V}_i) - \mathbf{x}_i \parallel^2 . \tag{3.13}$$

Hence $\dot{\mathcal{H}}$ is positive definite and $\dot{V}(\mathbf{x},t)$ is negative definite, which implies that the second condition is also met.

The second derivative of the Lyapunov function is

$$\ddot{V}(\mathbf{x},t) = \frac{-\ddot{\mathcal{H}}(\mathbf{x},t)\mathcal{H}^2(\mathbf{x},t) + 2\mathcal{H}(\mathbf{x},t)\dot{\mathcal{H}}^2(\mathbf{x},t)}{\mathcal{H}^4(\mathbf{x},t)} = \frac{-\ddot{\mathcal{H}}(\mathbf{x},t)\mathcal{H}(\mathbf{x},t) + 2\dot{\mathcal{H}}^2(\mathbf{x},t)}{\mathcal{H}^3(\mathbf{x},t)}, \tag{3.14}$$

where

$$\ddot{\mathcal{H}}(\mathbf{x},t) = \sum_{j=1}^{n} \frac{\partial \dot{\mathcal{H}}}{\partial \mathbf{x}_j}^{\mathsf{T}} \dot{\mathbf{x}}_j + \frac{\partial \dot{\mathcal{H}}}{\partial t}. \tag{3.15}$$

Substitution from (3.13) into (3.15) gives

$$\begin{aligned} \ddot{\mathcal{H}}(\mathbf{x},t) = &\sum_{i=1}^{n}\sum_{j=1}^{n} \left(2\kappa(\tilde{\mathbf{c}}_\phi(\mathcal{V}_i) - \mathbf{x}_i)^{\mathsf{T}} \left(\frac{\partial \tilde{\mathbf{c}}_\phi(\mathcal{V}_i)}{\partial \mathbf{x}_j} - \mathbf{I}\delta_{ij} \right) \dot{\mathbf{x}}_j \right) \\ &+ \sum_{i=1}^{n} \left(2\kappa(\tilde{\mathbf{c}}_\phi(\mathcal{V}_i) - \mathbf{x}_i)^{\mathsf{T}} \frac{\partial \tilde{\mathbf{c}}_\phi(\mathcal{V}_i)}{\partial t} \right), \end{aligned} \tag{3.16}$$

in which $\mathbf{I}$ is the identity matrix and δ_{ij} is the Kronecker delta.

In order to have a finite $\ddot{V}(\mathbf{x},t)$, $\dot{\mathcal{H}}(\mathbf{x},t)$ and $\ddot{\mathcal{H}}(\mathbf{x},t)$ should be bounded. Due to (3.13) and the boundedness of the workspace Ω, $\dot{\mathcal{H}}(\mathbf{x},t)$ is bounded, therefore the boundedness

of $\ddot{V}(\mathbf{x},t)$ depends only on the boundedness of $\ddot{\mathcal{H}}(\mathbf{x},t)$.

Due to the boundedness of the domain, $2\kappa(\tilde{\mathbf{c}}_\phi(\mathcal{V}_i)-\mathbf{x}_i)$ is bounded. On the other hand, the term $\frac{\partial\tilde{\mathbf{c}}_\phi(\mathcal{V}_i)}{\partial\mathbf{x}_j}-\mathbf{I}\delta_{ij}$ is also bounded as long as the domain is bounded, see [14] and [8]. As it is clear from (3.8), the feedback law $\dot{\mathbf{x}}_j$ is also bounded if $\frac{\partial\mathcal{H}_j}{\partial t}$ is bounded. Finally the expression of $\frac{\partial\tilde{\mathbf{c}}_\phi(\mathcal{V}_i)}{\partial t}$ is [14]:

$$\frac{\partial\tilde{\mathbf{c}}_\phi(\mathcal{V}_i)}{\partial t}=\frac{\tilde{m}_\phi(\mathcal{V}_i)\int_{\mathcal{V}_i}\mathbf{q}\frac{\partial\tilde{\phi}_i(\mathbf{q},t)}{\partial t}d\mathbf{q}-\frac{\partial\tilde{m}_\phi(\mathcal{V}_i)}{\partial t}\int_{\mathcal{V}_i}\mathbf{q}\tilde{\phi}_i(\mathbf{q},t)d\mathbf{q}}{\tilde{m}_\phi^2(\mathcal{V}_i)}, \tag{3.17}$$

where from (2.14) we have

$$\frac{\partial\tilde{m}_\phi(\mathcal{V}_i)}{\partial t}=\int_{\mathcal{V}_i}\frac{\partial\tilde{\phi}_i(\mathbf{q},t)}{\partial t}d\mathbf{q} \tag{3.18}$$

Hence, because of the boundedness of the domain, the partial time derivative of centroid is bounded if $\frac{\partial\tilde{\phi}_i(\mathbf{q},t)}{\partial t}$ is bounded.

In order to find the relationship between the boundedness of $\frac{\partial\mathcal{H}_i}{\partial t}$ and the evolution of the boundary conditions that in turn determine the risk density ϕ, the divergence theorem can be employed

$$\frac{\partial\mathcal{H}}{\partial t}=\sum_{i=1}^{n}\frac{\partial\mathcal{H}_i}{\partial t}=\sum_{i=1}^{n}\int_{\mathcal{V}_i}f(\mathbf{q},\mathbf{x}_i)\frac{\partial\phi(\mathbf{q},t)}{\partial t}d\mathbf{q}. \tag{3.19}$$

Substitution from (3.3a) into (3.19) gives

$$\frac{\partial\mathcal{H}}{\partial t}=\sum_{i=1}^{n}\frac{\partial\mathcal{H}_i}{\partial t}=\sum_{i=1}^{n}\int_{\mathcal{V}_i}kf(\mathbf{q},\mathbf{x}_i)\Delta\phi(\mathbf{q},t)d\mathbf{q}. \tag{3.20}$$

By using the vector calculus identity $f(\mathbf{q},\mathbf{x}_i)\Delta\phi(\mathbf{q},t)=\mathrm{div}(f(\mathbf{q},\mathbf{x}_i)\frac{\partial\phi(\mathbf{q},t)}{\partial\mathbf{q}})-\frac{\partial f(\mathbf{q},\mathbf{x}_i)}{\partial\mathbf{q}}\cdot\frac{\partial\phi(\mathbf{q},t)}{\partial\mathbf{q}}$ where from the definition of r_i

$$\begin{aligned}\frac{\partial f(\mathbf{q},\mathbf{x}_i)}{\partial\mathbf{q}}&=\frac{\partial f(\mathbf{q},\mathbf{x}_i)}{\partial r_i}\frac{\partial r_i(\mathbf{q},\mathbf{x}_i)}{\partial\mathbf{q}}\\&=-\frac{\partial f(\mathbf{q},\mathbf{x}_i)}{\partial r_i}\frac{\partial r_i(\mathbf{q},\mathbf{x}_i)}{\partial\mathbf{x}_i}=-\frac{\partial f(\mathbf{q},\mathbf{x}_i)}{\partial\mathbf{x}_i},\end{aligned} \tag{3.21}$$

we can write

$$\frac{\partial\mathcal{H}}{\partial t}=\sum_{i=1}^{n}\frac{\partial\mathcal{H}_i}{\partial t}=\sum_{i=1}^{n}\int_{\mathcal{V}_i}k\left(\mathrm{div}\left(f(\mathbf{q},\mathbf{x}_i)\frac{\partial\phi(\mathbf{q},t)}{\partial\mathbf{q}}\right)+\frac{\partial f(\mathbf{q},\mathbf{x}_i)}{\partial\mathbf{x}_i}\cdot\frac{\partial\phi(\mathbf{q},t)}{\partial\mathbf{q}}\right)d\mathbf{q}. \tag{3.22}$$

Finally by employing the divergence theorem to transfer the domain integral into an integral on the boundary we get

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial t} &= \sum_{i=1}^{n} \frac{\partial \mathcal{H}_i}{\partial t} = \sum_{i=1}^{n} \int_{\mathcal{V}_i} k \frac{\partial f(\mathbf{q}, \mathbf{x}_i)}{\partial \mathbf{x}_i} \cdot \frac{\partial \phi(\mathbf{q}, t)}{\partial \mathbf{q}} d\mathbf{q} \\ &+ \int_{\partial \Omega_\phi} k f(\mathbf{q}, \mathbf{x}_i) \frac{\partial \phi(\mathbf{q}, t)}{\partial \mathbf{q}} \cdot \mathbf{n} \; d\mathbf{q} + \int_{\partial \Omega_{\mathbf{\Phi}}} k f(\mathbf{q}, \mathbf{x}_i) \frac{\partial \phi(\mathbf{q}, t)}{\partial \mathbf{q}} \cdot \mathbf{n} \; d\mathbf{q},\end{aligned} \tag{3.23}$$

where $\mathbf{n}$ is the unit outward normal.
Substitution of (3.3c) and (3.3d) into (3.23) gives

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial t} &= \sum_{i=1}^{n} \frac{\partial \mathcal{H}_i}{\partial t} = \sum_{i=1}^{n} \int_{\mathcal{V}_i} k \frac{\partial f(\mathbf{q}, \mathbf{x}_i)}{\partial \mathbf{x}_i} \cdot \frac{\partial \phi(\mathbf{q}, t)}{\partial \mathbf{q}} d\mathbf{q} \\ &+ \int_{\partial \Omega_\phi} k f(\mathbf{q}, \mathbf{x}_i) \frac{\partial \bar{\phi}(\mathbf{q}, t)}{\partial \mathbf{q}} \cdot \mathbf{n} \; d\mathbf{q} - \int_{\partial \Omega_{\mathbf{\Phi}}} f(\mathbf{q}, \mathbf{x}_i) \bar{\mathbf{\Phi}} \; d\mathbf{q}.\end{aligned} \tag{3.24}$$

In this work, it is assumed that $\partial \Omega_{\mathbf{\Phi}} = 0$, so the equation (3.24) reduces to

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial t} &= \sum_{i=1}^{n} \frac{\partial \mathcal{H}_i}{\partial t} = \sum_{i=1}^{n} \int_{\mathcal{V}_i} k \frac{\partial f(\mathbf{q}, \mathbf{x}_i)}{\partial \mathbf{x}_i} \cdot \frac{\partial \phi(\mathbf{q}, t)}{\partial \mathbf{q}} d\mathbf{q} \\ &+ \int_{\partial \Omega_\phi} k f(\mathbf{q}, \mathbf{x}_i) \frac{\partial \bar{\phi}(\mathbf{q}, t)}{\partial \mathbf{q}} \cdot \mathbf{n} \; d\mathbf{q}.\end{aligned} \tag{3.25}$$

Due to absence of risk source inside the workspace and the boundedness of the domain, the first term is bounded provided that f and ϕ have at least piecewise continuous gradients, or more formally provided that the two functions are square summable in Ω, so that they belong to the Sobolev space of functions with bounded square gradient in the domain Ω, with respect to the metric defined by the integration. Therefore, in order to have a finite $\frac{\partial \mathcal{H}}{\partial t}$ and subsequently finite $\frac{\partial \mathcal{H}_i}{\partial t}$, $\frac{\partial \bar{\phi}(\mathbf{q},t)}{\partial \mathbf{q}} \cdot \mathbf{n}$ should be bounded.

On the other hand, from (2.14) we have

$$\frac{\partial \tilde{\phi}_i(\mathbf{q}, t)}{\partial t} = 2\alpha\beta e^{-\beta r_i^2} \frac{\partial \phi}{\partial t} \tag{3.26}$$

Therefore $\frac{\partial \tilde{\phi}_i(\mathbf{q},t)}{\partial t}$ is bounded if $\frac{\partial \phi(\mathbf{q},t)}{\partial t}$ is finite. With the same procedure it can be shown that $\frac{\partial \phi(\mathbf{q},t)}{\partial t}$ is bounded if $\frac{\partial \bar{\phi}(\mathbf{q},t)}{\partial \mathbf{q}} \cdot \mathbf{n}$ is finite. This links the stability of agents' trajectories to the boundedness of the gradient of the prescribed boundary risk.

3.4 Simulation Results

In this section, some numerical results that demonstrate the effect of time-varying boundary conditions on the non-uniform coverage optimization will be presented. As an application,

a harbour protection scenario in military domains or an area surrounded by hazardous material are considered.

3.4.1 Setup

Consider a two dimensional planar region Ω by a square of size 100×100 m. The discretization steps are set as $\Delta x = \Delta y = 1$ m.

Five agents ($n = 5$) are placed at initial positions $(20, 20)$ m, $(40, 20)$ m, $(80, 50)$ m, $(40, 80)$ m, and $(20, 60)$ m as shown in Fig. 3.2. In this work, the agents are modelled as fully actuated, omnidirectional mass particles moving in a two dimensional domain, therefore possessing two translational degrees of freedom. This kinematics may not be realistic for implementation in real life scenarios, where more complicated conditions have to be accounted for in order to realistically describe mobile agents and their interactions with the environment. Typically, mobile agents are under-actuated, and their kinematics has to include rotational degrees of freedom, which do not allow for instantaneous change of direction as it is in the cases simulated here.

Foremost, initial Voronoi partitions are selected according to the generalized model (2.7) with $f(r_i) = \alpha e^{-\beta r_i^2}$, where $\alpha = 5$ and $\beta = 10^{-3}$. The initial density is $\phi(\mathbf{q}, 0) = \phi_0(\mathbf{q}) = 3 \times 10^{-3}, \forall \mathbf{q} \in \Omega$. The agents' trajectories are updated according to the kinematic model (2.17) with $\mathbf{u}_i(t)$ being defined in (3.8) for $\kappa = 1$. The sampling times for the discrete-time implementation of (2.17) are chosen as $t_s = 1$ s such that $t = K \times t_s$, for $K = 0, 1, 2, \cdots$.

