Nonuniform Coverage With Time-Varying Risk Density Function

by

Arian Yazdan Panah

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

Mechanical Engineering
Faculty of Engineering
University of Ottawa

Abstract

Multi-agent systems are extensively used in several civilian and military applications, such as surveillance, space exploration, cooperative classification, and search and rescue, to name a few. An important class of applications involves the optimal spatial distribution of a group of mobile robots on a given area, where the optimality refers to the assignment of subregions to the robots, in such a way that a suitable coverage metric is maximized. Typically the coverage metric encodes a risk distribution 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 assign spatial distributions of risk in the region, as for example typically happens in surveillance applications in which high value units have to be protected against external threats coming into a given area surrounding them.

The solution of the optimal control problem in which the metric is autonomous (a function of time only through the state of the robots) is well known in the literature, with the optimal location of the robots given by the centroids of the Voronoi regions forming a Voronoi tessellation of the area. In other words, when the set of mobile robots configure themselves as the centroids of the Voronoi tessellation dictated by the risk density, the coverage itself is maximized.

In this work we advance on this result by considering a generalized area control problem in which the coverage metric is non-autonomous, that coverage metric is time varying independently of the states of the robots. This generalization is motivated by the study of coverage control problems in which the coordinated motion of a set of mobile robots accounts for the kinematics of objects penetrating from the outside. Asymptotic convergence and optimality of the non-autonmous system are studied by means of Barbalat's Lemma, and connections with the kinematics of the moving intruders is established. Several numerical simulation results are used to illustrate theoretical predictions.

Acknowledgements

With my sincerest gratitude, I would like to thank my supervisor, Dr. Davide Spinello, for giving me the opportunity to study my Masters degree at University of Ottawa, for his guidance and insight throughout this project. Also, my co-supervisor, Dr. Suruz Miah, for his advices and honest comments.

Furthermore, I would like to thank my friends who helped me and incented me to strive towards my goal.

A special thank and appreciation to my parents, brother and sisters who have supported me and encouraged me throughout my entire life.

Table of Contents

List of Figures

Chapter 1

Introduction

In recent years there has been increasing emphasis on Wireless Sensor Networks. A wireless sensor network typically consists of a number of sensor nodes working together to monitor a region to obtain data about the environment. These sensors can sense, measure, and gather information from the environment and they can transmit the sensed data to the other sensors or the user. Also they have wireless ability which they use to communicate with one another to form a network.

Wireless sensor networks have great potential for many applications such as space exploration, search and rescue, cooperative classification, health monitoring, target tracking and military surveillance. In military scenario, a wireless sensor network can detect the intrusion and also it can assist in identification. Some examples including motion coordination, optimal control and robots[1] configuration and distribution could be studied in the field of cooperative robotics. The goals of research in wireless sensor networks are to execute the mentioned scenarios by creating new applications, introducing new designs or improving existing protocols, and developing new algorithms.

Voronoi diagram, which will be defined in section 2.4, is a well known solution for optimal control problem. In other words, the coverage metric will be maximized when the interceptors obey the proposed feedback law and converge to their corresponding centroid. These centroids are generated by the Voronoi cells which form a Centroidal Vorornoi Tessellation (CVT) of the domain.

1.1 Motivation

Widespread interest in multi-agent systems has been fueled by the breadth of natural and technological systems that exhibit coordinated activity of a group of autonomous agents. In nature, the flocking of birds, schooling of fish, and foraging of insects provide insight into the emergence of group behaviors from the coordinated actions of individuals. Applications

[1]In this text, the objects that are distributed in our domain and are supposed to track the target and converge to the optimum coverage configuration, are called 'agents', 'interceptors', 'robots' or 'vehicles'.

have been identified in practically every environment accessible to vehicles, from mobile sensing in oceans, to navigation through unknown terrestrial environments, to surveillance by autonomous aircraft, to control of satellite formations in space. In [6], a structure for the design of some behaviors for networks of mobile robots was expressed. One of the possible locational optimization problems in cooperative robotics is how to use some configuration methods in the mobile sensor networks in order to optimize performance. In [7], they considered the problem of controlling motion in a distributed manner for a mobile sensor network for a specific form of motion capability. Currently there is a large interest in the design of stable control laws for optimizing the coverage metric also estimating the boundary condition and tracking the target by using robotic networks. Moreover some works have been done on the convergence of two coordination algorithms for double-integrator dynamics under fixed undirected/directed interaction in a sampled-data setting. [8][9][10][11].

This thesis is motivated by some research in the field of optimizing the non-autonomous coverage using time-varying risk density function. The proposed time-varying risk which enter the domain is caused by some mobile targets and the formation of the agents inside the domain are updated based on the targets motion.

1.2 Thesis Objectives and Contributions

The main goal of this research is to address a class of motion control problems that involve the optimal placement of a set of coordinated agents whose task is to cover a given area. Optimality is defined with respect to a coverage metric that encodes the sensing performance of the agents in the environment, and a risk function that quantifies the relative importance and vulnerability of different regions in the area. The vulnerability is affected by external threads eventually penetrating into the area to be defended, which makes the coverage metric non-autonomous.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 is devoted to an overview of literature on networked multi-robot systems, environment coverage and optimal configuration, Voronoi diagrams, locational optimization and reviewing some conventional coverage optimization techniques, and coverage control. In chapter 3, we introduce a centroidal Voronoi tessellation technique by means of a group of mobile autonomous agents to optimize a non-autonomous coverage metric with time-varying risk density function. Moreover, the asymptotic stability of our system with a suitable feedback law is discussed by using Barbalat's lemma. At the end of chapter 3, the setup and numerical simulation results of several scenarios are represented. Finally the summary, conclusions and some pros/cons of our work are discussed in chapter 4. The MATLAB code that was implemented for simulation, is presented in Appendix.

Chapter 2

Literature Review and Theoretical Background

2.1 Introduction

This chapter provides an overview of the literature on robotic networks. Various operations and features of networked multi robot systems are discussed. The review also covers some examples and applications of environment coverage and location optimizations by a group of robots. Moreover, some techniques used to optimize the coverage control in both time-variant and time-invariant cases are reviewed.

The literature review is organized as follows. Section (2.2) and (2.3) provide details on the development of networked multi robot systems and environment coverage respectively. Voronoi partitioning and centroidal Voronoi tessellation which are commonly used for locational optimization problems, are explained in section (2.4). Section (2.5) includes the locational optimization problem with some details. Some reviews and theoretical background of coverage control and optimization are presented in section (2.6).

2.2 Networked Multi-Robot Systems

A multi-robot system can be defined as a robot group in which the robots can communicate with others and can cooperatively implement different kinds of tasks. Multi-robot systems have the potential in numerous applications, such as military surveillance, space exploration, cooperative classification, rescue and search, etc. Three important problems in the field of cooperative robotics are: (i) environment coverage and locational optimization, (ii) task assignment, and (iii) target tracking. These three problems are the basic subtasks that must be solved in the global problem of tracking multiple targets in a pre-specified region. When the number of targets and their locations are unknown, the need for environment coverage is important to maximize the probability of detecting them. Task assignment is necessary to define when a given agent should perform a coverage behavior and when it

should do target tracking. For multiple targets it is also necessary to allocate targets to tracking agents. After assigning a target-tracking task to an agent, it is desirable to use control laws with guaranteed convergence.

In the multi-agent network, some sensors could work as agents. The paper [12] describes the concept of sensor networks which has been made viable by the convergence of micro-electro-mechanical systems technology, wireless communications and digital electronics. In [12], The sensing tasks, potential sensor networks applications, and the communication architecture for sensor networks are outlined, and then algorithms and protocols for each layer in the literature are explored. A more detailed review and an overview of several new applications are given in [13]. In this paper, the problems are categorized in three different fields: i) internal platform and underlying operating system, ii) communication protocol stack, and iii) network services, provisioning, and deployment. One of the interesting fields in the multi-robot networks is, distributed coordination of multiple vehicles which includes unmanned aerial vehicles, unmanned ground vehicles, and unmanned under-water vehicles. The distributed coordination of multiple vehicles has been a very active research subject studied extensively by the systems and control community. Results in this area are categorized into several categories, such as consensus, formation control, optimization, and estimation. An additional survey on main results and progress in distributed multi-agent coordination was done in [14].

2.2.1 Environment Monitoring

Environmental monitoring is one of the main applications of the emerging technology of wireless sensor networks. Preventing forest fires, energy conservation, oceanography, building science are some applications of environment coverage. A large number of sensor nodes can be used to automatically detect the emergence of critical points (such as heat sources in the context of forest fires) and notifying the base station which can then take further action. Various algorithms such as Lyod's algorithm has been used in order to optimize the coverage of an environment which will be explained later [15][16][17][18][19][20].

2.2.2 Task Assignment

One of the most common applications in mobile sensor and multi-robot networks is task assignment. In the work [21] studied motion coordination problems for groups of robots that exchange information through a rate-constrained communication network. For rendezvous and deployment problems, the authors propose an integrated control and communication scheme combining a logarithmic coder/decoder with linear coordination algorithms. They show that the closed-loop performance is comparable to the one achievable in the quantization-free model in which the time complexity is unchanged and the exponential convergence factor degrades smoothly as the quantization accuracy becomes coarser [21].

2.2.3 Target Tracking

In recent years there has been increasing emphasis on tracking, detection, situation assessment and classification of target by using multiple robots. For example, the main goal of modern surveillance systems is to track a target and estimate its location. The object of target tracking is to track the location of a target with accuracy determined by sensor devices. Typically, targets move through the sensor field, and the sensor field is assumed to be dense enough so that the entire area of interest is covered by sensors. The main components of tracking are estimation and data association, distributed fusion for tracking will involve distributed estimation and distributed association. For distributed estimation, most research has assumed a hierarchical fusion architecture. Distributed nonlinear results generally follow the same philosophy. One of the most common approaches that could be used in order to derive the linear and nonlinear fusion equations for hierarchical architectures is, the information graph approach for modeling the information flow in arbitrary distributed fusion architectures and developing fusion equations.

Modern surveillance systems often utilize multiple physically distributed sensors of different types to provide complementary and overlapping coverage on targets. In order to generate target tracks and estimates, the sensor data need to be fused. While a centralized processing approach is theoretically optimal, there are significant advantages in distributing the fusion operations over multiple processing nodes. The studies in [22] discuss architectures for distributed fusion, whereby each node processes the data from its own set of sensors and communicates with other nodes to improve on the estimates.

Typical advantages of using sensor networks include relatively lower costs, inherent robustness and greater coverage area. These advantages are enhanced by using mobile sensor networks. Moreover greater accuracy is possible since not only more observations being taken for the same target but also since multiple types of sensors can be used to obtain different types of measurements about the target. All the problems related to this field become more complicated when multiple targets are present. As the number of nodes in the network grows, it also becomes desirable for purposes of robustness and communication complexity that the solutions to these problems should not involve a central computation node. In [23], the problem of optimal positioning of sensors are specifically concerned. This problem arises when there are mobile sensors and the quality of observations of some target(s) varies with the placement of the sensors. This can be due to the fact, e.g., that the observation noise varies with the distance between a target and a sensor. Thus one of the questions is that where to place sensors to obtain the best estimations. Moreover the target might itself be mobile and thus the optimal sensor positions will change with time [24].

Cooperative control of multiple autonomous agents has become an important topic of robotics and control theory research. The main theme is to analyze and synthesize spatially distributed control architectures to coordinate groups of autonomous agents. A typical assumption is that each agent communicates relevant information to its neighbors. Cooperative target tracking is one form of motion coordination where a group of agents reach desired relative positions and orientations with respect to the target [25] [26] [27].

There are five major categories for the target tracking solutions: tree-based tracking,

cluster-based tracking, prediction-based tracking, mobicast message-based tracking, and hybrid tracking. Studies have shown that the cluster-based tracking algorithms have better network scalability and resource utilization compared with those in other categories. Prediction-based tracking rely on tree-based and cluster-based tracking in addition to the prediction method, but the tracking accuracy cannot be guaranteed. Mobicast message-based tracking method depends on prediction, which is a multi-cast method in which message is delivered to a group of nodes that change with time according to estimated velocity of moving entity. Scheduling strategies vary in target tracking protocols and time synchronization may be needed to set the wake up and sleep timings of sensor nodes. Since the proposed approaches fall into the category of cluster-based tracking, the research results of this category were focused in [28]. The paper [28] proposes a distributed method for cooperative target tracking in hierarchical wireless sensor networks and evaluates the merits and trade-offs of the protocol design towards developing more efficient and practical algorithms for object position estimation. The problem of moving target tracking with a group of mobile robots was focused in [29]. Each robot in the group has a pan/tilt camera to detect the target and has limited communication capability to communicate with neighbor robots [29].

Energy is one of the possible issues that should be considered when we study the target tracking in wireless sensor networks (WSN), because in these networks it is assumed that each sensor has a limited range for detecting the presence of the object, and the network is sufficiently dense so that the sensors can cover the area of interest. Due to the limited battery resources of sensors, there is a tradeoff between the energy consumption and tracking accuracy. An energy efficient tracking algorithm was proposed in [30]. Based on the cooperation of dispatchers, sensors in the area are scheduled to switch their working mode to track the target. Since the energy consumed in active mode is higher than that in monitoring or sleeping mode, for each sampling interval, a minimum set of sensors is woken up based on the selected mechanism. Meanwhile, other sensors stay in sleeping mode. Performance analysis and simulation results show that the proposed algorithm provides a better performance than other existing approaches [31] [32] [33].

Target tracking plays a key role for vehicular ad hoc networks (VANETs[1]) due to the fact that a wide variety of envisioned applications rely on the ability of this technique of detecting, localizing, and tracking objects surrounding a vehicle. This subject has been studied in some fields such as airborne traffic, computer vision, and wireless sensor networks. A VANET brings out new challenges that should be addressed [34] [35] [36] [37]. For instance, the cluttered and dense scenarios, communication issues such as short term links, and the variety of objects considered to be targets, are some of the new concerns to be taken into account. Applications such as collision warning/avoidance systems require strict time constrains, while others impose only mild restrictions. This complex and het-

[1]A vehicular ad hoc network (VANET) uses cars as mobile nodes in a mobile ad hoc network (MANET) to create a mobile network. A VANET turns every participating car into a wireless router or node, allowing cars approximately 100 to 300 meters of each other to connect and, in turn, create a network with a wide range. As cars fall out of the signal range and drop out of the network, other cars can join in, connecting vehicles to one another so that a mobile Internet is created. It is estimated that the first systems that will integrate this technology are police and fire vehicles to communicate with each other for safety purposes.

erogeneous environment was discussed in [38], where they didactically divided the main problems into four components: the target's motion model, measurement models, data association problem, and filtering. Also the communication issues and how they affect these systems were discussed.

2.3 Environment Coverage and Optimal Configurations

This section reviews some works about optimum configuration or location of some agents or sensors in order to have an optimized coverage of an environment. The locational optimization problem, also known as facility location or k center problem, is a branch of operations research and computational geometry concerned with the optimal placement of facilities.The study of locational optimization has a long history. It dates back to 1909 when Weber studied the locational optimization of a firm in a region, called the Weberian problem. Since then, various kinds of locational optimization problems have been studied in different fields and have been reviewed from many viewpoints. Locational optimization problems pervade a broad spectrum of scientific disciplines. Biologists rely on locational optimization tools to study how animals share territory and to characterize the behavior of animal groups obeying the following interaction rule: each animal establishes a region of dominance and moves toward its center. Locational optimization problems are spatial resource-allocation problems (e.g., where to place mailboxes in a city or cache servers on the Internet) and play a central role in quantization and information theory (e.g., how to design a minimum-distortion fixed-rate vector quantizer). Other technologies affected by locational optimization include mesh and grid optimization methods, clustering analysis, data compression, and statistical pattern recognition.

One of the interesting topics with various applications is the p-center location problem. this problem is about covering a given area in the plane with p identical circles which have the smallest possible radius. the sensors(or agents) are located at the centers of these circles. The location of emergency facilities such as fire stations or hospitals is frequently modeled by the p-center problem. Each customer patronizes the closest facility and the objective is to minimize the distance for the farthest customer. Other examples include covering an area with p television transmitters, warning sirens, or sprinkler systems. Each of these facilities covers a circular area and p locations need to be found such that the maximal distance to a facility is minimized. Another interesting application is a radar coverage problem. An area in the sky contains objects to be detected by radar. The radar beam covers a circular area of a given radius and the desired area needs to be covered by the minimum number of identical circles. In many applications, the radius of the covering circles is given and the problem is to find the minimum number of identical circles with a given radius that cover the area. This requires the solution of various p-center problems and the smallest p that covers the area within the pre-specified radius is selected. Furthermore, covering an area such as a square with p identical circles with minimal radius is equivalent to covering the maximal area of the same shape with identical circles of a given radius. This is the appropriate objective for a problem if the maximal area of a given shape such as

a square or a circle needs to be covered by p circles of a given radius. It is usually assumed that demand for the required service originates from a finite set of demand points. In many cases, assuming that an area represents the set of demand points is more realistic. Demand originating in an area rather than in a finite set of demand points, applies to location of mobile demand. For example, the location of stations for cellular telephones requires a continuous approach. A continuous representation means that the area where demand exists is defined. Suzuki and Drezner considered the unweighted version of the problem in a square area [39]. Thus, as long as demand exists at a point, its magnitude is irrelevant to the problem. Therefore it suffices to have the collection of points where demand exists. The heuristic solution approach described in this paper addresses any area on the plane, although areas with a convex polygon shape are easiest to handle [40].

Technological advances in communication systems and the growing ease in making small, low-power and inexpensive mobile systems now make it feasible to deploy a group of networked vehicles in a number of environments. Furthermore, network solutions offer potential advantages in performance, robustness, and versatility for sensor-driven tasks such as search, survey, exploration, and mapping. A cooperative mobile sensor network is expected to outperform a single large vehicle with multiple sensors or a collection of independent vehicles when the objective is to climb the gradient of an environmental field. The single, heavily equipped vehicle may require considerable power to operate its sensor payload, it lacks robustness to vehicle failure and it cannot adapt the configuration or resolution of the sensor array. An independent vehicle with a single sensor may need to perform costly maneuvers to effectively climb a gradient, for instance, wandering significantly to collect rich enough data much like the run and tumble behavior of flagellated bacteria [41]. They focus on gradient climbing missions in which the mobile sensor network seeks out local maxima or minima in the environmental field. The network can adapt its configuration in response to the sensed environment in order to optimize its gradient climb [42].

The problem of deploying a mobile sensor network in an unknown environment was considered in [43]. A mobile sensor network is composed of a distributed collection of nodes, each of which has sensing, computation, communication and locomotion capabilities. Such networks are capable of self-deployment; i.e., starting from some compact initial configuration, the nodes in the network can spread out such that the area covered by the network is maximized. In this work, they present a potential-field-based approach to deployment. The fields are constructed such that each node is repelled by both obstacles and by other nodes, thereby forcing the network to spread itself throughout the environment. The approach is both distributed and scalable [44].

The current technological development of relatively inexpensive communication, computation, and sensing devices has lead to an intense research activity devoted to the distributed control and coordination of networked systems. nowadays in robotic settings, the study of large groups of autonomous vehicles is a timely concern [45]. The potential advantages of networked robotic systems are their versatility and robustness in the realization of multiple tasks such as manipulation in hazardous environments, pollution detection, estimation and map-building of partially known or unknown environments. The coordination algorithm for groups of mobile agents performing deployment and coverage tasks was pre-

sented in [46]. As an important modeling constraint, they assume that each mobile agent has a limited sensing/communication radius. Based on the geometry of Voronoi[2] partitions and proximity graphs, a class of aggregate objective functions and propose coverage algorithms were analyzed in continuous and discrete time.

2.3.1 Cooperative Robotics

There has been a great deal of interest in cooperative robotics in the last few years, started mainly by the technological advances in control techniques for vehicles and the progress in computation and communication capabilities. Some subjects such as, the communication methods between robots(or sensors) such as data sharing, estimation the event of interest, motion coordination, control and robots configuration and distribution could be discussed in the category of cooperative robotics. The research in the field of control and coordination for multiple robots is currently progressing in areas like automated highway systems, formation flight control, unmanned under- water vehicles, satellite clustering, exploration, surveillance, search and rescue, mapping of unknown or partially known environments, distributed manipulation, and transportation of large objects.

One of the possible locational optimization problems in cooperative robotics is how to use some configuration methods in the mobile sensor networks in order to optimize performance. In [7], they considered the problem of controlling motion in a distributed manner for a mobile sensor network for a specific form of motion capability. Mobility itself may have a high resource overhead, hence they exploited motility, a constrained form of mobility, which has very low overheads but provides significant reconfiguration potential. An architecture was provided that allows each node in the network to learn the medium and phenomenon characteristics. they described a quantitative metric for sensing performance that is concretely tied to real sensor and medium characteristics, rather than assuming an abstract range based model. The problem of determining the desirable network configuration was expressed as an optimization of this metric. They presented a distributed optimization algorithm which computes a desirable network configuration, and adapts it to environmental changes. The relationship of the proposed algorithm to simulated annealing and incremental subgradient descent based methods was discussed. A key property of their algorithm is that convergence to a desirable configuration can be proved even though no global coordination is involved [47].