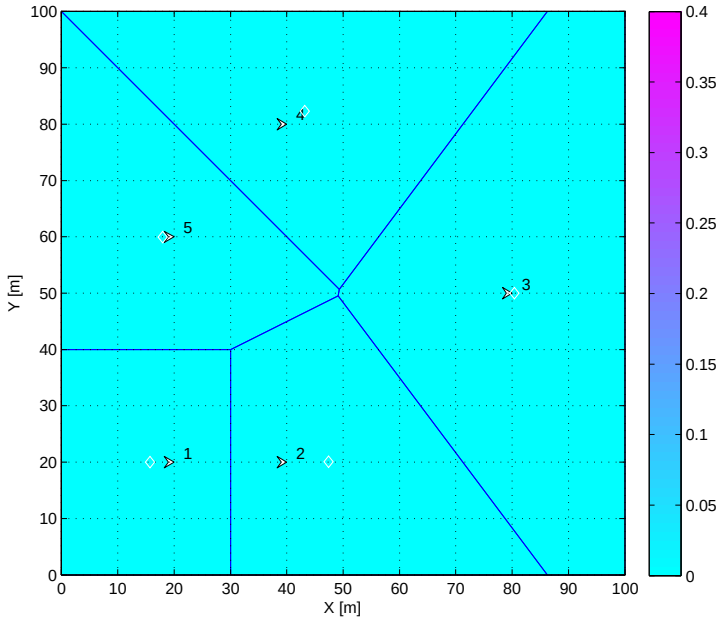


Figure 3.2: Initial positions of agents.

3.4.2 Coverage Performance

We assume $\partial\Omega_{\mathbf{\Phi}} = 0$ and consider Dirichlet boundary conditions, therefore assigning the density ϕ on the boundary $\partial\Omega_\phi \equiv \partial\Omega$. The four sides of Ω are labelled as $\partial\Omega_k, k = 1, \cdots, 4$, where the subscript 1 refers to the side $y = 0$, and the rest of the sequence is determined by the sides following in counter-clockwise direction.

In the first set of results which belong to scenarios with discontinuous boundary conditions, the time varying boundary density $\bar{\phi}(t)$ is defined by the piecewise constant (in time) function

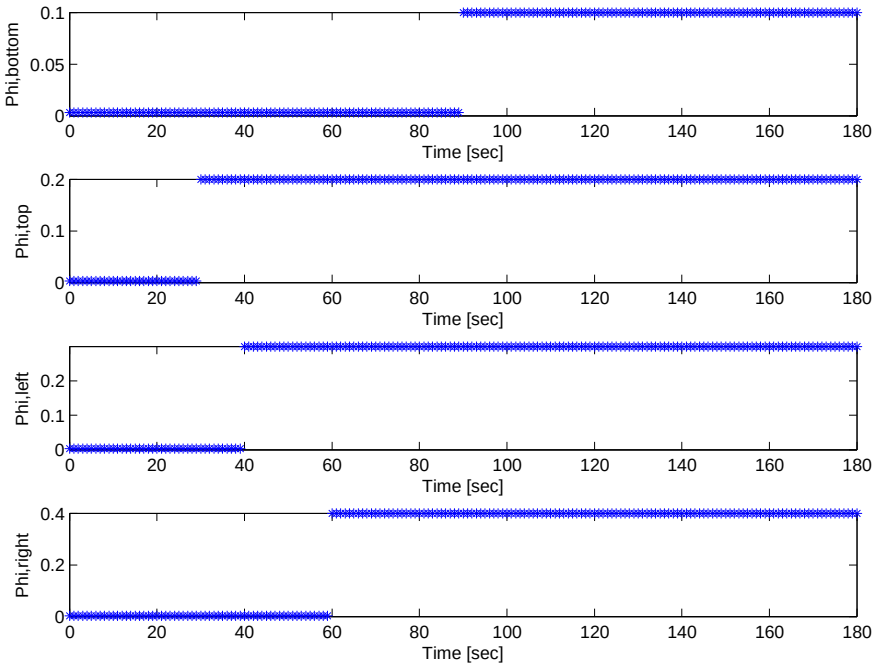
$$\bar{\phi}(t) = \frac{m_{kj}}{\mu(\partial\Omega_k)}(u(t - t_{j-1}) - u(t - t_j)) + \phi_0, \tag{3.27}$$

where m_{kj} quantifies the amount of risk on the boundary $\partial\Omega_k$ for $t \in [t_{j-1}, t_j]$, $j = 1, 2, 3, \cdots$, normalized with the measure (length) $\mu(\partial\Omega_k)$ of the side on which ϕ is prescribed. Moreover, $u(.)$ is the unit step function, which is 1 when the argument is greater than or equal to zero and 0 otherwise. Note that for this particular example, the term m_{kj} in the boundary condition (3.27) represents the number of asymmetric threats entering into the harbour's inner reaction zone through boundary $\partial\Omega_k$ in the time interval $[t_{j-1}, t_j]$, and the normalized term $m_{kj}/\mu(\partial\Omega_k)$ is interpreted as a uniform probability distribution of risk on the specific boundary, associated to the threats entering.

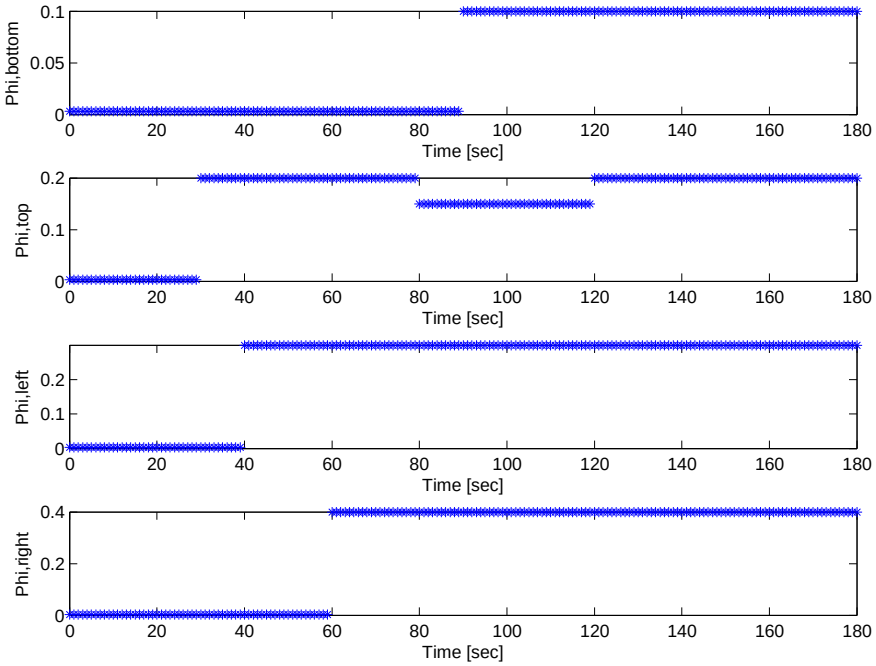
The time-varying risk densities prescribed at the four boundaries (top, bottom, left, and right) of the domain, Ω, are shown in Fig. 3.3a, and Fig. 3.3b for the first two scenarios with discontinuous boundary conditions. As can be seen, the main differences between these two sets are that in the second one, the risk density on the top boundary reduces to 75 percent of previous magnitude at $t = 80$ s and increases again at $t = 120$ s. At each sampling time instant, $\phi = \phi_0$ indicates minimum risk and $\phi = 1$ reflects the fact that boundary has the highest risk. However, highest risk (density) is weighted by the factor $m_{kj}/\mu(\partial\Omega_k)$ with $m_{1j} = 10$, $m_{2j} = 40$, $m_{3j} = 20$, $m_{4j} = 30$ The boundary length is $\mu(\partial\Omega_k) = 100$ m; for $k = 1, \cdots, 4$.

For the first scenario with discontinuous boundary conditions, the area coverage by fives agents at different time instants is summarized in Fig. 3.4, where the colour-bar represents the relative density level with blue (pink) indicating less (more) density level ($\phi = \phi_0$ for blue and $\phi = 0.4$ for pink). As can be seen in Fig. 3.3a, at time instant $t = 30$ s, the top boundary has the highest density (risk). The risk in the inner reaction zone is then updated according to the solution of the diffusion model (3.5). Fig. 3.4a shows the impact of the top boundary risk in the inner reaction zone at $t = 40$ s. The non-uniform coverage by the team of five agents subject to the time-varying densities in the inner reaction zone at time instants $t = 50$ s, $t = 70$ s, $t = 100$ s, $t = 140$ s, and $t = 180$ s are shown in Fig. 3.4b to Fig. 3.4f respectively. For clarity, the generalized Voronoi partitions are shown in all situations.

In the second scenario with discontinuous boundary conditions, the area coverage of agents at different time instants is demonstrated in Fig. 3.5. According to Fig. 3.3b, at time instant $t = 30$ s, the top boundary obtains the highest risk density but its risk level

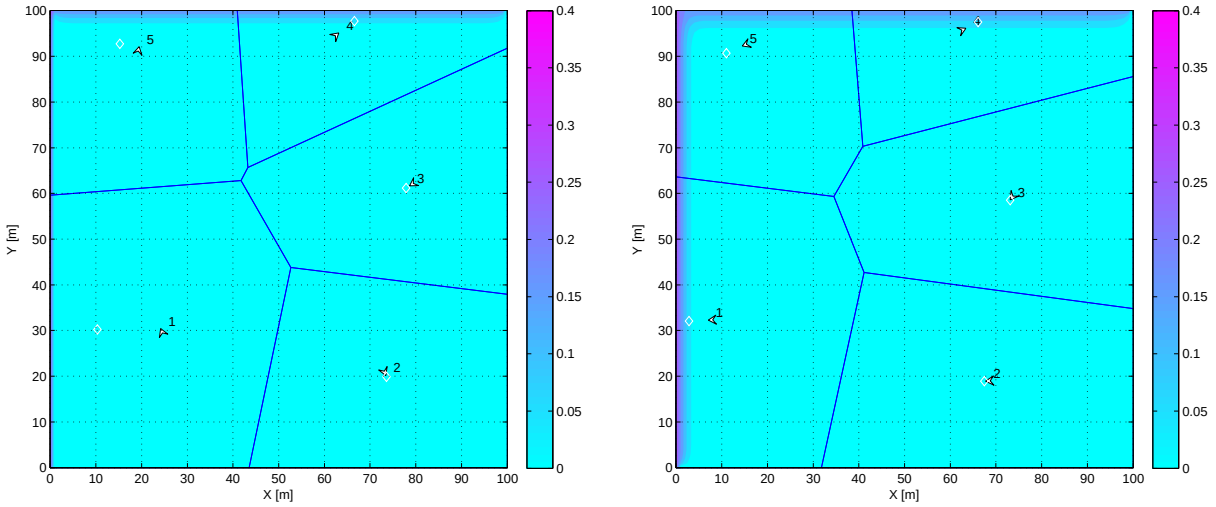


(a) The first discontinuous boundary conditions.

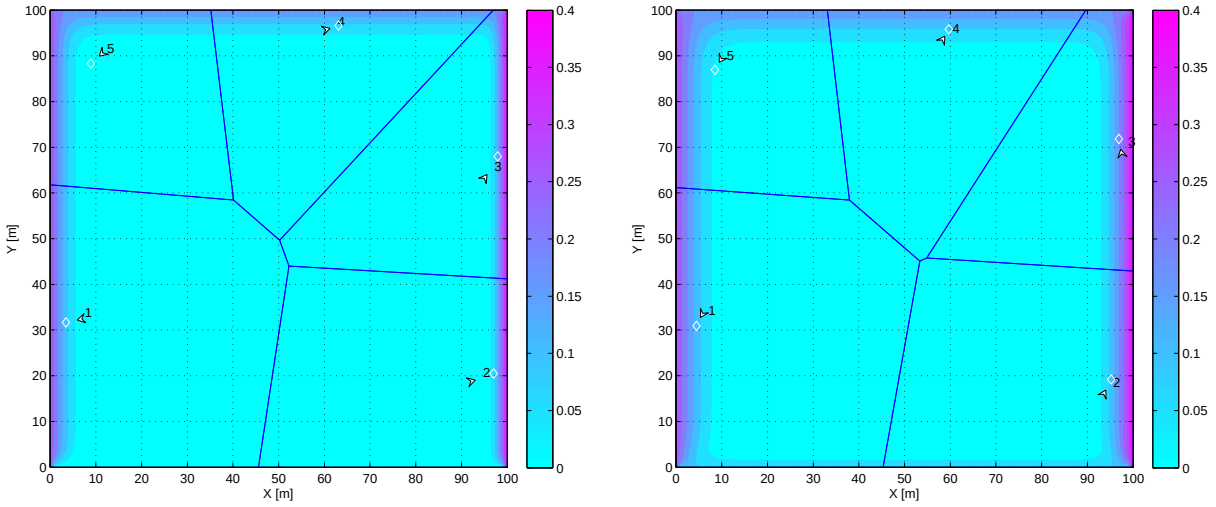


(b) The second discontinuous boundary conditions.

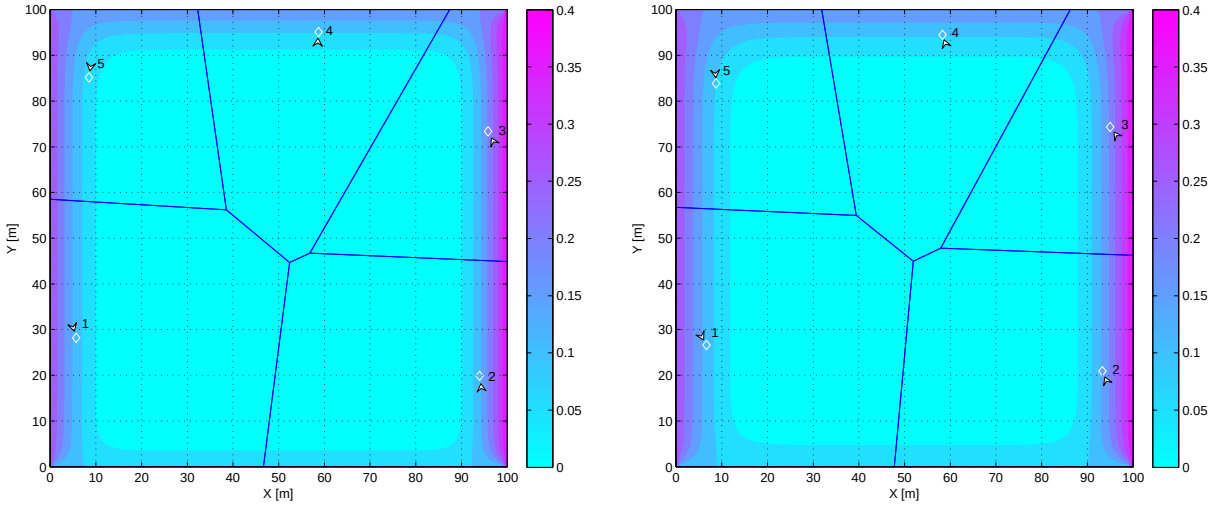
Figure 3.3: A comparison between discontinuous boundary conditions.