Widespread interest in multi-agent systems has been fueled by the breadth of natural and technological systems that exhibit coordinated activity of a group of autonomous agents. In nature, the flocking of birds, schooling of fish, and foraging of insects provide insight into the emergence of group behaviors from the coordinated actions of individuals. Applications have been identified in practically every environment accessible to vehicles, from mobile sensing in oceans, to navigation through unknown terrestrial environments, to surveillance by autonomous aircraft, to control of satellite formations in space. Simultaneous tracking and formation control is addressed for a team of autonomous agents that evolve dynamically in a space containing a measurable vector field. Each agent measures

[2]Voronoi partitioning will be explained in the section (2.4).

the local value of the field along its trajectory and occasionally shares relevant information with other agents, in order to estimate the spatial average obtained from averaging measurements across all agents. Using shared information, agents control their trajectories in a cooperative manner, with the dual goals of driving the average field measurement to a specified value and maintaining a desired formation about the average. Two approaches to virtual leader estimation could be considered. The first involves the synthesis of a common virtual leader state, whereas the second involves decentralized estimation of the virtual leader by individual agents. Under the second approach, control is posed as a two-level consensus problem, where agents reach agreement on the virtual leader state at one level and reach formation about the virtual leader at the other level. The decentralized approach is effective even when communication among agents is limited, in the sense that the associated network graph can be disconnected in frozen time [48] [49].

In [6], a framework for the design of collective behaviors for groups of identical mobile agents was described. The approach is based on decentralized simultaneous estimation and control, where each agent communicates with neighbors and estimates the global performance properties of the swarm needed to make a local control decision. Challenges of the approach include designing a control law with desired convergence properties, assuming each agent has perfect global knowledge; designing an estimator that allows each agent to make correct estimates of the global properties needed to implement the controller; and possibly modifying the controller to recover desired convergence properties when using the estimates of global performance. In [6], this framework was applied to the problem of controlling the moment statistics describing the location and shape of a swarm. They derived conditions which guarantee that the formation statistics are driven to desired values, even in the presence of a changing network topology.

Currently there is a large interest in the design of stable and decentralized control laws for distributed motion coordination. In [8], they studied deployment of a robotic network where each agent is equipped with limited-range communication and sensing capabilities. The footprint of each sensor is a wedge-shaped region centered about each robots orientation with an angular width less than or equal to π radians. Laventall and Corets in [8] considered the deployment of a network of robotic agents with limited-range communication and anisotropic sensing capabilities. They encode the environment coverage provided by the network by means of an expected-value objective function. A constant-factor approximation of this measure via an alternative aggregate objective function whose gradient is spatially distributed over the limited-range Delaunay proximity graph was provided. They characterize the smoothness properties of the aggregate expected-value function and propose a distributed deployment algorithm that enables the network to optimize it.

Much recent attention has been given to the problem of boundary estimation and tracking by means of robotic networks. The common goal is to design a distributed algorithm that allows a limited number of mobile agents to detect the boundary of a region of interest and estimate it as it evolves. An algorithm to monitor an environmental boundary with mobile agents was proposed and analyzed in [9] where the objective is to optimally approximate the boundary with a polygon. The mobile sensors rely only on sensed local information to position some interpolation points and define an approximating polygon. They designed an algorithm that distributes the vertices of the approximating polygon uni-

formly along the boundary. The notion of uniform placement relies on a metric inspired by approximation theory for convex bodies. The algorithm is provably convergent for static boundaries and efficient for slowly-moving boundaries because of certain input-to-state stability properties.

Consensus plays an important role in achieving distributed multi-vehicle coordination. The basic idea of consensus is that a team of vehicles achieves an agreement on a common value by negotiating with their neighbors. in [10], they worked on the convergence of two coordination algorithms for double-integrator dynamics under fixed undirected/directed interaction in a sampled-data setting. The first algorithm guarantees that a team of vehicles achieves coordination on their positions with a zero final velocity while the second algorithm guarantees that a team of vehicles achieves coordination on their positions with a constant final velocity. They showed necessary and sufficient conditions on the sampling period, the control gain, and the communication graph such that coordination is achieved using these two algorithms under, respectively, an undirected interaction topology and a directed interaction topology. Tools like matrix theory, bilinear transformation, and Cauchy theorem were used for convergence analysis [50].

A different approach to coordination of multiple mobile robots is presented in [11]. The approach relies on the notion of constraint forces which are used in the development of the dynamics of a system of constrained particles with inertia. A familiar class of dynamic, nonholonomic robots are considered. The goal is to design a distributed coordination control algorithm for each robot in the group to achieve, and maintain, a particular formation while ensuring navigation of the group. The theory of constraint forces is used to generate a stable control algorithm for each mobile robot that will achieve, and maintain, a given formation. The advantage of the proposed method is that the formation keeping forces (constraint forces) cancel only those applied forces which act against the constraints. Another feature of the proposed distributed control algorithm is that it allows to add/remove other mobile robots into/from the formation gracefully with simple modifications of the control input. Further, the algorithm is scalable [51].

2.3.2 Perimeter Surveillance

Aerial robots are particularly adapt to the application of surveillance due to their maneuverability and extended workspace in comparison to ground robots. However, there are several challenges in developing a viable surveillance approach. In particular, it must be ensured that the robots maintain some continuous coverage of the environment. Further, parallel to pursuing methods that leverage large numbers of robots to cover large spaces, the planning and control problems must guarantee convergence, regardless of the system size. An approach was studied in [52] that relies on decentralized individual robot control laws that drives a team of aerial robots to converge and circulate along a curve in a three-dimensional environment. This curve may be considered as a perimeter to be surveilled by the robots. The solution presented in the paper [52] is based on an artificial vector field modulated by a collision avoidance scheme and relies only on local sensing.

The problem of perimeter detection has variety of applications. A hybrid system of

finite states is proposed in [53] for multiple autonomous robotic agents with the purpose of hazardous spill perimeter detection and tracking. In the system, each robotic agent is assumed to be in one of three states: searching, pursuing, and tracking. The agents are prioritized based on their states, and a potential field is constructed for agents in each state. For an agent in the tracking state, the agents location and velocity as well as those of its closest leading and trailing agents are utilized to control its movement. The convergence of the tracking algorithm is analyzed for multiple spills under certain conditions. Simulation and experiment results of their work show that with the proposed method, the agents can successfully detect and track the spills of various shapes, sizes, and movements.

2.3.3 Planning

Planning has been an attractive area of research in artificial intelligence (AI) for over three decades. Planning techniques have been applied in a variety of domains including robotics, process planning, web-based information gathering, spacecraft mission control, autonomous agents, military operations, etc. Military planning is one of the functions of Command and Control (C2) and is a subject attracting much attention in military world affairs. Military planning includes both decision making and anticipatory or contingency decision making. Usually time is a very important component for planning military operations, especially for those carried out in the theater of operations. From a planning perspective, time can be treated as a resource to model the duration of actions. Usually, explicit consideration of time in planning can significantly extend the expressiveness of a plan. Indeed, the durations of actions, the dynamic nature of the environment (event expected to occur in the future), and the overlapping of actions that have interacting and joint effects are main reasons (among a set of other ones) for making time explicit in planning. Coordination of actions and plans that must be achieved by multiple agents is one of the most difficult tasks in the multi-agent domain. In order to work together and achieve a common goal, agents need to coordinate their plans in a way that guarantees, if possible, the success of each individual agent plan. A temporal fusion mechanism that allows a set of agents to fuse their plans and generate a global coordinated plan was proposed in [54]. First, they defined a temporal plan as a set of temporally constrained actions. The fusion of several temporal plans is a temporal plan, which can be executed by several agents. The proposed framework could be applied to a Combat Search and Rescue application.

2.3.4 Protection Of High Value Units

Requirements expected from Unmanned Surface Vehicles(USV) used by state authorities to minimize threats to security in harbour areas show how significant harbour infrastructure and security of the inshore navigation are for the national security in its most comprehensive context. It should also be underlined that harbour basins and inshore waters are specific enough to adopt a special way of USVs operation and consequently the architecture of their control system. Harbour channels, terminals and narrow passages are water areas which are characterized not only by specific hydrologic and bathymetric conditions, but

also by large intensity of navigation streams. The determinants mentioned above have an impact on the surface vessels designed for their protection and the performance of specific task and therefore their control systems should have specific architecture depending on executed tasks. Kitowski in [55] presents basic issues connected with the development of the optimal control of an Unmanned Surface Vehicle (USV) when used in the process of minimizing threats to security within the harbour areas. Specific nature of USV operation when carrying out protection tasks requires a different view on the system structure from the hardware perspective and on the processes of decision making other than that when carrying out other tasks. Correct determination of both factors has a decisive influence on the effectiveness of the mission carried out by the vehicle in this case [56] [57] [58].

2.3.5 Animal Behavior

Coverage problems have a compelling analogy in possible models of social foraging by animal groups. Backed by observations of animal behavior across a number of species, biologists model distribution of animals over patchy resource environments according to a measure of patch suitability that depends on factors such as resource richness or conditions for survival. Suitability varies with animal density; a typical assumption is that suitability decreases with increasing animal population. For example, suitability declines when more animals converge on a given patch since resources (e.g., prey) may be limited, and thus average consumption rates go down with more hungry consumers. Since animals prefer patches with higher suitability, nonuniformity in the distribution of resources (i.e., the suitability) reflects nonuniformity in the distribution of the animal group.

The distributed estimation algorithms that were presented in [59], allow robots in a communication network to maintain estimates of summary statistics describing the shape of the swarm. They show that these estimators, combined with motion controllers implemented on each robot, result in the swarm formation statistics being driven to desired values in the presence of a changing network topology and the addition and deletion of robots.

2.3.6 Image Processing

The development of image processing techniques has brought many technological advances in communications, entertainment, security, medicine, and manufacturing. Image processing includes the enhancement, restoration, coding, and understanding of images and is aimed not only at providing quality pictorial information and transmission efficiency, but also at assisting in machine reasoning and recognition. Naturally, the enhancement and restoration of an image often depends on a good understanding of the image itself. One of the oldest and yet still popular tools employed to analyze and understand images is provided by clustering. Broadly speaking, clustering is a commonly used technique for the determination and extraction of desired features from large data sets and for the determination of similarities and dissimilarities between elements in the data set. In the context of image processing, data sets take the form of one or more images. The concept of centroidal

Voronoi tessellations (CVT)[3] has recently received much attention in numerous applications, including computer graphics and image processing. Centroidal Voronoi tessellations are special Voronoi tessellations for which the generators of the tessellation are also the centers of mass (or means) of the Voronoi cells or clusters. centroidal Voronoi tessellations have been found to be useful in many disparate and diverse settings. CVT-based algorithms were developed in [60] for image compression, image segmentation, and multichannel image restoration applications. In the image processing context and in its simplest form, the CVT-based methodology reduces to the well-known k-means clustering technique. However, by viewing the latter within the centroidal Voronoi tessellations context, very useful generalizations and improvements can be easily made. Several such generalizations were exploited in the paper [60] including the incorporation of cluster dependent weights, the incorporation of averaging techniques to treat noisy images, extensions to treat multichannel data, and combinations of the aforementioned.