(a) $t = 40\ s$ (b) $t = 50\ s$

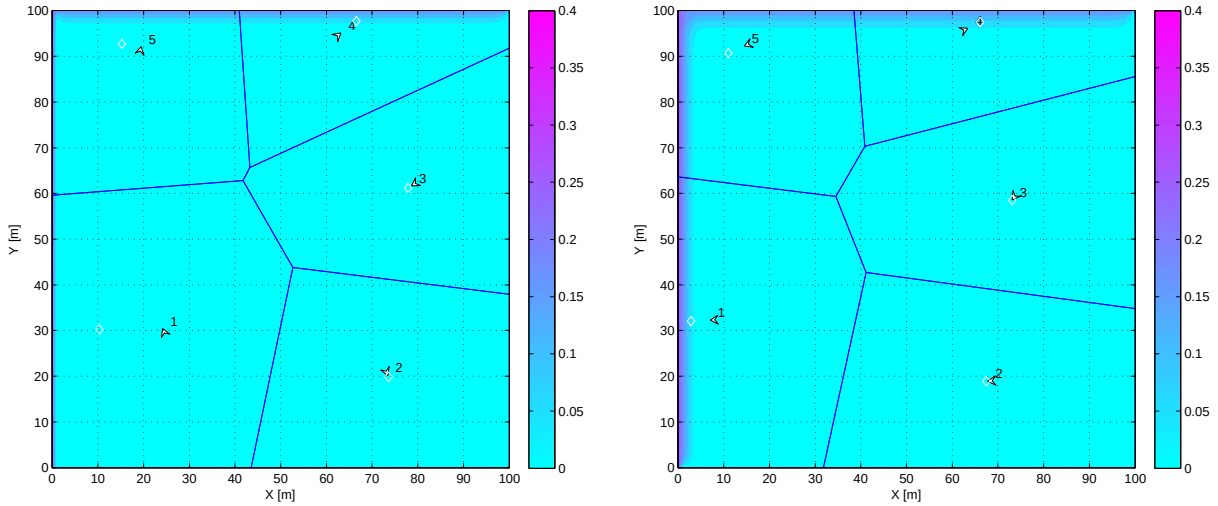


(c) $t = 70\ s$ (d) $t = 100\ s$

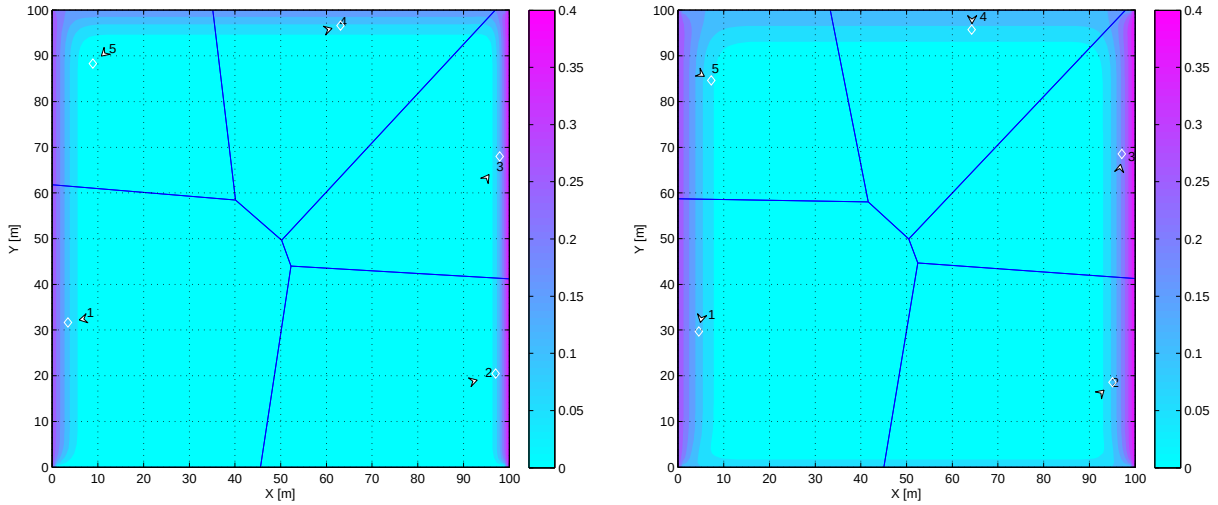


(e) $t = 140\ s$ (f) $t = 180\ s$

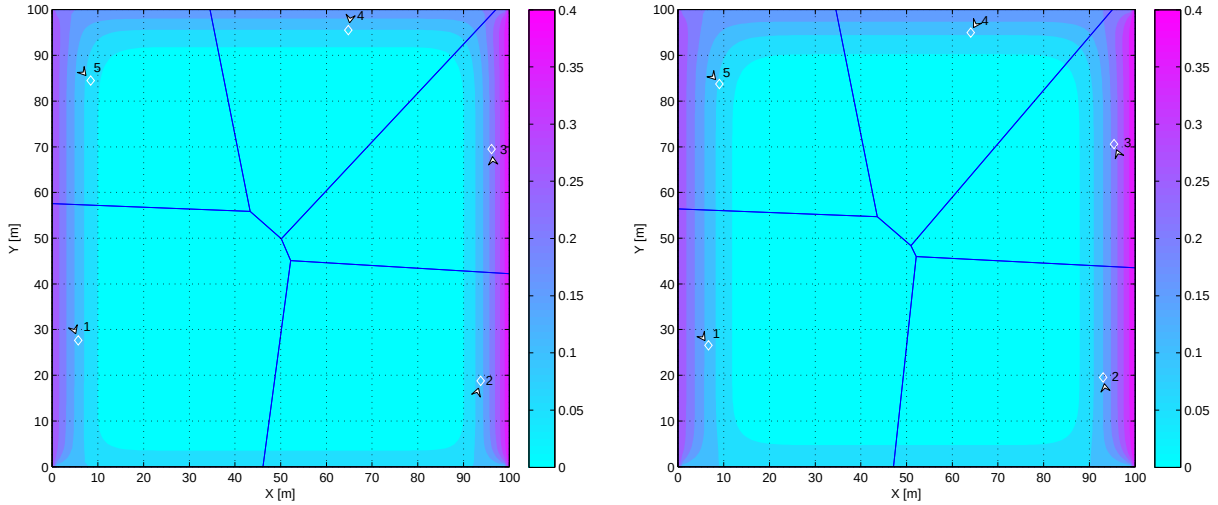
Figure 3.4: The area coverage of the first scenario with discontinuous boundary conditions (3.3a), by five agents at different time instants.



(a) $t = 40\ s$ (b) $t = 50\ s$



(c) $t = 70\ s$ (d) $t = 100\ s$



(e) $t = 140\ s$ (f) $t = 180\ s$

Figure 3.5: The area coverage of the second scenario with discontinuous boundary conditions (3.3b), by five agents at different time instants.

reduces suddenly at $t = 80\ s$ to 75 percent of previous magnitude and increases again at $t = 120\ s$. The risk in the domain is updated according to the solution of the diffusion model (3.5). The non-uniform coverage by the agents team, subjected to prescribed time-varying boundaries shown in Fig. 3.3b, at time instants $t = 40\ s$, $t = 50\ s$, $t = 70\ s$, $t = 100\ s$, $t = 140\ s$, and $t = 180\ s$ are shown in Fig. 3.5a to Fig. 3.5f respectively.

In the second two sets of results which are scenarios with continuous boundary conditions, the time-varying boundary conditions $\bar{\phi}(t)$ evolve as shown in Fig. 3.6a, and Fig. 3.6b. As can be observed, unlike the scenarios with discontinuous boundary conditions, any variation in the boundary conditions of second two scenarios takes place smoothly. The main difference between these two scenarios is that the risk level of the left boundary starts its reduction procedure at $t = 120\ s$ and stops at $t = 140\ s$ when the risk has 80 percent of the previous magnitude.

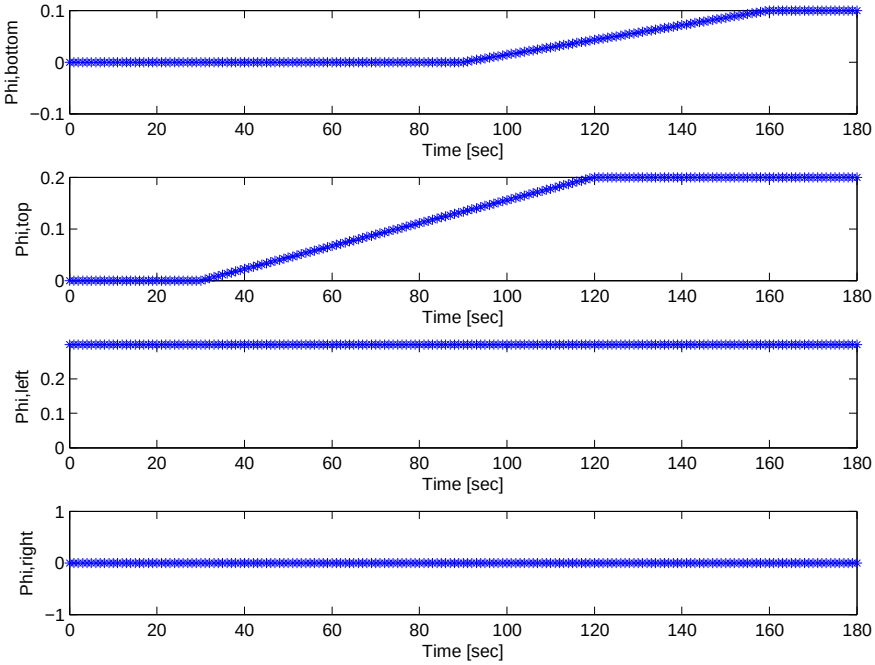
The area coverage by five agents at different time instants is shown in Fig. 3.7, for the first scenario with continuous boundary conditions. As can be seen, the left boundary risk has always its maximum amount and the right one is permanently equal zero. The risk levels of top and down boundaries grow at $t = 30\ s$ and $t = 90\ s$ respectively. The risk in the inner reaction zone is updated according to the solution of the diffusion model (3.5). Agents' positions in the inner reaction zone at time instants $t = 20\ s$, $t = 50\ s$, $t = 80\ s$, $t = 120\ s$, $t = 150\ s$, and $t = 180\ s$ are shown in Fig. 3.7a to Fig. 3.7f respectively.

In the second scenario with continuous boundary condition, the non-uniform coverage by the agents team, subjected to demonstrated time-varying boundaries, at time instants $t = 20\ s$, $t = 50\ s$, $t = 80\ s$, $t = 120\ s$, $t = 150\ s$, and $t = 180\ s$ are shown in Fig. 3.8a to Fig. 3.8f separately. The left boundary risk density has its maximum amount till $t = 120\ s$ and after that it starts the reduction procedure gradually till $t = 140\ s$ in which it has 80 percent of previous amount. The risk density of right boundary is always equal zero and the risk levels of top and down boundaries grow at $t = 30\ s$ and $t = 90\ s$ respectively. The risk in the inner reaction zone is updated with respect to the solution of the diffusion model (3.5). For clarity, at each time instant, the Voronoi cells are also shown.

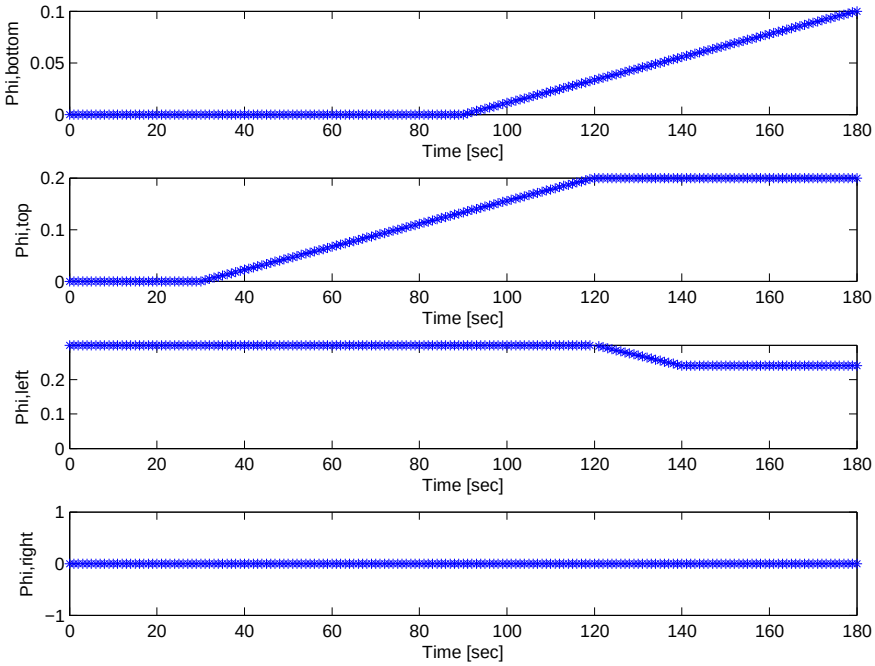
3.5 Discussion

Comparisons between absolute and partial time derivatives of coverage metrics related to discontinuous and continuous boundary conditions are shown in Fig. 3.9 and Fig. 3.10 respectively. As it is expected in the first scenario with discontinuous and with continuous boundary conditions, $\frac{\partial \mathcal{H}}{\partial t}$ and subsequently $\dot{\mathcal{H}}$, are positive as shown in Fig. 3.9a and Fig. 3.10a. The common specification of these two scenarios is having semi-monotonically increasing boundary conditions. In the second scenario with discontinuous and continuous boundary conditions, although the boundary conditions reduce, $\frac{\partial \mathcal{H}}{\partial t}$ and $\dot{\mathcal{H}}$ are still positive. It means that the reduction of boundary values in these sets are compensated by the proposed feedback law.

In all scenarios, in Fig. 3.4, Fig 3.5, Fig. 3.7 and Fig. 3.8, agents tend to distribute closer to the boundaries. This is a consequence of the non-uniform risk distribution during

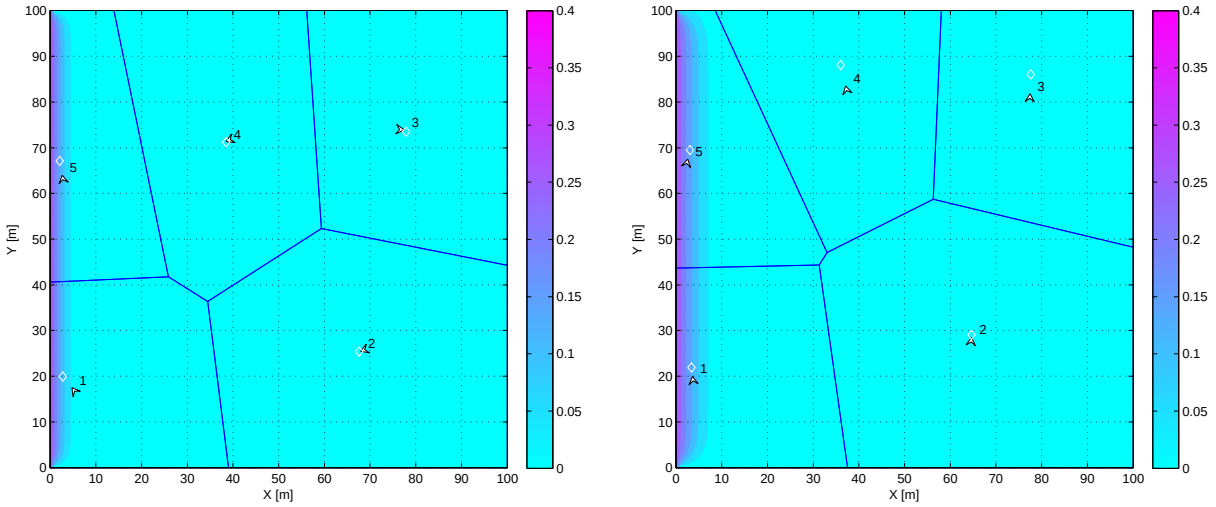


(a) The first continuous boundary conditions.

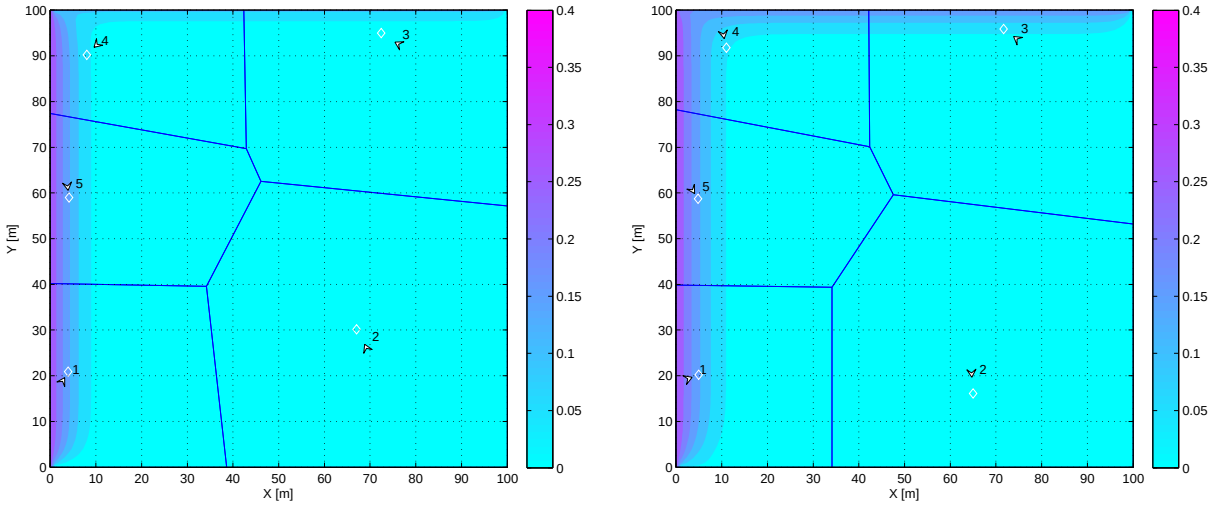


(b) The second continuous boundary conditions.

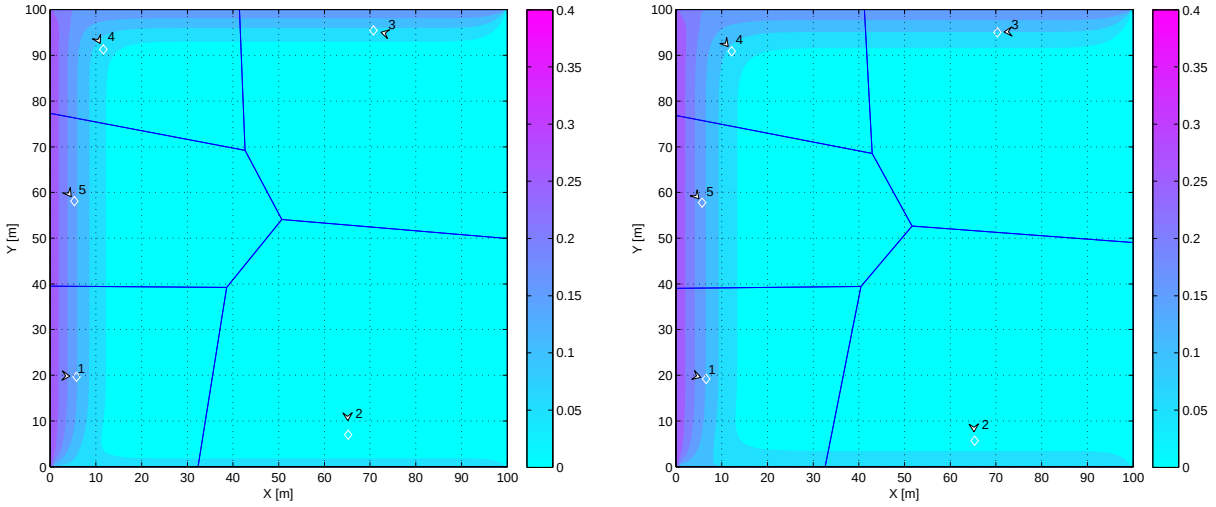
Figure 3.6: A comparison between continuous boundary conditions.



(a) $t = 20\ s$ (b) $t = 50\ s$

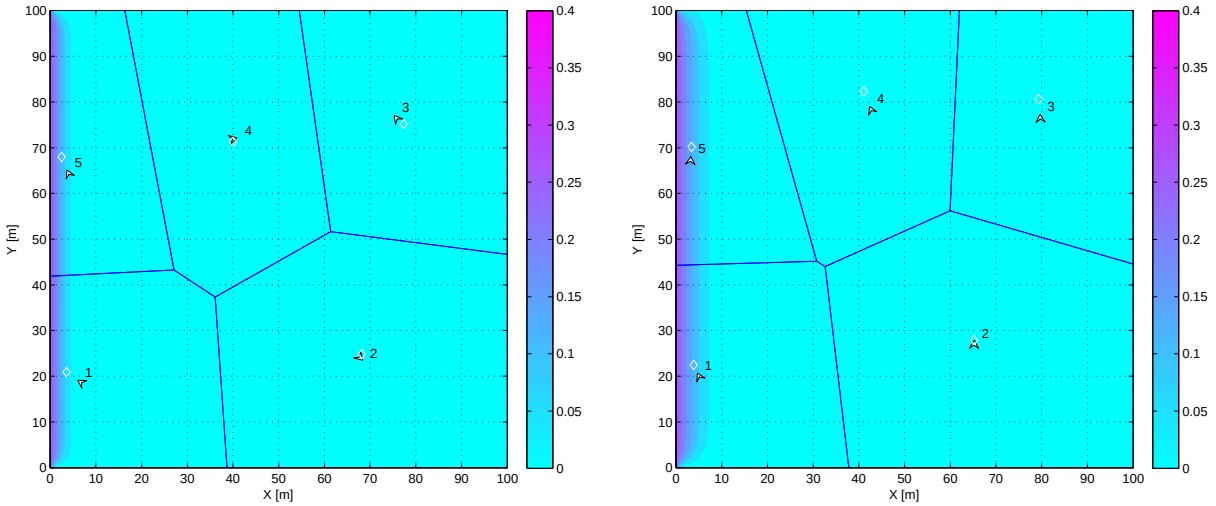


(c) $t = 80\ s$ (d) $t = 120\ s$

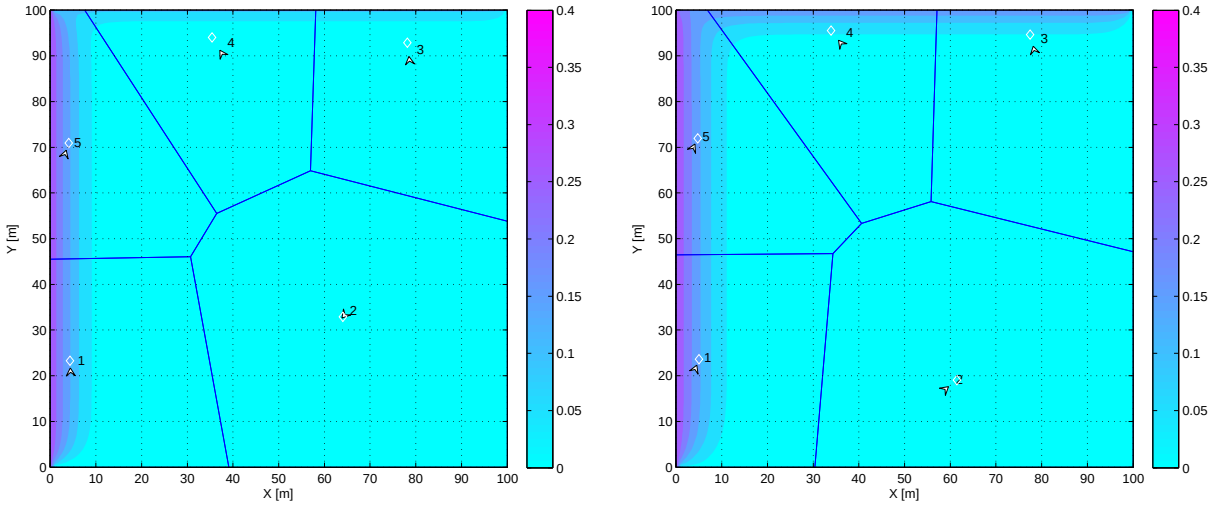


(e) $t = 150\ s$ (f) $t = 180\ s$

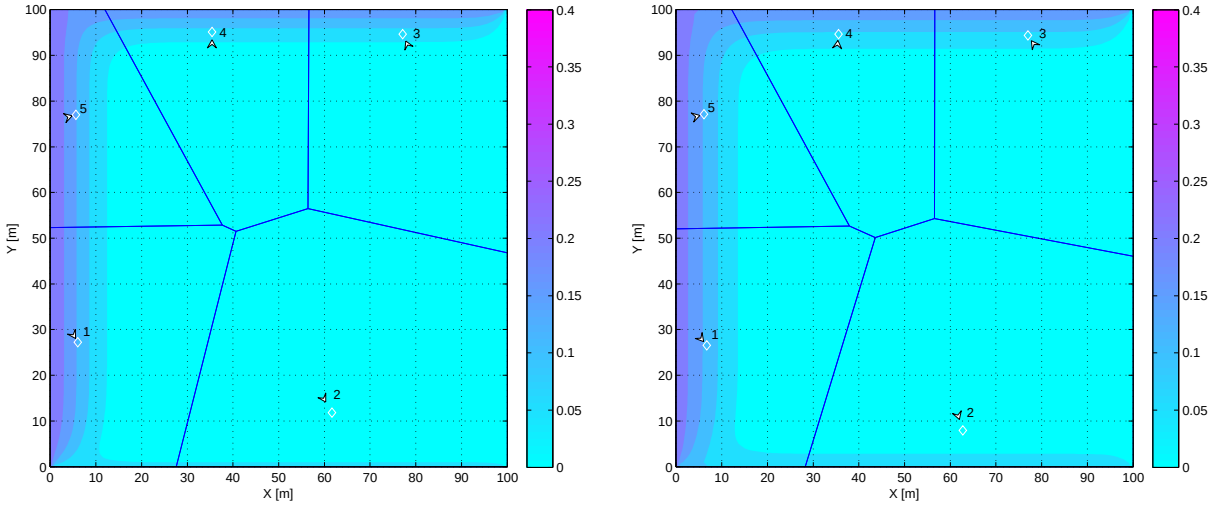
Figure 3.7: The area coverage of the first scenario with continuous boundary conditions (3.6a), by five agents at different time instants.