2.4 Voronoi Diagrams

In mathematics, a Voronoi diagram is a partition of a hypervolume into cells determined by a metric that measure the proximity with respect to points in a specific subset of the space. The set of points, called sites, seeds, or generators, is specified, and for each generator there is a corresponding cell consisting of all points closer to that generator than to any other generator. the algorithm is named after Georgy Voronoy, and it is alternatively known as Voronoi decomposition, Voronoi tessellation, Dirichlet tessellation (it is named after Peter Gustav Lejeune Dirichlet), or a Voronoi partition. Voronoi diagrams have theoretical and practical applications to a vast number of fields, especially in technology and science but also in visual arts.

2.4.1 Formal Definition

Let Ω be a nonempty set equipped with a distance function d. Let $(\mathbf{p}_i)_{i\in I}$ be an ordered collection (tuple) of generators (nonempty subsets) in the space Ω and I be a set of indices. The Voronoi region, or Voronoi cell, $\mathcal{V}_i$, associated with the site (generator) $\mathbf{p}_i$ is the set of all points in Ω whose distance to $\mathbf{p}_i$ is less than or equal to their distance from the other sites $\mathbf{p}_j$, where j is any index different from i. In other words, if $d(x, E) = \inf \{d(x, e)|e \in E\}$ describes the distance between the point x and the subset E, then the Voronoi diagram is simply the collection of cells $(\mathcal{V}_i)_{i\in I}$. The generators are assumed to be separate and limited (see Figure 2.1) [61].

$$\mathcal{V}_i = \{x \in \Omega | d(x, \mathbf{p}_i) \leq d(x, \mathbf{p}_j), \forall i \neq j\}. \tag{2.1}$$

[3]CVT will be explained in the section (2.4.4).

In the case in which the space is a finite-dimensional Euclidean space, each generator is a point, there are finitely many different points, then the Voronoi regions are convex polytopes and they can be presented in a way using their sides, 2-dimensional faces, vertices, etc. Each such cell is obtained from the intersection of half-spaces, and hence it is a convex polygon. However, in general the Voronoi regions may not be convex or even connected. The segments and vertices of the Voronoi tessellation are all the points in the plane that have the same distance to the two and three nearest generators respectively. Then, as represented in [62], "all locations in the Voronoi polygon are closer to the generator point of that polygon than any other generator point in the Voronoi diagram in Euclidian plane".

Example

As a simple demonstration, consider a group of sport centers in a flat town. Suppose the goal is to estimate the number of athletes of a given center. By assuming that membership fee, facilities, quality of service. are equal in all centers, it is reasonable to assume that athletes choose their preferred center simply by distance considerations: they will go to the sport center located nearest to them. In this case the Voronoi cell of a given sport center can be used for giving an estimate on the number of potential athletes going to this sport center (which is modeled by a point in our flat town). The Euclidean distance is defined as

$$d[(b_1, b_2), (c_1, c_2)] = \sqrt{(b_1 - c_1)^2 + (b_2 - c_2)^2}. \tag{2.2}$$

However, if we consider the case where athletes only go to the sport centers by a vehicle and the traffic paths are parallel to the and x and y axes, as in Manhattan, then a more realistic distance function will be

$$d[(b_1, b_2), (c_1, c_2)] = |b_1 - c_1| + |b_2 - c_2|. \tag{2.3}$$

According to the definition, Voronoi cells can be defined for metrics other than Euclidean distance, such as the Manhattan distance or Mahalanobis distance but the boundaries of the Voronoi cells may be more complicated than in the Euclidean case.

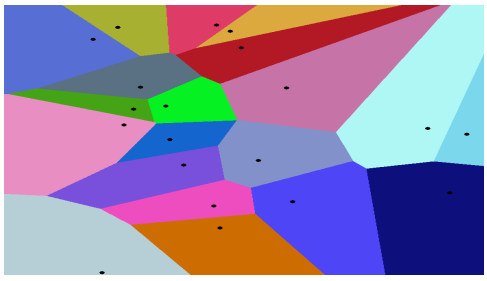


Figure 2.1: Euclidean Voronoi diagram [1].

2.4.2 Some Applications of Voronoi Tesselations

- Epidemiology, Voronoi tessellations can be used to correlate sources of infections in epidemics. A British physician, John Snow, implemented one of the early applications of Voronoi diagrams in 1854 to demonstrate how the majority of people who died in the Soho cholera epidemic lived closer to the infected Broad Street pump than to any other water pump (Figure 2.2) [2].

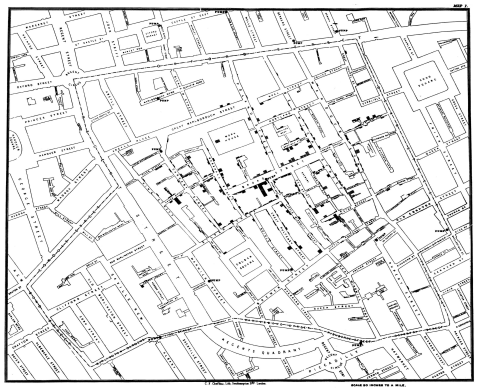


Figure 2.2: John Snow's original map [2].

- Biology, Voronoi tessellations are used to model biological structures, such as cells [63] and bone micro-architecture [64].
- Autonomous robot navigation, Voronoi diagrams are used to find clear routes. If the points are obstacles, then the edges of the graph will be the routes furthest from obstacles (and theoretically any collisions).
- Networking, Voronoi diagrams can be used in derivations of the capacity of a wireless network.
- Astrophysics, Voronoi diagrams are used to generate adaptative smoothing zones on images, adding signal fluxes on each one. The main objective for these procedures is to maintain a relatively constant signal-to-noise ratio on all the image.
- A point location data structure can be built on top of the Voronoi diagram in order to answer nearest neighbor queries, where one wants to find the object that is closest to a given query point. Nearest neighbor queries have numerous applications. For example, one might want to find the nearest hospital, or the most similar object in a database. A large application is vector quantization, commonly used in data compression.

2.4.3 Algorithms for Generating Voronoi Diagrams

A number of algorithms such as, discretization algorithm [65], naive algorithm, incremental algorithm, divide/Conquer algorithm, and fortune algorithm [66] have been proposed for

generating Voronoi partitions. Algorithms to generate Voronoi diagrams can be non-trivial, especially for inputs of dimension higher than two, so the steps of two of algorithms for generating this diagram are represented in this section.

Incremental Algorithm

The Voronoi diagram is computed by incremental insertion [67] i.e., $\mathcal{V}(\mathbf{X})$ is obtained from $\mathcal{V}(\mathbf{X} \setminus \{\mathbf{x_n}\})$ by inserting the site $\mathbf{x}$. As the the cell associated to $\mathbf{x}$ can have up to $(n-1)$ edges, for $n = |\mathbf{X}|$, this leads to a runtime of $O(n^2)$. The technique of inserting Voronoi regions have been improved by Ohya, Iri, Murota [68] and Sugihara [69] , by achieving average time complexity of $O(n)$. The incremental method set up with a simple Voronoi diagram for two or three sites, and modified the diagram by adding sites one by one. The major part of the incremental method is to translate $\mathcal{V}(\mathbf{X} \setminus \{\mathbf{x_n}\})$ to $\mathcal{V}(\mathbf{X})$ for each $\mathbf{x_n}$.

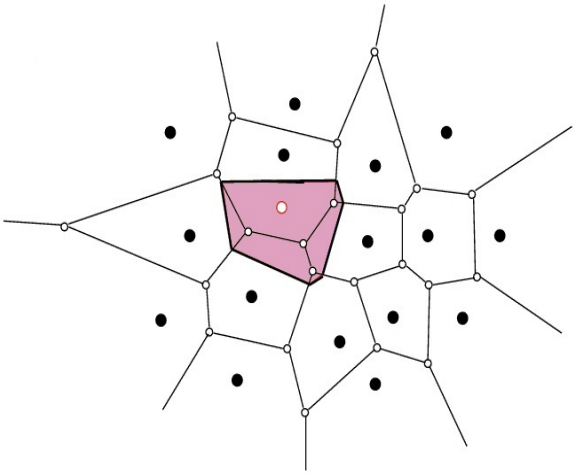


Figure 2.3: Incremental algorithm [3].

The steps of generating Voronoi diagram by using incremental algorithm are as follow: Suppose that we have already built the Voronoi diagram $\mathcal{V}(\mathbf{X} \setminus \{\mathbf{x_n}\})$ as shown by solid lines (Figure 2.3), and would like to add a new site $\mathbf{x_n}$. First, find the site, say $\mathbf{x}_i$, whose Voronoi polygon contains $\mathbf{x_n}$, and draw the perpendicular bisector between $\mathbf{x_n}$ and $\mathbf{x}_i$, denoted by $B(\mathbf{x_n}, \mathbf{x}_i)$. The bisector crosses the boundary of $\mathcal{V}(\mathbf{x}_i)$ at two points, point s_1 and point s_2. Site $\mathbf{x_n}$ is to the left of the directed line segment s_1s_2. The line segment s_1s_2 divides the Voronoi polygon $\mathcal{V}(\mathbf{x}_i)$ into two pieces. The one on the left belonging to the Voronoi polygon of $\mathbf{x_n}$. Thus, we get a Voronoi edge on the boundary of the Voronoi polygon of $\mathbf{x}_i$.

Starting with the edge s_1s_2, expand the boundary of the Voronoi polygon of $\mathbf{x_n}$ by the following procedure. The $B(\mathbf{x_n}, \mathbf{x}_i)$ crosses the boundary of $\mathcal{V}(\mathbf{x}_i)$ at s_2, entering the adjacent Voronoi polygon, say $\mathcal{V}(\mathbf{x}_j)$. Therefore, next draw the $B(\mathbf{x_n}, \mathbf{x}_j)$, and find the point, s_3, at which the bisector crosses the boundary of $\mathcal{V}(\mathbf{x}_j)$. Similarly, find the sequence

of segments of perpendicular bisectors of $\mathbf{x_n}$ and the neighboring sites until we reach the starting point s_1. Let this sequence be $(s_1s_2, s_2s_3, ..., s_{m-1}s_m, s_ms_1)$. This sequence forms a counterclockwise boundary of the Voronoi polygon of the new site $\mathbf{x_n}$. Finally, we delete from $\mathcal{V}(\mathbf{X} \setminus \{\mathbf{x_n}\})$ the substructure inside the new Voronoi polygon, and thus get $\mathcal{V}(\mathbf{X})$.

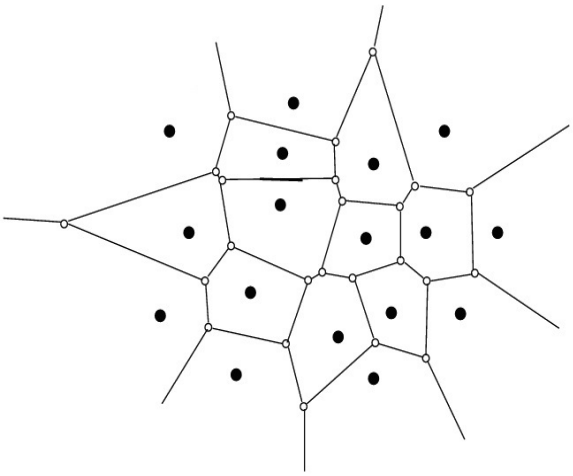


Figure 2.4: A Voronoi diagram generated by Incremental algorithm [3].