(a) $t = 20\ s$ (b) $t = 50\ s$



(c) $t = 80\ s$ (d) $t = 120\ s$

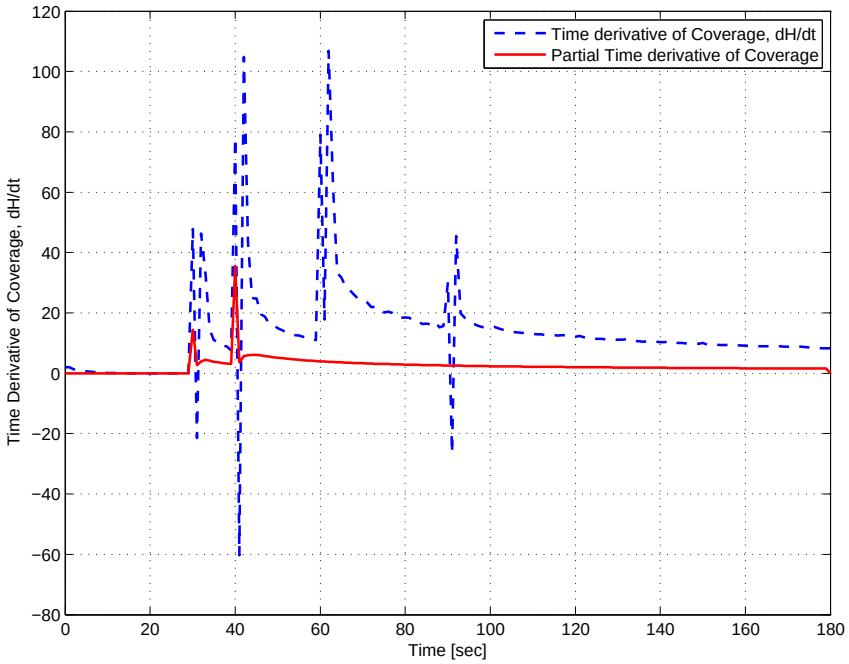


(e) $t = 150\ s$ (f) $t = 180\ s$

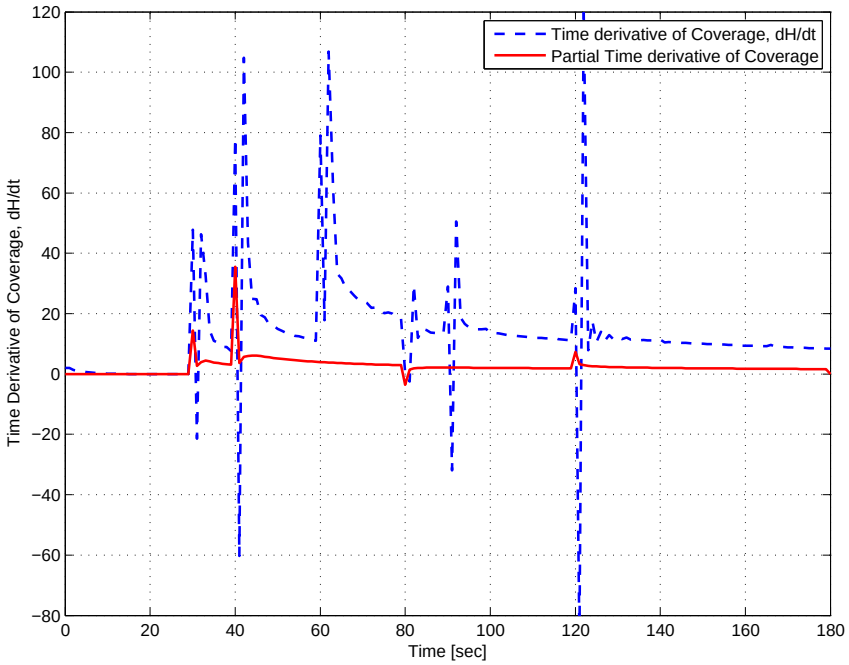
Figure 3.8: The area coverage of the second scenario with continuous boundary conditions (3.6b), by five agents at different time instants.

the transient of the diffusion generated by changing boundary conditions, which consistently with centroids defined in (2.14), imply a non-uniform distribution of agents which tend towards regions with higher risks.

Time derivatives of the coverage metric versus $\int_{\partial\Omega} \frac{\partial\bar{\phi}(\mathbf{q},t)}{\partial\mathbf{q}} \cdot \mathbf{n}\, d\mathbf{q}$ calculated for the cases with discontinuous and continuous boundary conditions, normalized with respect to their length, are compared in Fig. 3.13 and Fig. 3.14 respectively. It is clear that time derivative of the coverage metric reduces when $\int_{\partial\Omega} \frac{\partial\bar{\phi}(\mathbf{q},t)}{\partial\mathbf{q}} \cdot \mathbf{n}\, d\mathbf{q}$ decreases. Therefore, every variation in boundary conditions related to the term $\int_{\partial\Omega} \frac{\partial\bar{\phi}(\mathbf{q},t)}{\partial\mathbf{q}} \cdot \mathbf{n}\, d\mathbf{q}$, should be bounded to guarantee the stability of the agents' trajectories.

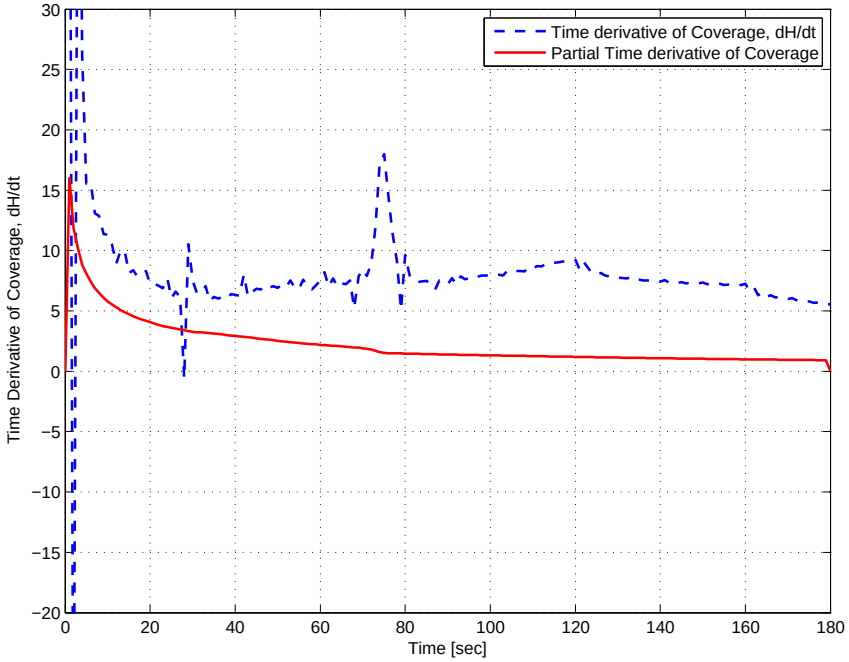


(a) The first scenario with discontinuous boundary conditions.(3.3a)

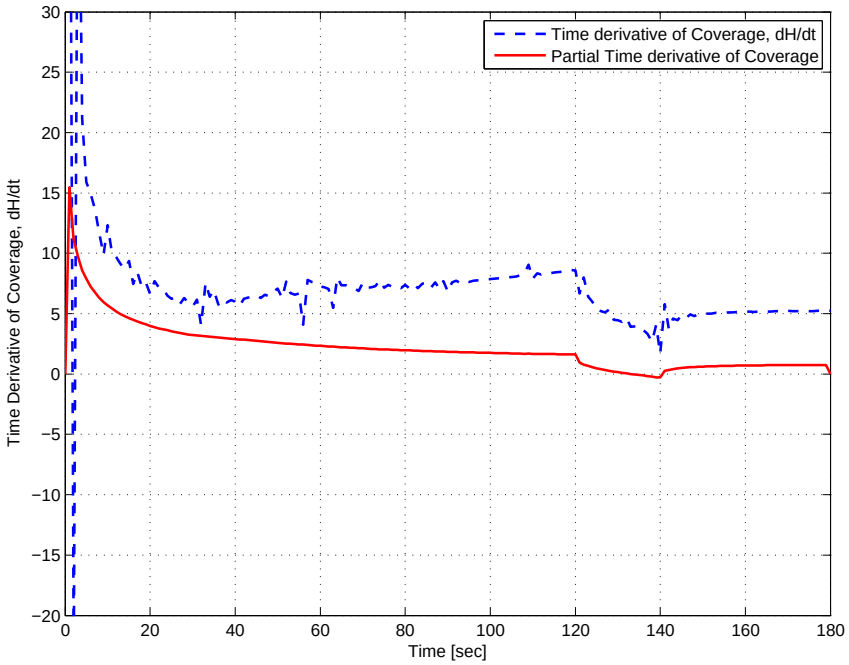


(b) The second scenario with discontinuous boundary conditions.(3.3b)

Figure 3.9: A comparison between absolute and partial time derivatives of the coverage metric of the scenarios with discontinuous boundary conditions.

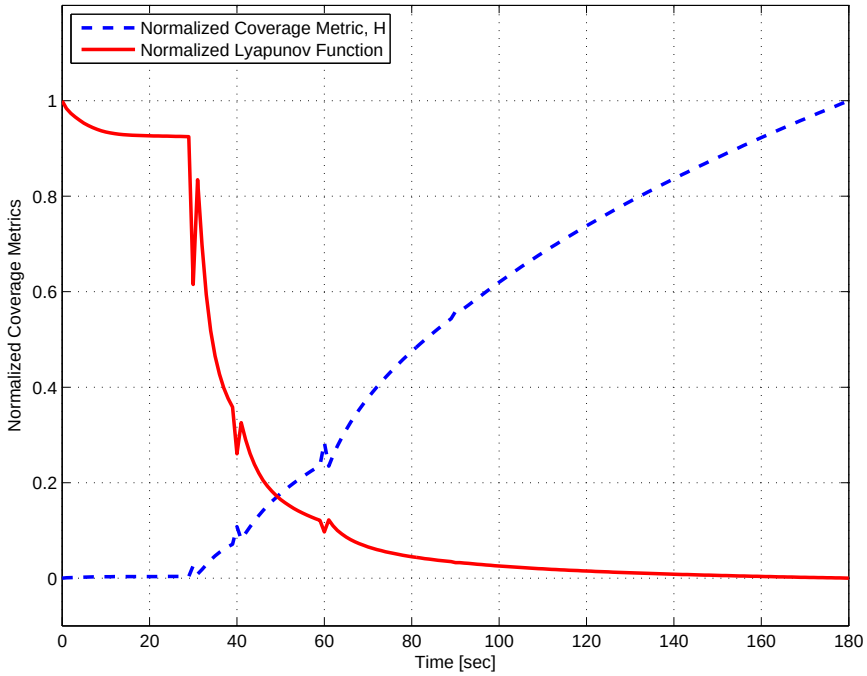


(a) The first scenario with continuous boundary conditions.(3.6a)

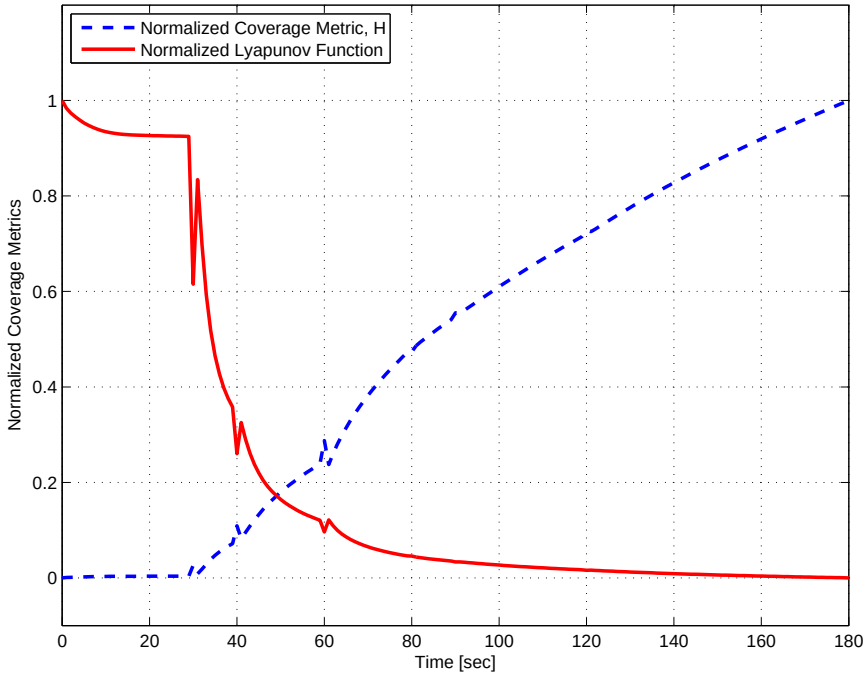


(b) The second scenario with continuous boundary conditions.(3.6b)

Figure 3.10: A comparison between absolute and partial time derivatives of the coverage metrics of the scenarios with continuous boundary conditions.

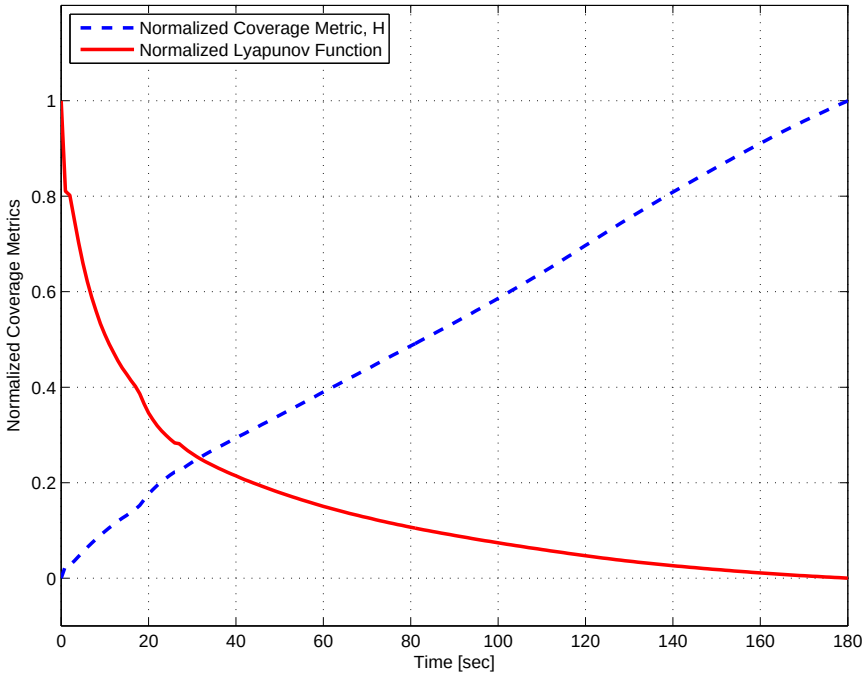


(a) The first scenario with discontinuous boundary conditions.(3.3a)

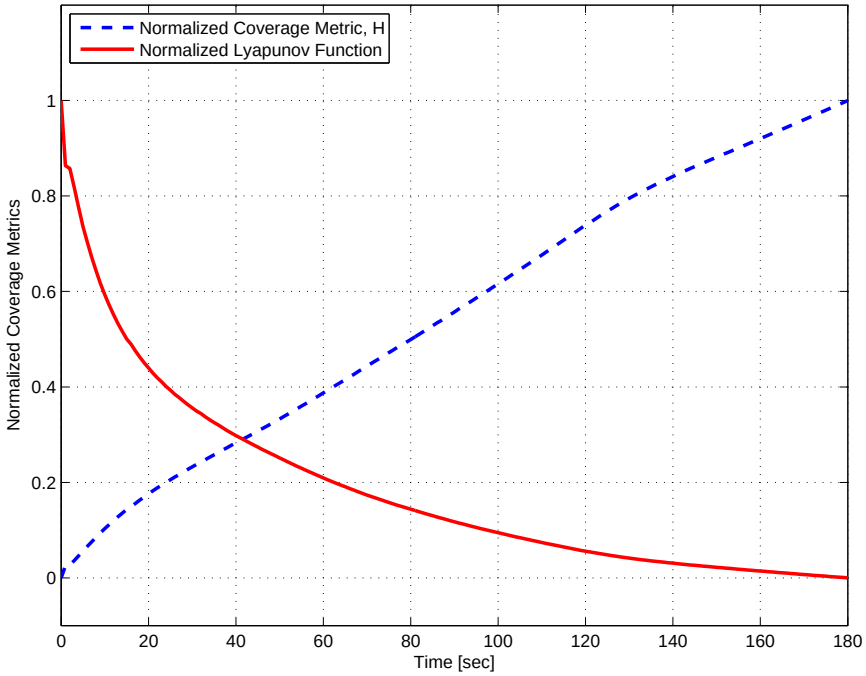


(b) The second scenario with discontinuous boundary conditions.(3.3b)

Figure 3.11: A comparison between total coverage metrics and Lyapunov functions of the scenarios with discontinuous boundary conditions.