Divide and Conquer Algorithm

Shamos and Hoey [70] described the first $O(n \log n)$ deterministic algorithm for computing the Voronoi diagram on the plane. The 'Divide-and-Conquer' is a fundamental paradigm for designing efficient algorithms. In this paradigm, the original problem is recursively divided into several simpler sub-problems of roughly equal size, and the solution of the original problem obtained by merging the solutions of the sub-problems. In the divide-and-conquer approach by Shamos and Hoey, the set of sites, $\mathbf{X}$, is split up by a dividing chain (the blue line in Figure 2.5) into subsets $\mathbf{X}_R$ (the black ones) and $\mathbf{X}_L$ (the red ones) of about same sizes (Figure 2.5). Then, the Voronoi diagram, $\mathcal{V}(\mathbf{X}_R)$, of subset $\mathbf{X}_R$ and Voronoi diagram, $\mathcal{V}(\mathbf{X}_L)$, of subset $\mathbf{X}_L$ are computed recursively. The unwanted or dotted red lines and black lines in Figure 2.5 need to be clipped to the right and left side of the blue dividing chain respectively.

The important part of algorithm consists of finding the split line, and merging $\mathcal{V}(\mathbf{X}_L)$ and $\mathcal{V}(\mathbf{X}_R)$, to obtain $\mathcal{V}(\mathbf{X})$ of original set $\mathbf{X}$. If computing the split line and merging of two Voronoi diagrams could carried out in time $O(n)$ then the overall running time is $O(n \log n)$ from the recurrence relation $T(n) = 2T(n/2) + O(n)$.

Calculation of vertical or horizontal split lines is straightforward during recursion if the sites in $\mathbf{X}$ are sorted by their x and y coordinates beforehand. Any optimal sorting algorithm such as a heap sort or a merge sort does this task in $O(n \log n)$ time. The Figure 2.6 shows the steps of constructing dividing line.

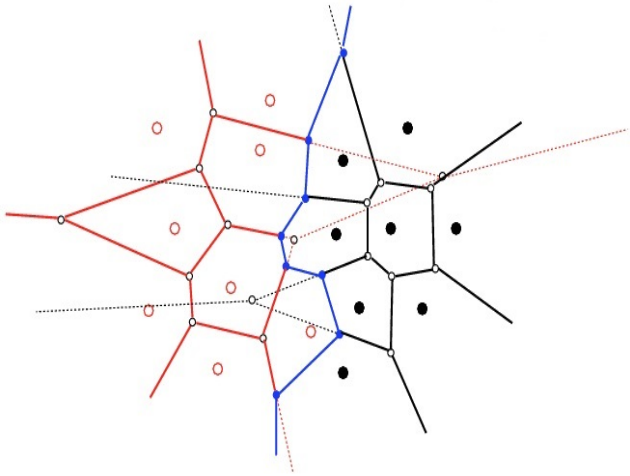


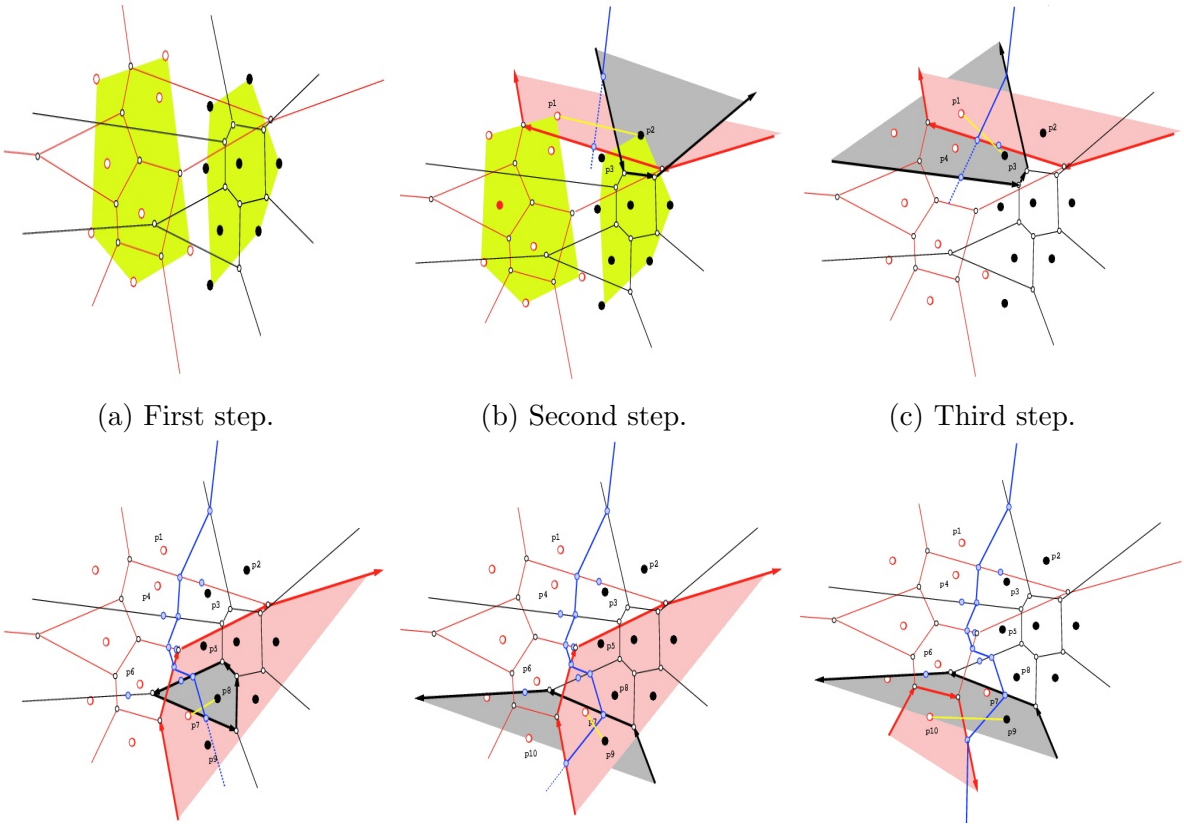
Figure 2.5: Dividing the set of sites into two subsets [4].

2.4.4 Centroidal Voronoi Tesselation (CVT)

Centroidal Voronoi tessellation (CVT) is a special type of Voronoi tessellation. It can be viewed as an optimal partition corresponding to an optimal distribution of generators. Centroidal Voronoi tessellation is not unique. A Voronoi tessellation is called centroidal when the generating point of each Voronoi cell is also its center of mass. A number of algorithms can be used to generate centroidal Voronoi tessellations, including Lloyd's algorithm for K-means clustering. Gersho's conjecture, proven for one and two dimensions, says that "asymptotically speaking, all cells of the optimal centroidal Voronoi tessellation, while forming a tessellation, are congruent to a basic cell which depends on the dimension. In two dimensions, the basic cell for the optimal centroidal Voronoi tessellation is a regular hexagon. Centroidal Voronoi tessellations are useful in data compression, optimal quadrature, optimal quantization, clustering, and optimal mesh generation [71]. Many patterns seen in nature are closely approximated by a centroidal Voronoi tessellation. Examples of this include the Giant's Causeway, the cells of the cornea, and the breeding pits of the male tilapia [71]. A weighted centroidal Voronoi diagrams is a CVT in which each centroid is weighted according to a certain function.

Lloyd's Algorithm

Lloyd's algorithm, also known as Voronoi iteration or relaxation, is an algorithm named after Stuart P. Lloyd for finding evenly-spaced sets of points in subsets of Euclidean spaces, and partitions of these subsets into well-shaped and uniformly sized convex cells [15]. This algorithm repeatedly finds the centroid of each set in the partition, and then re-partitions the input based on which of these centroids is closest. The input of Lyod's algorithm is a continuous geometric region rather than a discrete set of points. Thus, when re-partitioning the input, Lloyd's algorithm uses Voronoi diagrams rather than simply determining the nearest center to each of a finite set of points.



(a) First step. (b) Second step. (c) Third step.

(d) Third last step. (e) Second last step. (f) Last step.

Figure 2.6: Constructing the dividing chain [4].

Although the algorithm may be applied most directly to the Euclidean plane, similar algorithms may also be applied to higher-dimensional spaces or to spaces with other non-Euclidean metrics. Lloyd's algorithm can be used to construct close approximations to centroidal Voronoi tessellations of the input, [71] which can be used for quantization, dithering, and stippling. Other applications of Lloyd's algorithm include smoothing of triangle meshes in the finite element method.

Algorithm Description: Lloyd's algorithm starts by an initial placement of some number k of point sites in the input domain. In mesh smoothing applications, these would be the vertices of the mesh to be smoothed; in other applications they may be placed at random, or by intersecting a uniform triangular mesh of the appropriate size with the input domain. It then repeatedly executes the following relaxation steps (Figure 2.7):

- The Voronoi diagram of the k sites is computed.
- Each cell of the Voronoi diagram is integrated and the centroid is computed.
- Each site is then moved to the centroid of its Voronoi cell.

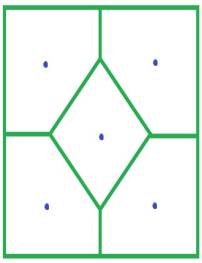
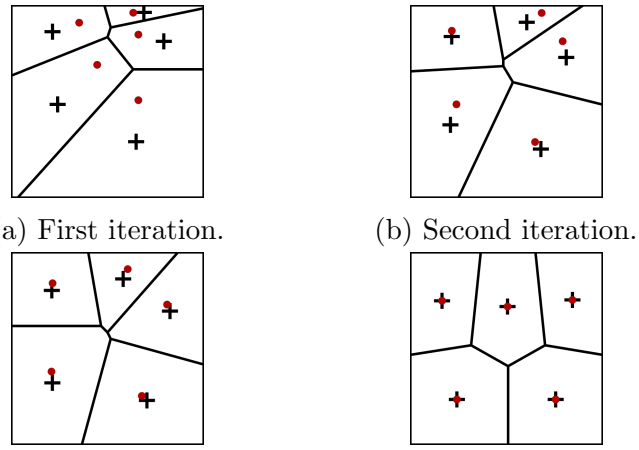


Figure 2.7: Centroidal Voronoi Tesselation(CVT) .



(a) First iteration. (b) Second iteration.

(c) Third iteration. (d) Last iteration.

Figure 2.8: Lloyd's algorithm [5].