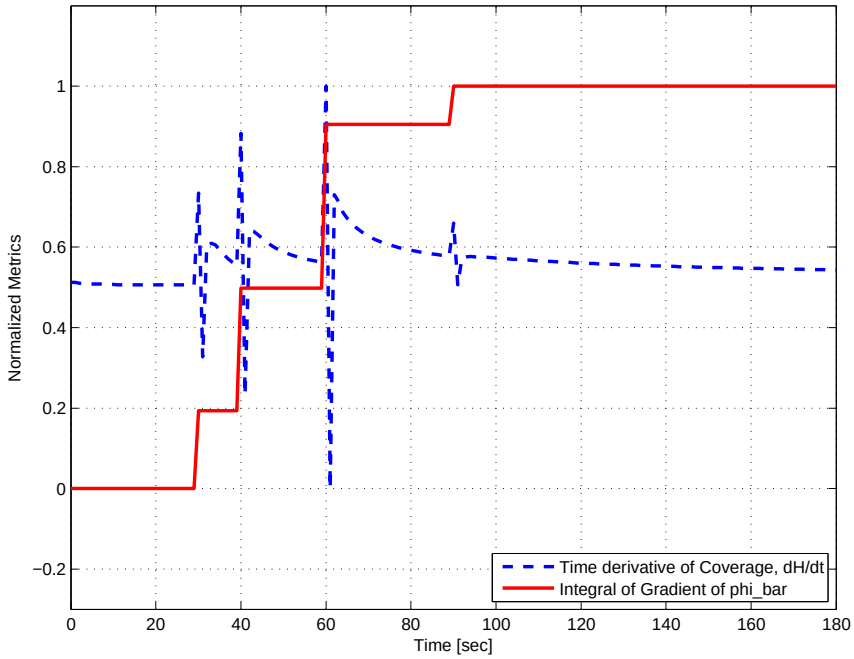


(a) The first scenario with continuous boundary conditions.(3.6a)

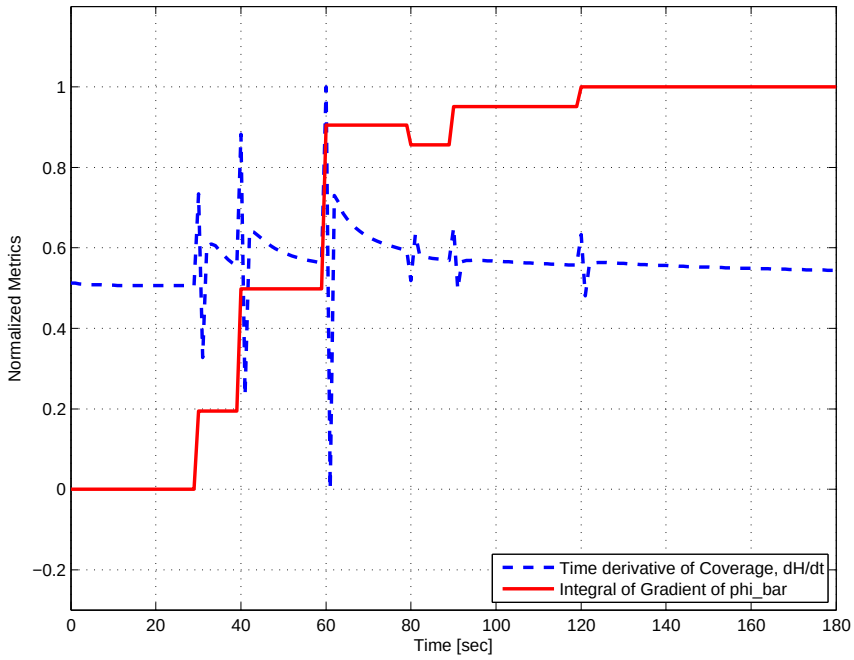


(b) The second scenario with continuous boundary conditions.(3.6b)

Figure 3.12: A comparison between total coverage metrics and Lyapunov functions of the scenarios with continuous boundary conditions.

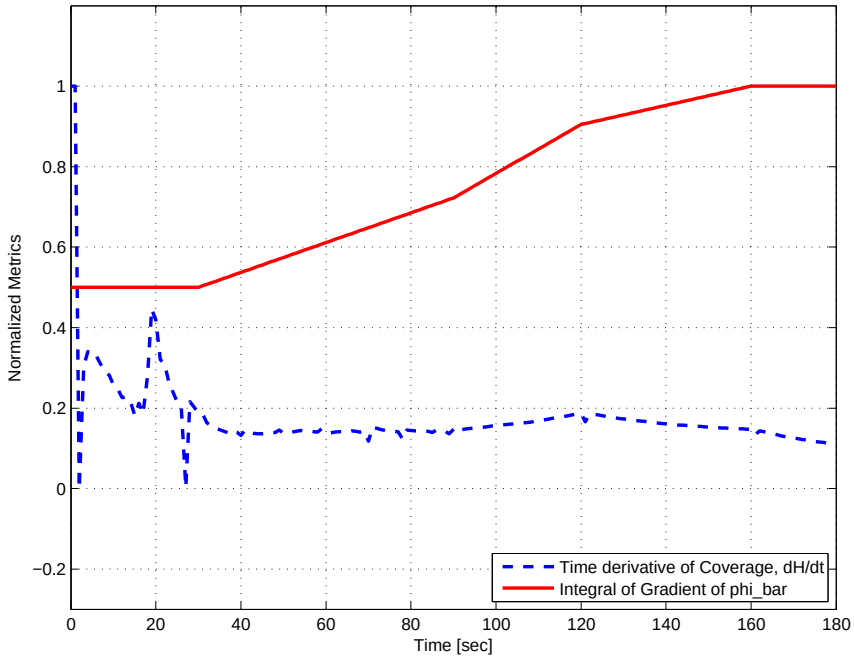


(a) The first scenario with discontinuous boundary conditions.(3.3a)

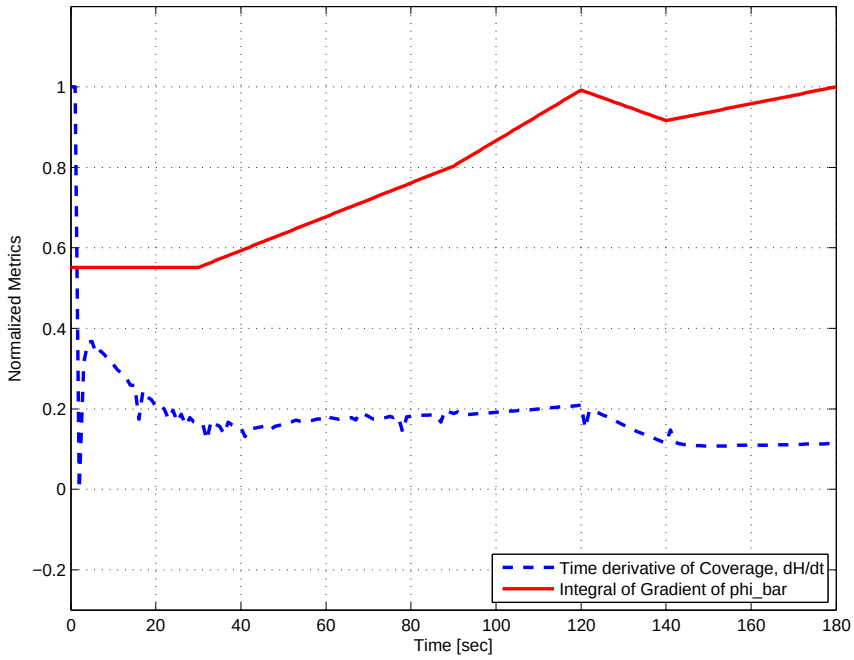


(b) The second scenario with discontinuous boundary conditions.(3.3b)

Figure 3.13: A comparison between time derivatives of the coverage metric and normalized $\int_{\partial\Omega} \frac{\partial\bar{\phi}(\mathbf{q},t)}{\partial\mathbf{q}} \cdot \mathbf{n}\, d\mathbf{q}$ of the scenarios with discontinuous boundary conditions.



(a) The first scenario with continuous boundary conditions.(3.6a)



(b) The second scenario with continuous boundary conditions.(3.6b)

Figure 3.14: A comparison between time derivatives of the coverage metric and normalized $\int_{\partial\Omega} \frac{\partial\bar{\phi}(\mathbf{q},t)}{\partial\mathbf{q}} \cdot \mathbf{n}\, d\mathbf{q}$ of the scenarios with continuous boundary conditions.

Chapter 4

Summary and Conclusions

4.1 Summary

This work contributed to the development of a non-uniform and non-autonomous coverage control algorithm for networked multi-agent systems, in which the evolution of the risk density obeys a conservation law in a spatial workspace where mobile agents are deployed.

In the absence of risk sources inside the domain, the evolution of the risk is triggered by time varying boundary conditions that represented averaged (over the boundary) quantities of external threats penetrating through the perimeter inside an area to be defended. Therefore, instead of assuming or estimating the kinematics of external threats, their effect is described in terms of boundary fields. By adopting a first gradient constitutive relation between the risk flux and the risk, a simple diffusion equation is obtained and numerically solved by a finite difference scheme. The solution is the space time description of the risk density inside the area.

A distributed velocity feedback law coupled with the risk diffusion was presented, with generated trajectories optimal with respect to a non-autonomous coverage metric $\mathcal{H}$, which plays the role of the objective function in this optimal control problem. The feedback law presented in this thesis work is distributed in the sense that it can be computed by each agent having only local information, that is its own Voronoi cell and the state of the risk inside it. However, the computation of Voronoi tessellations and the determination of the risk depend on global knowledge about the environment, and therefore it was assumed that each agent knows the states of the other agents in the group (to compute Voronoi cells) and the risk boundary conditions.

Due to the presence of time-varying risk density over the domain and by considering a non-autonomous coverage metric, Barbalat's Lemma was employed to study asymptotic convergence and optimality of the non-autonomous system, which dictates a constraint on boundary conditions for asymptotic stability.

Four sets of scenarios with different time-varying boundary conditions were simulated, by considering conditions that illustrate theoretical predictions.

4.2 Conclusions

By employing Barbalat's Lemma to study asymptotic convergence and optimality of the non-autonomous system, a criterion on time-varying boundary values should be met to guarantee the stability of agents' trajectories: any increase or decrease of boundary conditions should be bounded.

Relaxation of mentioned hypotheses in computing the feedback is the object of ongoing additional work, which is based on the introduction of embedded local observers that through a suitable information flow among agents can estimate the risk boundary conditions. Moreover, the effect of inter-agent intermittent communications, eventually with neighbour rules to group agents, is being investigated for the purpose of understanding the influence of these conditions on the asymptotic behaviour of the system, so that realistic conditions determined by environmental constraints and/or tactical situations can be realistically simulated.

On the other side, in this work, the agents are assumed as fully actuated, omnidirectional mass particles moving in a two dimensional domain and possessing two translational degrees of freedom. As it mentioned before, this kinematics may not be realistic for implementation in real life scenarios, where more complicated conditions have to be considered in order to realistically describe interactions of mobile agents with the environment. Typically, agents are under-actuated, and this constraints should be translated into constraints on evolution of boundary conditions.

Appendices

Appendix A

Matlab Code

A.1 The Main Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
close all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Some Constants %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nl = 100; %the number of steps in each direction
dl = 1; %length of each step in x or y direction (m)
Nt = 180; %the number of time steps
ts = 1; %Length of each time step (s)
tf= Nt*ts; %total time (s)
tin =  0:ts:tf; %time span
K =  0.25; %heat conduction coeficient
n = 5; %number of interceptors
alpha=5; %Sensors constant
beta=0.001; %Sensors constant
Umax = 4*0.514; %Maximum velocity of interceptors [knots]
eps=10^-14; %constant magnitude in finding neighberhoods
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Points in harbour %%%%%%%%%%%%%%%%%%%%%%%%%%%%
bounds = [0,100,100,0  ,;...
0,0  ,100,100]; %[m]
%%% Divide the area of interest into a set of discrete point set
x = 0:dl:Nl; % x axis range
y = 0:dl:Nl; % y axis range
pts = zeros(length(x)*length(y),2);%total # of possible discrete points
for i=1:length(x)
ptsi = [x(i)*ones(1,length(y));y];
pts((i-1)*length(y)+1:i*length(y),1) = (ptsi(1,:))';
pts((i-1)*length(y)+1:i*length(y),2) = (ptsi(2,:))';
```

```
end
%%% Find all the discrete set of points inside the harbour boundary
pts_harbour = [pts(:,1) pts(:,2)]; % discrete points in the harbour
npts = max(size(pts_harbour)); % number of points in the harbour area
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Phi_new=zeros(Nl+1,Nl+1,Nt+1); %dimention of risk density array
Phi_dot=zeros(Nl+1,Nl+1,Nt+1); %dimention of time derivative of risk
%%%%%%%%%%%%%%%%%%%% Definition of boundary condition %%%%%%%%%%%%%%%%%%%%
Bot=5;Rig=20;Top=10;Lef=15;
Normalized=Bot+Rig+Top+Lef;
for time=0:ts:tf
Phi = 0.003*ones(Nl+1,Nl+1); % Initial guess
tt=time/ts+1;
%bottom
if (time>=0) && (time<90)
Phi(:,1)=0;
elseif ((time>=90) && (time<160))
Phi(:,1)=(Bot/Normalized)/70*time-(9/7)*(Bot/Normalized);
else
Phi(:,1)=(Bot/Normalized)*1;
end
%top
if ((time>=0) && (time<30))
Phi(:,Nl+1)=0;
elseif ((time>=30) && (time<120))
Phi(:,Nl+1)=(Top/Normalized)/90*time-(1/3)*(Top/Normalized);
else
Phi(:,Nl+1)=(Top/Normalized)*1;
end
%left
Phi(1,:)=(Lef/Normalized)*1;
%right
Phi(Nl+1,:)=0;
Phi_new(:,:,tt)=Phi;
bottom(tt)=Phi_new(3,1,tt);
top(tt)=Phi_new(3,Nl+1,tt);
left(tt)=Phi_new(1,3,tt);
right(tt)=Phi_new(Nl+1,3,tt);
grad(tt)=sum(Phi_new(:,1,tt)-Phi_new(:,2,tt))+...
sum(Phi_new(:,Nl+1,tt)-Phi_new(:,Nl,tt))+sum(Phi_new(1,:,tt)-...
Phi_new(2,:,tt))+sum(Phi_new(Nl+1,:,tt)-Phi_new(Nl,:,tt));
end
%%%%%%%%%%%%%%%%%%%% Drawing the boundary conditions %%%%%%%%%%%%%%%%%%%%
figure;
subplot(4,1,1)
```

```
plot (tin,bottom,'*')
xlabel('Time [sec]');
ylabel('Phi,bottom');
subplot(4,1,2)
plot (tin,top,'*')
xlabel('Time [sec]');
ylabel('Phi,top');
subplot(4,1,3)
plot (tin,left,'*')
ylim([0,0.3]);
xlabel('Time [sec]');
ylabel('Phi,left');
subplot(4,1,4)
plot (tin,right,'*')
xlabel('Time [sec]');
ylabel('Phi,right');
savefilename=...
('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\Boundary_Condition');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
saveas(gcf, savefilename, 'pdf');
%%%%%%%%%%%%%%%%%%%%%%%%% Risk density function %%%%%%%%%%%%%%%%%%%%%%%%%
maxiter = 500; %maximum number of itteration
for iter = 1:maxiter
Phi_last = Phi_new; %Save the last guess
for i=2:Nt+1,
depth_2D = (Phi_new(1:end-2,2:end-1,i-1)+...
Phi_new(3:end,2:end-1,i-1)-4*Phi_new(2:end-1,2:end-1,i-1)+...
Phi_new(2:end-1,1:end-2,i-1)+Phi_new(2:end-1,3:end,i-1))/dl^2;
time_1D = K*depth_2D;
Phi_new(2:end-1,2:end-1,i)=time_1D*ts+Phi_new(2:end-1,2:end-1,i-1);
end
%Find difference between last two solutions
err(iter) = max(abs(Phi_new(:)-Phi_last(:)));
if err(iter)<1E-4
break; % Stop if solutions very similar, we have convergence
end
end
%%%%%%%%%%%%%%%%%% Derivative of risk density function %%%%%%%%%%%%%%%%%%
Phi_dot(:,:,1)=0;
for t=2:tf
Phi_dot(:,:,t)=(Phi_new(:,:,t)-Phi_new(:,:,t-1))/ts;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for t=1:size(tin,2)
```

```
for i=1:size(Phi_new,2)
for j=1:size(Phi_new,2)
Phi((i-1)*(Nl+1)+j,t)=Phi_new(i,j,t);
Phidot((i-1)*(Nl+1)+j,t)=Phi_dot(i,j,t);
end
end
end
if iter==maxiter;
warning('Convergence not reached') %#ok<WNTAG>
end
%%%%%%%%%%%%% appropriate risk density function for drawing %%%%%%%%%%%%%
for i=1:size(Phi_new,2)
for j=1:size(Phi_new,2)
Phi_con(i,j,:)=Phi_new(j,i,:);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Start of algorithm %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% initial states of interceptors
xx = [20,40,80,40,20];
yy = [20,20,50,80,60];
intcpt_pos = [xx;yy];
intcpt_velocity = zeros(2,n);
init_state = [intcpt_pos;intcpt_velocity];
%%% initialize some matric for state-space equations
L = [1 0 0 0;0 1 0 0];
%%% Q1 means how fast we want to converge to "state".
Q1 = eye(4);
%%% Q2 means how fast we want to converge to "position".
Q2 = diag([1,1]);
L_bar = eye(4)-L'*((L*L')\L);
Q_bar = L_bar'*Q1*L_bar + L'*Q2*L;
%%% Interceptor's system matrices
Phik= [1 0 ts 0;...
0 1 0 ts;...
0 0 1 0;...
0 0 0 1];
%%%%%%%%%%%%% Finding Voronois' borders, mass and centroids %%%%%%%%%%%%%
directions = zeros(1,n);
pointsize = 14;
aviobj=avifile...
('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\example.avi',...
'compression','None','fps',10);
fig=figure;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAIN LOOP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:size(tin,2)% t=0:ts:tf
```

```
par_c_par_p=zeros(2*n,2*n);
clf
%%% Draw the risk function as a function of time:
contourf(0:100,0:100,Phi_con(:,:,k),'edgecolor','none')
colormap(cool(64)); % in cool colorbar
caxis([0 0.4]);
colorbar
hold on
%%% Draw the Voronois with respect to interceptors' Positions:
regions = voronoi_andrew(bounds,intcpt_pos);
drawRegionsMostafa(bounds, regions, directions);
axis square
xlim([0,100]);
ylim([0,100]);
xlabel('X [m]');
ylabel('Y [m]');
for i=1:n
pt = regions{i,1}; % interceptor position
tmp = regions{i,3}; % interceptor Voronoi
verticei = tmp(1:end,1:2); % [(x1,y1);(x2,y2);....; (xm,ym)]
I=[];
for aa=1:(size(verticei,1)-1)
for bb=aa+1:(size(verticei,1))
if norm(verticei(aa,:)-verticei(bb,:))<eps
I=[I,bb];
end
end
end
I=unique(I,'first');
vertice_i=removerows(verticei,I);
%%%%%%%%%%%% find centroid of i-th voronoi polygon %%%%%%%%%%%%
in_Vi = inpolygon(pts_harbour(:,1),pts_harbour(:,2),...
[vertice_i(:,1);vertice_i(1,1)],[vertice_i(:,2);vertice_i(1,2)]);
pts_Vi = [pts_harbour(in_Vi,1) pts_harbour(in_Vi,2)];
rho_Vi = Phi(in_Vi,k);
rho_dot_Vi = Phidot(in_Vi,k);
%%% Generalized rho %%%
ri2 = (pts_Vi(:,1)-pt(1)).^2+(pts_Vi(:,2)-pt(2)).^2;
rho_bar = (2*alpha*beta*exp(-beta*ri2)).*rho_Vi;
%%% compute generalized mass %%%
risk_sum_Vi = sum(rho_bar); % sum of risk (density) of polygon Vi
%%% compute generalized centroid %%%
x_rho_Vi = (1/risk_sum_Vi)*sum(pts_Vi(:,1).*rho_bar);
y_rho_Vi = (1/risk_sum_Vi)*sum(pts_Vi(:,2).*rho_bar);
cm = [x_rho_Vi y_rho_Vi];
```

```
%%% compute generalized L(where c=L/m) %%%
Lx_rho_Vi = sum(pts_Vi(:,1).*rho_bar);
Ly_rho_Vi = sum(pts_Vi(:,2).*rho_bar);
Lm = [Lx_rho_Vi Ly_rho_Vi];
%%% Plot the generalized centroid of Voronoi %%%
op = plot(cm(1),cm(2),'wd');
delx = cm(1)-pt(1);
dely = cm(2)-pt(2);
del = [delx;dely];
directions(i) = atan2(dely,delx);
% Distance between Interceptor and Corresponding Voronoi Centroid
Tas(i,k)=norm(del);
% partial derivative of H with respect to time
parH_part=sum(alpha*exp(-beta*ri2).*rho_dot_Vi);
dH_dt(k)= parH_part;
%gamma
gamma_x=(-delx*parH_part)/(risk_sum_Vi*(norm(del)^2));
gamma_y=(-dely*parH_part)/(risk_sum_Vi*(norm(del)^2));
gamma=[gamma_x gamma_y];
%%%%%%%%%% 5.Feedback Law (Appropriate Velocity of Interceptor) %%%%%%%%%%
x_tilde = L'*((L*L')\(cm'));
gamma_tilde = L'*((L*L')\(gamma'));
u_bar_k = Q_bar*(x_tilde-init_state(:,i))+ gamma_tilde;
%%%%%%%%%%%%%%%%%%%%%%%%% 3.Continue of algorithm %%%%%%%%%%%%%%%%%%%%%%%%%
init_state(:,i) = Phik*init_state(:,i) + ts*u_bar_k;
intcpt_pos(:,i) = L*init_state(:,i);
h_Vori = (alpha*exp(-beta*ri2)).*rho_Vi;
H_Vori(i) = sum (h_Vori);
end
if k==1
savefilename =...
('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\Initial_Position');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
if k==20*ts+1
savefilename =('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\t=20');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
if k==50*ts+1
savefilename =('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\t=50');
saveas(gcf, savefilename, 'pdf');
```

```
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
if k==80*ts+1
savefilename =('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\t=80');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
if k==120*ts+1
savefilename=('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\t=120');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
if k==150*ts+1
savefilename=('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\t=150');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
if k==180*ts+1
savefilename=('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\t=180');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
end
Tas_Kol(k)=sum(Tas(:,k));
H_Vi(k) = sum (H_Vori);
F = getframe(fig);
aviobj = addframe(aviobj,F);
end
close(fig);
aviobj = close(aviobj);
%%%% Draw Errors %%%%
m=ceil(n/4);
for i = 1:m
figure;
Nom=(i-1)*4;
subplot(2,2,1)
plot (tin,Tas(1+Nom,:),'b-')
xlabel('Time [sec]');
ylabel('Distance [m]')
title (['Distance between ',num2str(1+Nom),...
'-th interceptor and corresponding centroid'])
```

```
if Nom+2>n, break, end
subplot(2,2,2)
plot (tin,Tas(2+Nom,:),'b-')
xlabel('Time [sec]');
ylabel('Distance [m]')
title (['Distance between ',num2str(2+Nom),...
'-th interceptor and corresponding centroid'])
if Nom+3>n, break, end
subplot(2,2,3)
plot (tin,Tas(3+Nom,:),'b-')
xlabel('Time [sec]');
ylabel('Distance [m]')
title (['Distance between ',num2str(3+Nom),...
'-th interceptor and corresponding centroid'])
if Nom+4>n, break, end
subplot(2,2,4)
plot (tin,Tas(4+Nom,:),'b-')
xlabel('Time [sec]');
ylabel('Distance [m]')
title (['Distance between ',num2str(4+Nom),...
'-th interceptor and corresponding centroid'])
savefilename =...
(['E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\Error set ',...
num2str(i)]);
saveas(gcf, savefilename, 'fig');

end
savefilename=...
(['E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\Error set ',...
num2str(i)]);
saveas(gcf, savefilename, 'fig');
figure;
hold on
plot(tin,smooth(Tas(1,:)),'LineWidth',1.5);
plot(tin,smooth(Tas(2,:)),'g--','LineWidth',2);
plot(tin,smooth(Tas(3,:)),'y.-','LineWidth',2.5);
plot(tin,smooth(Tas(4,:)),'--r','LineWidth',2.5);
plot(tin,smooth(Tas(5,:)),'-k','LineWidth',1.5);
h=legend('i=1','i=2','i=3','i=4','i=5',5);
set(h,'Interpreter','none')
xlabel('Time [sec]');
ylabel('Errors');
ylim([-2,18]);
grid on
hold off
```

```
savefilename =('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\errors');
saveas(gcf, savefilename, 'fig');
saveas(gcf, savefilename, 'pdf');
%%%% Draw Coverage and Lyapunov %%%%
figure;
Lyapunov=1./H_Vi;
H_Vi_N=H_Vi/(max(H_Vi));
Lyapunov_N=Lyapunov/(max(Lyapunov));
plot(tin,H_Vi_N,'b--',tin,Lyapunov_N,'-r','LineWidth',2);
grid on
xlabel('Time [sec]');
ylabel('Normalized Coverage Metrics');
legend('Normalized Coverage Metric, H',...
'Normalized Lyapunov Function','Location','northwest');
ylim([-0.1,1.2]);
savefilename =('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\Coverage');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
saveas(gcf, savefilename, 'pdf');
%%%% Draw Total Error %%%%
figure;
plot(tin,Tas_Kol,'b-');
grid on
xlabel('Time [sec]');
ylabel('Total Error');
title ('Total Error');
savefilename =...
('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\Total Error');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
saveas(gcf, savefilename, 'pdf');
%%%% Draw Coverage %%%%
figure
TAS=smooth(Tas_Kol);
TAS_N=TAS/(max(TAS));
plot(tin,H_Vi_N,'-r',tin,TAS_N,'--k','LineWidth',1.5)
grid on
hh=legend('Total Coverage, H','Total Error',2);
set(hh,'Interpreter','none')
xlabel('Time [sec]');
ylabel('Normalized Metrics');
ylim([-0.1,1.05]);
savefilename=('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\HandError');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
```

```
saveas(gcf, savefilename, 'pdf');
H_Dot_Vi=(H_Vi(2:end)-H_Vi(1:end-1))/ts;
H_Dot_Vi=[H_Dot_Vi(1),H_Dot_Vi];
H_Dot_Vi_N = H_Dot_Vi/(max(H_Dot_Vi));
%%%%% Draw Time derivative of Coverage and grad Phi %%%%%
grad_N=grad/(max(grad));
y=0;
plot(tin,H_Dot_Vi_N,'--b',tin,grad_N,'-r','LineWidth',2)
grid on
hhh=legend('Time derivative of Coverage, dH/dt',...
'Integral of Gradient of phi_bar','Location','southeast');
set(hhh,'Interpreter','none')
xlabel('Time [sec]');
ylabel('Normalized Metrics');
ylim([-0.8,1.1]);
savefilename =...
('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\H_dotandGrad');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
saveas(gcf, savefilename, 'pdf');
%%%%% Draw Time derivative and Partial Time derivative of Coverage %%%%%
plot(tin,H_Dot_Vi,'--b',tin,dH_dt,'-r','LineWidth',1.5)
grid on
hhhh = legend('Time derivative of Coverage, dH/dt',...
'Partial Time derivative of Coverage','Location','northeast');
set(hhhh,'Interpreter','none')
xlabel('Time [sec]');
ylabel('Time Derivative of Coverage, dH/dt');
ylim([-20,30]);
savefilename =('E:\My Courses\Thesis\02-Codes\DiffusiveDensityD\H_dot');
saveas(gcf, savefilename, 'fig');
print('-depsc2','-r300',[savefilename, '.eps']);
saveas(gcf, savefilename, 'pdf');
```

A.2 Drawing Voronoi Diagrams

A.2.1 Voronoi Partition Generator