Convergence: Each time a relaxation step is performed, the generators are left in a slightly more even distribution: closely spaced generators move farther apart, and widely spaced generators move closer together. In one dimension, this algorithm has been shown to converge to a centroidal Voronoi diagram, also named a centroidal Voronoi tessellation [18]. In higher dimensions, some slightly weaker convergence results are known [72] [73]. The algorithm converges slowly or, due to limitations in numerical precision, may not converge. Therefore, real-world applications of Lloyd's algorithm typically stop once the distribution is "good enough." One common termination criterion is to stop when the maximum distance moved by any site in an iteration falls below a preset threshold. Convergence can be accelerated by over-relaxing the points, which is done by moving each point k times the distance to the center of mass, typically using a value slightly less than 2 for k [74].

Applications: Lloyd's method was originally used for scalar quantization, but it is clear that the method extends for vector quantization as well. As such, it is extensively used in data compression techniques in information theory. Lloyd's method is used in computer graphics because the resulting distribution has blue noise characteristics, meaning there

are few low-frequency components that could be interpreted as artifacts. It is particularly well-suited to picking sample positions for dithering. Lloyd's algorithm is also used to generate dot drawings in the style of stippling [75]. In this application, the centroids can be weighted based on a reference image to produce stipple illustrations matching an input image. In the finite element method, an input domain with a complex geometry is partitioned into elements with simpler shapes; for instance, two-dimensional domains (either subsets of the Euclidean plane or surfaces in three dimensions) are often partitioned into triangles. It is important for the convergence of the finite element methods that these elements be well shaped; in the case of triangles, often elements that are nearly equilateral triangles are preferred. Lloyd's algorithm can be used to smooth a mesh generated by some other algorithm, moving its vertices and changing the connection pattern among its elements in order to produce triangles that are more closely equilateral [76]. These applications typically use a smaller number of iterations of Lloyd's algorithm, stopping it to convergence, in order to preserve other features of the mesh such as differences in element size in different parts of the mesh [77].

Different Distances: Lloyd's algorithm is usually used in a Euclidean space. The Euclidean distance plays two roles in the algorithm: it is used to define the Voronoi cells, but it also corresponds to the choice of the centroid as the representative point of each cell, since the centroid is the point that minimizes the average squared Euclidean distance to the points in its cell. Alternative distances, and alternative central points than the centroid, may be used instead. For example, Hausner used a variant of the Manhattan metric (with locally-varying orientations) to find a tiling of an image by approximately-square tiles whose orientation aligns with features of an image, which he used to simulate the construction of tiled mosaics [78]. In this application, despite varying the metric, Hausner continued to use centroids as the representative points of their Voronoi cells. However, for metrics that differ more significantly from Euclidean, it may be appropriate to choose the minimizer of average squared distance as the representative point, in place of the centroid [79].

2.5 Locational Optimization

Locational optimization problem is about an optimal location or an optimal configuration of facilities that is found in a continuum on a plane or a network. In this section we describe some known facts about a meaningful optimization problem which is related to our work such as, centroidal Voronoi tessellations that could be used in order to find optimal configuration of some agents to perform the environmental coverage.

Consider Ω to be a convex polygon in $\Re^2$. We call a map $\phi : \Omega \times \Re_+ \to \Re_+$ a distribution risk density which represents the probability measure that some events take place over Ω. In equivalent words, we can consider Ω to be the bounded support of the function ϕ [80] [81]. Let $\mathbf{x} = (\mathbf{x}_1^T, ..., \mathbf{x}_n^T)^T$ be the location of n agents moving in the space Ω. The loss of resolution and noise cause the sensing performance at point $\mathbf{q} \in \Omega$, taken from ith agent

at position $\mathbf{x}_i$, changes with the Euclidean distance $r_i = \| \mathbf{q} - \mathbf{x}_i \|$ between $\mathbf{q}$ and $\mathbf{x}_i$. The sensing performance is represented by a strictly decreasing and differentiable function $f(r_i) = \alpha e^{-\beta r_i^2}$, with $f : \Re_+ \to \Re_+$ and $\alpha, \beta > 0$. Accordingly, $f(r_i)$ provides a quantitative assessment of how poor(rich) is the agents sensing performance by increasing (decreasing) the distance r_i.

Consider the task of maximizing locational optimization function

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

where $\mathcal{W} = \{\mathcal{W}_1, ..., \mathcal{W}_n\}$ is a partition (or a region) of Ω, and the ith sensor is responsible for measurements over its "dominance partition" $\mathcal{W}_i$. The equation of $\mathcal{H}$ describes the richness of the coverage in domain Ω. In fact, the higher value of $\mathcal{H}$ indicates that the corresponding distribution of agents attains better coverage of the area Ω. Therefore, the statement of locational optimization problem is: given the density function, we try to find spatially distributed agents such that the coverage $\mathcal{H}$ is maximized as t goes to infinity.

2.5.1 Voronoi Partition

The optimal partition of Ω at fixed sensors location, is the Voronoi partition which is defined as $\mathcal{V}(\mathbf{x}) = (\mathcal{V}_1(\mathbf{x}), \cdots, \mathcal{V}_n(\mathbf{x}))$ such that $\Omega \equiv \bigcup_{i=1}^{n} \mathcal{V}_i(\mathbf{x})$. The $\mathbf{x}_j$ is called a neighbor of $\mathbf{x}_i$ when two Voronoi regions $\mathcal{V}_j$ and $\mathcal{V}_i$ share an edge, and vice versa. The ith Voronoi cell is defined by

$$\mathcal{V}_i(\mathbf{x}) = \{\mathbf{q} \in \Omega : f(r_i) \geq f(r_j), \forall i, j \in \mathcal{L}, i \neq j\}, \tag{2.5}$$

where $\mathcal{L} \equiv \{1, 2, \cdots, n\}$ is the set of agents unique identifiers. $\mathcal{V}_i(\mathbf{x})$ describes an area where each point is best sensed by the ith agent than by all other agents. The boundary of each Voronoi cell ($\mathcal{V}_i$) has a convex polygon shape because the domain (Ω) is a convex polygon in a finite dimensional Euclidean space. It is known that the nearest generator $\mathbf{x}_j$ to $\mathbf{x}_i$ is a neighbor and the average number of neighbors on $\Omega \subset \Re^2$ is six. So we can write

$$\mathcal{H}_\mathcal{V} = \mathcal{H}(\mathbf{x}, \mathcal{V}(\mathbf{x})).$$

According to the definition of Voronoi diagram, $\max_{i \in \{1,...,n\}} f(r_i) = f(r_j)$ for all $\mathbf{q} \in \mathcal{V}_j$. Therefore

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

This is the normal way in which the problem is discussed in the facility location and operations research literature [71]. Remarkably [71] [80],

$$\frac{\partial \mathcal{H}_\mathcal{V}}{\partial \mathbf{x}_i}(\mathbf{x}) = \frac{\partial \mathcal{H}}{\partial \mathbf{x}_i}(\mathbf{x}, \mathcal{V}(\mathbf{x})) = \int_{\mathcal{V}_i} \frac{\partial}{\partial \mathbf{x}_i} f(r_i)\phi(\mathbf{q})d\mathbf{q}. \tag{2.7}$$

The computation of $\frac{\partial \mathcal{H}_\mathcal{V}}{\partial \mathbf{x}_i}(\mathbf{x})$ is decentralized in the sense of Voronoi. In addition, we can conclude some smoothness features of $\mathcal{H}_\mathcal{V}$: the function $\mathcal{H}_\mathcal{V}$ is at least continuously differentiable on Ω, because the Voronoi cell $\mathcal{V}$ depends at least continuously on $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, for all $\mathbf{x}_i \neq \mathbf{x}_j$.

Centroidal Voronoi Tessellation

The centroid (or center of mass) and the mass of Voronoi cell $\mathcal{V}_i$ are defined as

$$\begin{aligned} M_\phi(\mathcal{V}_i(\mathbf{x})) &= \int_{\mathcal{V}_i} \phi(\mathbf{q}, \mathbf{x}_i) d\mathbf{q}, \qquad \textit{and} \\ \mathbf{C}_\phi(\mathcal{V}_i(\mathbf{x})) &= \frac{1}{m_\phi(\mathcal{V}_i(\mathbf{x}))} \int_{\mathcal{V}_i} \mathbf{q}\phi(\mathbf{q}, \mathbf{x}_i) d\mathbf{q}, \end{aligned} \tag{2.8}$$

2.5.2 Distributed Gradient

Here we are going to explain some important factors about the coverage metric [82]

$$\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} = \int_{\mathcal{V}_i} \frac{\partial}{\partial \mathbf{x}_i} f(r_i)\phi(\mathbf{q}) d\mathbf{q}. \tag{2.9}$$

According to Divergence theorem [82]

$$\begin{aligned} \frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} = &\int_{\mathcal{V}_i} \frac{\partial}{\partial \mathbf{x}_i} f(r_i)\phi(\mathbf{q}) d\mathbf{q} \\ &+ \int_{\partial\mathcal{V}_i \cap \partial\Omega} f(r_i)\phi(\mathbf{q}) \frac{\partial \mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})}{\partial \mathbf{x}_i} \mathbf{n}_{\partial\mathcal{V}_i} d\mathbf{q} \\ &+ \sum_{j \in \mathcal{N}_i} \int_{l_{ij}} (f(r_i) - f(r_j))\phi(\mathbf{q}) \frac{\partial \mathbf{q}_{lij}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} \mathbf{n}_{lij} d\mathbf{q} \end{aligned} \tag{2.10}$$

where $\partial\mathcal{V}_i$ and $\partial\Omega$ are the boundary of $\mathcal{V}_i$ and Ω, respectively. $\mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})$ means a point $\mathbf{q} \in \partial\mathcal{V}_i$, and $\mathbf{n}_{\partial\mathcal{V}_i}$ is 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 $\mathcal{V}_i$, $\mathcal{N}_i = \{j | \mathcal{V}_i \cap \mathcal{V}_j \neq \varnothing\}$. The set of points on the Voronoi boundary shared by agents i and j are denoted by $l_{ij} = \mathcal{V}_i \cap \mathcal{V}_j$. Then $\mathbf{q}_{lij}(\mathbf{x}_i, \mathbf{x}_j)$ is a point on that shared boundary, and $\mathbf{n}_{lij}$ is the unit normal of l_{ij} from $\mathbf{x}_i$ to $\mathbf{x}_j$. $\frac{\partial \mathbf{q}_{lij}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i}$ and $\frac{\partial \mathbf{q}_{\partial\mathcal{V}_i}(\mathbf{x})}{\partial \mathbf{x}_i}$ are $N \times N$ matrices. We have

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

because points on the boundary of environment do not change position as a function of $\mathbf{x}_i$. Therefor the second term vanishes. Also based on the definition of l_{ij}, $r_i = r_j$ $\forall \mathbf{q} \in l_{ij}$, so the last term of the equation (2.10) vanishes.