```
% VORONOI   ordinary Voronoi partition generator
%   VORONOI will take in a bounding region along with a set of generator
%   points and produce the ordinary Voronoi partition defined by
%       |q - p_i| <= |q - p_j|
%   for all j ~= i.
```

```
%   ***** Inputs *****
%   bounds - a 2 x n_b array specifying the positively oriented bounding
%       polygon in 2D
%   points - the 2 x n_p array specifying generator points.
%   ***** Outputs *****
%   regionData - a n_p x 3 cell with:
%       -- the {i,1} element containing generator point information,
%       -- the {i,2} element containing the arc information. For the
%          ordinary Voronoi partition, this is always empty.
%       -- the {i, 3} element containing edge segments. These are
%          oriented, and each row represents a segment:
%          [x1, y1, x2, y2]
%          And the region is always to the left of the vector from (x1, y1)
%          to (x2, y2).
function regionData = voronoi_andrew(bounds, points)
n = size(points, 2);
regionData = cell(n, 3);
for i = 1:n
p1 = points(:,i);
regionData{i, 1} = p1;
vi = bounds;
for j = 1:n
if (j ~= i)
p2 = points(:, j);
[s1, s2] = bisectPlane(p1, p2);
vi = intersect_andrew(vi, s1(1:2), s2(1:2));
end
end
% Create the edges from the polygon data
edges = zeros(size(vi, 2), 4);
vi = [vi, vi(:, 1)]';
for j = 1:size(edges, 1)
edges(j, :) = [vi(j, :), vi(j+1, :)];
end
regionData{i, 2} = [];
regionData{i, 3} = edges;
end
% Helper function to determine the bisection plane
function [segStart, segEnd] = bisectPlane(p1, p2)
d = p2 - p1;
c = [-d(2); d(1)];
segStart = 0.5 * d + p1;
segEnd = segStart + c;
```

A.2.2 Find Geometric Intersection of a Polygon and Half-plane

```
%   ***** Inputs *****
%   polygon - a 2 x n array defining the vertices of a polygon in 2D
%   b0 - a point on the dividing line
%   b1 - a second point on the dividing line.  The vector b1-b0 defines the
%        intersecting halfplane.  The intersection of the ray with the
%        polygon would define a new positvely oriented edge.  If the polygon
%        is to the left of the ray, the intersection is the entire polygon.
%        If the polygon is to the right of the ray, the intersection is the
%        null set.
%   ***** Outputs *****
%   vertices - the vertices of the intersection
function vertices = intersect_andrew(polygon, b0, b1)
b0 = b0(:);
b1 = b1(:);
n = size(polygon, 2);
if (n == 0)
vertices = [];
return;
end
% Determine which segments the halfplane intersects
intersections = zeros(n, 1);
newPoints = zeros(2, 1, n);
for i = 1:n
a0 = polygon(:, i);
if (i == n)
a1 = polygon(:, 1);
else
a1 = polygon(:, i+1);
end
[bool, pt] = isIntersect(a0, a1, b0, b1);
if (bool)
intersections(i) = 1;
newPoints(:,:,i) = pt;
end
end
% Now figure out which half of the polygon to cut.
if (max(intersections) == 1)    % if the halfplane cuts thru the polygon
intersectIndex = [0, 0]';
ctr = 1;
for i = 1:n
if (intersections(i) == 1)
intersectIndex(ctr) = i;
ctr = ctr + 1;
```

```
end
end
% Check positive orientation of the intersected points
p1 = newPoints(:,:,intersectIndex(1));
p2 = newPoints(:,:,intersectIndex(2));
vec = p2 - p1;
if ((vec' * (b1 - b0)) < 0) % in the wrong direction
p1 = p2;
p2 = newPoints(:,:,intersectIndex(1));
temp = intersectIndex(1);
intersectIndex(1) = intersectIndex(2);
intersectIndex(2) = temp;
end
% Begin definition of new polygon starting with the cutting segment
vertices = [p1, p2];
polygon = circshift(polygon, [0, -intersectIndex(2)]);
if (intersectIndex(2) > intersectIndex(1))
cutSize = intersectIndex(2) - intersectIndex(1);
else
cutSize = intersectIndex(2) - intersectIndex(1) + n;
end
vertices(:, end+1:end+n-cutSize) = polygon(:, 1:n-cutSize);
else
% no intersection with polygon, but depending on orientation, the
% halfplane will either include or exclude the polygon.  Check using
% cross-product.
v1 = b1 - b0;
for i = 1:n
p = polygon(:,i) - b0;
% c = cross(v1, p);
c = v1(1)*p(2) - v1(2)*p(1);
if (c < 0)
vertices = [];
return
end
end
vertices = polygon;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% helper function to detect intersection of a segment u0->u1 with a line
% v0->v1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [bool, pt] = isIntersect(u0, u1, v0, v1)
a1 = u1(1) - u0(1);
a2 = u1(2) - u0(2);
```

```
b1 = v1(1) - v0(1);
b2 = v1(2) - v0(2);
c1 = v0(1) - u0(1);
c2 = v0(2) - u0(2);
D = a2*b1 - a1*b2;
if (D == 0)     % parallel lines
bool = 0;
pt = NaN;
return;
end
t = -b2*c1/D + b1*c2/D;
% s = -a2*c1/D + a1*c2/D
% deal with numerical precision
if (t < -eps(10) || t > 1 + eps(10))
bool = 0;
pt = 0;
elseif (0<=t && t<=1)
bool = 1;
pt = u0 + (u1 - u0)*t;
elseif (abs(t) <= eps(10))
bool = 1;
pt = u0;
elseif (abs(1-t) <= eps(10))
bool = 1;
pt = u1;
end
```

A.2.3 Voronoi Graphical Realization

```
%   This function will take in a set of generator points and regions and
%   will draw the associated graph
%  ***** Inputs *****
%   I.  bounds - the bounding positively oriented polygon (Coordinates of
%       region vertices)
%   II. regionData - the regions are a nx3 cell array with
%        the {i,1} element containing generator point information
%        the {i,2} element containing the arc information, and
%        the {i,3} element containing edge segments (from intersections
%        with the boundary)
%   III.orintations - the 1*n matrix which contains n arctan (theta(i)).
%       theta (i) is the orientation angle of i-th interceptors.
%   Author: Andrew Kwok, University of California at San Diego
%   Date: 17 August 2009
function drawRegionsMostafa(bounds, regionData, orientations)
```

```
n = size(regionData, 1);
hold on
% Define agent label position
delta = max(max(bounds(1,:)) - min(bounds(1,:)), ...
max(bounds(2,:)) - min(bounds(2,:))) / 60;
for i = 1:n
pt = regionData{i, 1};
arcs = regionData{i, 2};
edges = regionData{i, 3};
% Draw arcs and edges
for j = 1:size(arcs, 1)
drawArc(arcs(j, 1:2), arcs(j, 3), arcs(j, 4), arcs(j, 5));
end
for j = 1:size(edges, 1)
line(edges(j, [1, 3]), edges(j, [2, 4]), 'Color', 'blue');
end
% Plot the agent position
[Xa,Ya] = plot_DDMRMostafa([pt;orientations(i)],axis(gca));
% DDMR => Differential drive mobile robot
fill(Xa,Ya,'w');
% Label the agent
label = num2str(i);
text(pt(1)+delta, pt(2)+delta, label, 'Color', 'black');
end
% % Plot the bounding polygon
% bounds(:, end+1) = bounds(:,1);
% lblbounds = plot(bounds(1,:), bounds(2,:), 'r', 'LineWidth', 2);
% text(0.05, 1.4, 'HVU', 'Color', 'red');
% axis equal
axis square
xlim([min(bounds(1,:)-0.25),max(bounds(1,:)+0.25)]);
ylim([min(bounds(2,:)-0.25),max(bounds(2,:)+0.25)]);
xlabel('X [km]');
ylabel('Y [km]');
% title('Interceptor deployment with sensing range')
% legend([lblintcpt lblbounds], 'Interceptor', 'harbour boundary');
grid on
%axis off
%hold off
```

A.2.4 Draw an Arrow

```
function [X,Y] = plot_DDMRMostafa(Q,AX);
% ****** inputs ******
```

```
% Q : a 3 by 1 matrix whose first entry is "x", the second one is "y" and
%     the third one is "theta" or orientation angle.
%****** outputs ******
% The vertices of arrow. The center point of the arrow is the interceptor's
% position.
x     = Q(1);
y     = Q(2);
theta = Q(3);
l1 =0.02*max([AX(2)-AX(1),AX(4)-AX(3)]);
X =[x,x+l1*cos(theta-5*pi/6),x-0.5*l1*cos(theta),x+l1*cos(theta+5*pi/6),x];
Y =[y,y+l1*sin(theta-5*pi/6),y-0.5*l1*sin(theta),y+l1*sin(theta+5*pi/6),y];
```

References

[1] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[2] Calin Belta and Vijay Kumar. Abstraction and control for groups of robots. *Robotics, IEEE Transactions on*, 20(5):865–875, 2004.

[3] Ying-Chih Chen and Chih-Yu Wen. Decentralized cooperative toa/aoa target tracking for hierarchical wireless sensor networks. *Sensors*, 12(11):15308–15337, 2012.

[4] Timothy H Chung, Vijay Gupta, Joel W Burdick, and Richard M Murray. On a decentralized active sensing strategy using mobile sensor platforms in a network. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1914–1919. IEEE, 2004.

[5] Jorge Cortés and Francesco Bullo. Coordination and geometric optimization via distributed dynamical systems. *SIAM Journal on Control and Optimization*, 44(5):1543–1574, 2005.

[6] Jorge Cortes, Sonia Martinez, and Francesco Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11(04):691–719, 2005.

[7] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2004. Proceedings. ICRA'02. IEEE International Conference on*, volume 20, pages 243–255. IEEE, 2004.

[8] Qiang Du and Maria Emelianenko. Acceleration schemes for computing centroidal voronoi tessellations. *Numerical linear algebra with applications*, 13(2-3):173–192, 2006.

[9] Qiang Du, Maria Emelianenko, and Lili Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM journal on numerical analysis*, 44(1):102–119, 2006.

[10] Maria Emelianenko, Lili Ju, and Alexander Rand. Nondegeneracy and weak global convergence of the lloyd algorithm in r^d. *SIAM Journal on Numerical Analysis*, 46(3):1423–1441, 2008.

[11] Harley Flanders. Differentiation under the integral sign. *American Mathematical Monthly*, pages 615–627, 1973.

[12] 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*, pages 299–308. Springer, 2002.

[13] Aman Kansal, William Kaiser, Gregory Pottie, Mani Srivastava, and Gaurav Sukhatme. Reconfiguration methods for mobile sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(4):22, 2007.

[14] Sung G Lee and Magnus Egerstedt. Controlled coverage using time-varying density functions. In *Estimation and Control of Networked Systems*, volume 4, pages 220–226, 2013.

[15] Francois Lekien and Naomi Ehrich Leonard. Nonuniform coverage and cartograms. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5518–5523. IEEE, 2010.

[16] Martin E Liggins, Chee-Yee Chong, Ivan Kadar, Mark G Alford, Vincent Vannicola, Stelios Thomopoulos, et al. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1):95–107, 1997.

[17] Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On centroidal voronoi tessellationenergy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):101, 2009.

[18] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[19] Lili Ma and Naira Hovakimyan. Vision-based cyclic pursuit for cooperative target tracking. *Journal of Guidance, Control, and Dynamics*, 36(2):617–622, 2013.

[20] Sonia MartíNez and Francesco Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

[21] S Miah, B Nguyen, A Bourque, and D Spinello. Nonuniform coverage control with stochastic intermittent communication.

[22] Petter Ogren, Edward Fiorelli, and Naomi Ehrich Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *Automatic Control, IEEE Transactions on*, 49(8):1292–1302, 2004.

[23] Suhas Patankar. *Numerical heat transfer and fluid flow*. CRC Press, 1980.

[24] Luciano CA Pimenta, Guilherme AS Pereira, Mateus M Gonçalves, Nathan Michael, Matthew Turpin, and Vijay Kumar. Decentralized controllers for perimeter surveillance with teams of aerial robots. *Advanced Robotics*, 27(9):697–709, 2013.

[25] Luciano CA Pimenta, Mac Schwager, Quentin Lindsey, Vijay Kumar, Daniela Rus, Renato C Mesquita, and Guilherme AS Pereira. Simultaneous coverage and tracking (scat) of moving targets with robot networks. In *Algorithmic Foundation of Robotics VIII*, pages 85–99. Springer, 2009.

[26] Maurizio Porfiri, D Gray Roberson, and Daniel J Stilwell. Tracking and formation control of multiple autonomous agents: A two-level consensus approach. *Automatica*, 43(8):1318–1328, 2007.

[27] Slobodan N Simić and Shankar Sastry. Distributed environmental monitoring using random sensor networks. In *Information Processing in Sensor Networks*, pages 582–592. Springer, 2003.

[28] Sara Susca, Francesco Bullo, and Sonia Martínez. Monitoring environmental boundaries with a robotic sensor network. *Control Systems Technology, IEEE Transactions on*, 16(2):288–296, 2008.

[29] Zongyao Wang and Dongbing Gu. Cooperative target tracking control of multiple robots. *Industrial Electronics, IEEE Transactions on*, 59(8):3232–3240, 2012.

[30] Peng Yang, Randy A Freeman, and Kevin M Lynch. Multi-agent coordination by decentralized estimation and control. *Automatic Control, IEEE Transactions on*, 53(11):2480–2496, 2008.

[31] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[32] Chun Zhang and Shumin Fei. Energy efficient target tracking algorithm using cooperative sensors. *Systems Engineering and Electronics, Journal of*, 23(5):640–648, 2012.

[33] Guoxian Zhang, Gregory K Fricke, and Devendra P Garg. Spill detection and perimeter surveillance via distributed swarming agents. *Mechatronics, IEEE/ASME Transactions on*, 18(1):121–129, 2013.