As a result the gradient

$$\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = [\cdots \frac{\partial \mathcal{H}^T}{\partial \mathbf{x}_i} \cdots]^T,$$

is distributed among the agents in the sense that an agent i can compute its own gradient component, $\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i}$, using only information relevant to its Voronoi cell. Thus the derivation of $\mathcal{H}$ with respect to $\mathbf{x}_i$ is

$$\begin{aligned}
\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} &= \int_{\mathcal{V}_i} \frac{\partial}{\partial \mathbf{x}_i} f(r_i)\phi(\mathbf{q})d\mathbf{q} \\
&= \int_{\mathcal{V}_i} \frac{\partial}{\partial \mathbf{x}_i} (\alpha e^{-\beta r_i^2})\phi(\mathbf{q})d\mathbf{q} \\
&= \int_{\mathcal{V}_i} \frac{\partial}{\partial \mathbf{x}_i} (\alpha e^{-\beta \|\mathbf{q}-\mathbf{x}_i\|^2})\phi(\mathbf{q})d\mathbf{q} \\
&= \int_{\mathcal{V}_i} -2(\mathbf{q}-\mathbf{x}_i)(\alpha\beta e^{-\beta \|\mathbf{q}-\mathbf{x}_i\|^2})\phi(\mathbf{q})d\mathbf{q}.
\end{aligned} \tag{2.11}$$

By following the equation (2.8) and letting

$$\tilde{m}_i = \int_{\mathcal{V}_i} \tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q} \qquad \tilde{\mathbf{c}}_i = \frac{1}{\tilde{m}_i}\int_{\mathcal{V}_i} \mathbf{q}\tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q}, \tag{2.12}$$

with generalized risk density function

$$\tilde{\phi}(\mathbf{q}, \mathbf{x}_i) = -2\phi(\mathbf{q})(\partial f(r_i)/\partial(r_i^2)), \tag{2.13}$$

then the equation (2.11) can be written as

$$\begin{aligned}
\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} &= \int_{\mathcal{V}_i} -(\mathbf{q}-\mathbf{x}_i)\tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q} \\
&= \int_{\mathcal{V}_i} -\mathbf{q}\tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q} - \int_{\mathcal{V}_i} \mathbf{x}_i\tilde{\phi}(\mathbf{q}, \mathbf{x}_i)d\mathbf{q} = \tilde{m}_i(\tilde{\mathbf{c}}_i - \mathbf{x}_i),
\end{aligned} \tag{2.14}$$

where $\tilde{m}_i$ and $\tilde{\mathbf{c}}_i$ are generalized mass and generalized centroid of Voronoi cell $\mathcal{V}_i$ respectively.

Therefore, the centroid of Voronoi cell is local optimum point for the locational optimization function $\mathcal{H}_\mathcal{V}$ (which is not necessarily unique), i.e., each location $\mathbf{x}_i$ is the generator for the Voronoi cell $\mathcal{V}_i$ and it is its centroid simultaneously

$$\mathbf{c}_\phi(\mathcal{V}_i(\mathbf{x})) = \arg\max_{\mathbf{x}_i} \mathcal{H}_\mathcal{V}(\mathbf{x}).$$

Accordingly, centroidal Voronoi tessellations (partitions) are the critical partitions and points for $\mathcal{H}$. If the agent's configuration gives rise to a centroidal Voronoi tessellation, we will refer to it as a centroidal Voronoi configuration. Centroidal Voronoi configurations

depend on the specific distribution density function ϕ, and an arbitrary pair (Ω, ϕ) admits, in general, multiple centroidal Voronoi configurations [71] [80].

2.6 Coverage Control and Optimization

In this section, we explain how to compute location of agents in order to maximize the coverage $\mathcal{H}$. The continuous-time version of the classic Lloyd algorithm [15] is proposed in this thesis. The classic Lloyd algorithm is a discrete method to compute CVT. This algorithm computes the Voronoi regions, then computes the centroids and finally moves each generator point to the corresponding centroid.

We study on a multi-agent system defining a group of n-homogenous mobile agents, $i = 1, \cdots, n$, where their states are updated based on the first order kinematic state model. Assume the agents location follows a first-order dynamical behavior defined by

$$\dot{\mathbf{x}}_i = \mathbf{u}_i, \tag{2.15}$$

where $\mathbf{x}_i(0) = \mathbf{x}_i^0, \mathbf{x}_i(t) \in \Re^2$ is the state at time $t \in \Re^+$, $\mathbf{x}_i^0$ is the initial state, and $\mathbf{u}_i(t) \in \Re^2$ is the velocity of the ith agent at time t. Let $(\mathbf{x}_1(t), \cdots, \mathbf{x}_n(t))^\mathsf{T} \equiv \mathbf{x}(t) \in \Re^{2n}$ be the state of the network (collection of agent states). The state trajectories of a team of n agents by using the model (2.15), can be defined as

$$\dot{\mathbf{x}}(t) = \mathbf{u}(t), \mathbf{x}^0 = \mathbf{x}(0), \tag{2.16}$$

where $(\mathbf{u}_1(t), \cdots, \mathbf{u}_n(t))^\mathsf{T} \equiv \mathbf{u}(t) \in \Re^{2n}$.

In order to optimize the coverage, we face two different cases: I)Time-Invariant Case, II)Time-Variant Case.

2.6.1 Time-Invariant Case

Consider $\mathcal{H}$ a cost function to be maximized and impose that the location $\mathbf{x}_i$ follows a gradient ascent. Given $\phi(\mathbf{q}, t) = \phi(\mathbf{q}), \mathbf{q} \in \Omega, \kappa > 0$, the equation (2.16) of agent i and using the equation (2.14), the feedback law

$$\mathbf{u}_i = \kappa \frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} = \kappa m_\phi(\mathcal{V}_i)(\mathbf{c}_\phi(\mathcal{V}_i) - \mathbf{x}_i) \tag{2.17}$$

maximizes the coverage metric defined in (2.4) with asymptotic convergence of agent i to its generalized Voronoi centroid (critical point) $\mathbf{c}_\phi(\mathcal{V}_i)$. Some methods could be used in order to guarantee the exponential convergence by the proposed feedback law, such as I)the Hessian of $\mathcal{H}$ be negative definite at $\mathbf{c}_\phi(\mathcal{V}_i)$, and II)the more general method which is Lasalle's principle(Krasovskii-Lasalle principle). In our work that is based on a kinematically actuated system, we use Lasalle's principle and Barbalat's lemma in order

to prove the stability of agents trajectory around the centroid in time-invariant and time-variant case respectively.

LaSalle's Principle

The LaSalle's principle (also known as the invariance principle) is a criterion for the asymptotic stability of a system. the LaSalle's principle has two versions, Local and Global. The agents' configuration in this work is defined as Centroidal Voronoi Tessellation (CVT) and it was mentioned in section (2.4.4) that CVT is not unique, so the Local optimization can be used to optimize the coverage, therefore the Local version of LaSalle's principle can be applied in our case.

Local version of the LaSalle's principle: Let assume $V(x)$ as the Lyapunov function, if

$$\begin{aligned} &V(\mathbf{x}) > 0, \quad \text{when} \quad \mathbf{x} \neq \mathbf{0} \\ &\dot{V}(\mathbf{x}) \leq 0 \end{aligned} \tag{2.18}$$

hold only for $\mathbf{x}$ in some neighborhood D of the origin, and the set $\{\dot{V}(\mathbf{x}) = 0\} \bigcap D$ does not include any trajectories of the system besides the trajectory $\mathbf{x}(t) = 0$, $t \geq 0$, then the origin is locally asymptotically stable based on the local version of LaSalle's principle.

In our work consider the candidate Lyapunov function

$$V(\mathbf{x}) = \frac{1}{\mathcal{H}(\mathbf{x})}. \tag{2.19}$$

The functions f and ϕ were defined as positive definite functions, therefore $\mathcal{H}$ and V should be positive definite functions too and consequently the proposed Lyapunov function $V(\mathbf{x})$ is a positive definite function (satisfies the first condition of LaSalle's principle). in the time-invariant case,

$$\dot{V}(\mathbf{x}) = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \dot{\mathbf{x}} = \frac{-\frac{\partial \mathcal{H}(\mathbf{x})}{\partial \mathbf{x}}}{\mathcal{H}(\mathbf{x})^2} \cdot \dot{\mathbf{x}},$$

where by using the equations (2.14), (2.15) and (2.17)

$$\dot{V}(\mathbf{x}) = \frac{-m_\phi(\mathcal{V})(\mathbf{c}_\phi(\mathcal{V}) - \mathbf{x})}{\mathcal{H}(\mathbf{x})^2} \cdot \kappa m_\phi(\mathcal{V})(\mathbf{c}_\phi(\mathcal{V}) - \mathbf{x}) = -\frac{\kappa m_\phi(\mathcal{V})^2 \parallel \mathbf{c}_\phi(\mathcal{V}) - \mathbf{x} \parallel^2}{\mathcal{H}(\mathbf{x})^2}. \tag{2.20}$$

Hence it is clear that $\dot{V}(\mathbf{x})$ is a negative definite function. As a result all the conditions of LaSalle's principle (2.18) are satisfied and the asymptotic stability of system via the feedback law (2.17) is ensured.

2.6.2 Time-Variant Case

Some techniques have been introduced in order to optimize the coverage control with time-variant risk density, which are reviewed in this section. Our work on non-autonomous coverage optimization will be illustrated in the next chapter.

Cortes and Bullo [80] focus on vehicles that perform distributed sensing tasks and refer to them as active sensor networks. Such systems are being developed for applications in remote autonomous surveillance, exploration, information gathering, and automatic monitoring of transportation systems. For active sensor networks, they envisioned the need for a distributed control and coordination architecture: a wireless network provides the vehicles with the ability to share some information, but no overall leader might be present to coordinate the group. As the vehicle network evolves in time, the ad-hoc communication graph and neighborhood relationships change. It is interesting therefore to design distributed algorithms for ad-hoc multi-vehicle networks. They describe algorithms to compute location of sensors that minimize the cost function. They propose a continuous-time version of the classic Lloyd algorithm. In their setting, both the positions and partitions evolve in continuous time, whereas Lloyd algorithm for vector quantization is usually designed in discrete time. Similarly to the original Lloyds scheme, the proposed algorithm is a gradient descent flow.

An important problem in this context is to determine how best to distribute sensors over a given area in which the observational field is distributed so that the likelihood of detecting an event of interest is maximized. If the probability distribution of the event is uniform over the area, then the optimal solution is uniform coverage, i.e., uniform distribution of sensors. On the other hand, if this probability distribution is nonuniform, then the sensors should be more (less) densely distributed in subregions with higher (lower) event probability. Further, if the probability distribution changes with time, then the nonuniform distribution should likewise change with time. A related coverage problem derives from the classic objective analysis (OA) mapping error in problems of sampling (possibly time-varying) scalar fields, e.g., temperature in the ocean. OA is linear statistical estimation based on specified field statistics, and the mapping error provides a measure of statistical uncertainty of the model as a function of where and when the data is taken. Since reduced uncertainty, equivalent to increased entropic information, implies better measurement coverage, OA mapping error can be used as a coverage metric. If the priori error correlation between any two points in the plane is homogeneous and isotropic, then uniform coverage will be optimal initially. However, the optimal coverage solution will not be static (unless the scalar field of interest changes very quickly), since, once a particular location has been sampled, it should not be sampled again until the measurement value has decayed sufficiently. Robotic vehicles carrying sensors in space, in the air, on land, and in the ocean make possible mobile sensor networks that can adapt to change coverage requirements. Given a coverage metric that is independent of time and of the history of samples taken, the goal is to design coordinated control dynamics for arbitrary initial conditions. The second objective is to extend these coordinated control dynamics to the case in which the coverage metric definition changes in time or, as in the case of the OA error metric, as a function of past measurement locations and times. In [83], they concentrated exclusively

on two-dimensional planar regions, and they proposed an approach to coverage control that makes use of existing algorithms designed for uniform coverage and extends these to nonuniform metrics. They are particularly interested in metrics defined in terms of non-Euclidean distance functions that effectively stretch and shrink space in lower and higher density regions of a given space. This yields optimal configurations where the resource or information is evenly distributed among the agents. Non-Euclidean distance metrics present challenges to existing techniques. For example, in the case of [16] and [83], computing Voronoi cells with non-Euclidean metrics is computationally complex. For each point on a dense grid, one needs to compute the (non-Euclidean) distance to each agent and find the minimum. Computing Voronoi cells for a non-Euclidean metric is therefore much more demanding than the corresponding problem with an Euclidean metric where the polygonal boundaries of the Voronoi cells can be computed directly. The first step in their method is to compute a nonuniform change of coordinates on the original compact set with a non-Euclidean metric that maps to a new compact set with a near Euclidean metric. Such a map is called a cartogram which is computed the from a diffusion equation. A uniform control law can be used in the cartogram space since the metric in this space is almost Euclidean. The preimage of the control law yields convergent dynamics in the original space. Under certain conditions was proved that these convergent dynamics optimize the nonuniform coverage metric. They showed how to extend the approach to the case of a time-varying metric. A limitation of this approach is the centralized computation of the cartogram. However, they note the density function does not need to be known a priori; it can be measured or computed by the agents in real time. For example, the changing OA metric can be computed only on the fly since it is a function of where and when samples have been taken.

The problem of optimally covering an environment while tracking intruders into the problem of covering environments with time-varying density functions was presented in [82]. The proposed framework allows for the coupling of the three previously mentioned basic subtasks: target assignment, coverage, and tracking in an elegant distributed manner. For the first time, a decentralized controller with guaranteed exponential convergence was also derived.

Lee and Egerstedt [81] presented an approach to controlling a system of multiple agents by choosing a time-varying density function. A control law that causes the agents to track the density function while providing optimal coverage of the density function was proposed. The intention in their work is to control the multi-agent system to provide surveillance over the domain of interest by employing optimal coverage ideas on general time-varying density functions. This approach can have applications in diverse situations where they specify what time-varying region in the domain is needed to be more considered by robots. One example is monitoring of oil spills. A search and rescue scenario is another example, where the density function represents the probability of a lost person being at a certain point in an area. A variety of military applications may exist as well. The goal of optimal coverage was achieved in the sense that the agents are to first converge to CVT under time-invariant choice of density function, and then the density function is switched to a time-varying one. Under the proposed algorithm, the agents maintain CVT for time-varying density functions if the agents were in CVT initially. The proposed algorithm was

used to achieve desired behavior of the network of agents while viewing the choice of the time-varying density function as an input. The proposed algorithm places no additional assumptions on the time evolution of the density function, at the expense of becoming a centralized control law. The results from robot implementation in their work show that the proposed algorithm guides the agents well over the chosen density functions, and that the effectiveness of the coverage is higher than other comparable algorithms.

Chapter 3

Optimal Motion Control for a Non-Autonomous Coverage

3.1 Introduction

In Chapter 2 we reviewed well established coverage optimization algorithms that apply to autonomous coverage metrics. The centroidal Voronoi tessellation plays a significant role in determining optimal configurations of multiple agents with respect to a coverage metric that encodes their sensing performances and a risk density function defined in the operating environment. However, in most cases, the risk density were assumed to be constant during the operating time interval which might be too restrictive for many applications. In this chapter, we present a generalized centroidal Voronoi tessellation technique with a group of mobile autonomous agents to maximize a non-autonomous coverage function with time-variant risk density. The stability of the system around system's trajectories is addressed in the framework of Barbalat's lemma. Numerical results and different scenarios are simulated and discussed.

3.2 Agent Kinematics and Motion Control

Section (3.2) describes the case when the density function is time varying and the coverage metric $\mathcal{H}$ is non-autonomous. For this case, the trajectories described by the feedback law (2.17) need to be redefined in order to ensure convergence to time-varying generalized Voronoi centroids.

3.2.1 Modelling Time-Varying Risk Density

Various density functions that lead the multi-vehicle network to predetermined geometric patterns could be presented, such as simple density functions that lead to segments, ellipses,

polygons, or uniform distributions inside convex environments [80]. In our scenarios[1], we are interested in optimizing a coverage metric defined in the region Ω. In this thesis, the density function is affected by the motion of uncontrolled objects inside the domain. In typical area coverage and protection scenarios, these objects are external threats to be engaged and eliminated by the set of autonomous agents distributed in the area. Therefore the agents act as interceptors, and the external threads are labeled as targets. Since the kinematics of the targets is uncontrolled by the interceptors, and since the targets' motion influences the coverage metric by varying the risk density function inside the area to be covered, the metric has a time dependency that cannot be described through the state of the agents/interceptors, and therefore it is non-autonomous. In order to describe the influence of the targets on the risk density function, we adopt a function that attains the maximum value at the position of target(s) and decreases to a negligible value (almost zero) in a far distance from the center of target. One of the good options for our desired scenarios in order to design a time-varying risk density is the well-known normal distribution(Gaussian distribution) which has a bell shape [80].

Let the density $\phi(\mathbf{q}, t)$ be defined by the following expression

$$\phi(\mathbf{q}, t) = \phi_c(\mathbf{q}) + \sum_{j=1}^{L} \phi_j(\mathbf{q}, t), \tag{3.1}$$

where L is the number of targets, and $\phi_c : \Omega \longrightarrow \Re^+$ is a biased density which can be described as the nonuniform distribution of density in the absence of targets ($L = 0$). As it was explained above, we adopt a Gaussian distribution function, $\phi_j(\mathbf{q}, t)$, that represents the risk which is caused by the jth target,

$$\phi_j(\mathbf{q}, t) = \sum_{j=1}^{L} e^{-\frac{1}{2}(\frac{(q_x - \tau_{jx})}{\sigma})^2 - \frac{1}{2}(\frac{(q_y - \tau_{jy})}{\sigma})^2}, \tag{3.2}$$

where $\tau_j = [\tau_{jx}, \tau_{jy}]^T$ is the position vector of jth mobile target in Cartesian coordinates, and σ is a positive constant. The σ is shape parameter which determines how the risk function can get wider\shorter or narrower\higher. By substituting the equation (3.2) in equation (3.1), the total density function $\phi(\mathbf{q}, t)$ is

$$\phi(\mathbf{q}, t) = \phi_c(\mathbf{q}) + \sum_{j=1}^{L} e^{-\frac{1}{2}(\frac{(q_x - \tau_{jx})}{\sigma})^2 - \frac{1}{2}(\frac{(q_y - \tau_{jy})}{\sigma})^2}. \tag{3.3}$$

[1]The scenarios that were investigated in this thesis, will be explained in the section (3.3)

3.2.2 Coverage Optimization

Let $\phi(\mathbf{q},t)$ be the density defined by (3.3), $\mathcal{H}$ be the non-autonomous coverage metric defined by

$$\mathcal{H}(\mathbf{x},\mathcal{V},t)=\sum_{i=1}^{n}\mathcal{H}_i(\mathbf{x}_i,\mathcal{V}_i,t)=\sum_{i=1}^{n}\int_{\mathcal{V}_i}f(r_i)\phi(\mathbf{q},t)d\mathbf{q}. \tag{3.4}$$

In this case the feedback law (2.17) needs to be modified as in that form, due to the non-autonomous part of the coverage metric, maximization of $\mathcal{H}$, and therefore optimality, is not guaranteed. Indeed, by taking the time derivative of the coverage metric along the trajectories generated by the feedback law (2.17) we obtain

$$\dot{\mathcal{H}}=\kappa m_\phi^2(\mathcal{V}_i)\parallel\mathbf{c}_\phi(\mathcal{V}_i)-\mathbf{x}_i\parallel^2+\frac{\partial\mathcal{H}}{\partial t}.$$

In order to account for the non-autonomous part of the coverage metric, for the multi-agent kinematics in (2.15) consider the individual feedback laws

$$\mathbf{u}_i(t)=\frac{\partial\mathcal{H}/\partial\mathbf{x}_i}{\parallel\partial\mathcal{H}/\partial\mathbf{x}_i\parallel^2}\left(\kappa\parallel\tilde{\mathbf{c}}_i-\mathbf{x}_i(t)\parallel^2-\frac{\partial\mathcal{H}_i}{\partial t}\right), \tag{3.5}$$

where

$$\frac{\partial\mathcal{H}_i}{\partial t}=\int_{\mathcal{V}_i}f(r_i)\frac{\partial\phi}{\partial t}(\mathbf{q},t)d\mathbf{q}, \tag{3.6}$$

and the partial derivative of $\phi(\mathbf{q},t)$ with respect to time using the equation (3.3) is given by:

$$\frac{\partial\phi(\mathbf{q},t)}{\partial t}=\sum_{j=1}^{L}\Big(\frac{\partial\tau_{jx}}{\partial t}\frac{(q_x-\tau_{jx})}{\sigma^2}+\frac{\partial\tau_{jy}}{\partial t}\frac{(q_y-\tau_{jy})}{\sigma^2}\Big)e^{-\frac{1}{2}\left(\frac{(q_x-\tau_{jx})}{\sigma}\right)^2-\frac{1}{2}\left(\frac{(q_y-\tau_{jy})}{\sigma}\right)^2}, \tag{3.7}$$

where $\frac{\partial\tau_j}{\partial t}=[\frac{\partial\tau_{jx}}{\partial t},\ \frac{\partial\tau_{jy}}{\partial t}]^T=\mathbf{v}_j$ is the Cartesian velocity vector of jth mobile target. By substituting equation (3.7) into equation (3.6) yields

$$\frac{\partial\mathcal{H}_i}{\partial t}=\frac{1}{\sigma^2}\int_{\mathcal{V}_i}f(r_i)\sum_{j=1}^{L}\mathbf{v}_j\cdot(\mathbf{q}-\tau_j)\phi_j(\mathbf{q},t)d\mathbf{q},$$

then interchanging the summations and considering the equation (2.8), the equation above can be written as

$$\frac{\partial\mathcal{H}_i}{\partial t}=\frac{1}{\sigma^2}\sum_{j=1}^{L}m_{ij}\mathbf{v}_j\cdot(\mathbf{c}_{ij}-\tau_j), \tag{3.8}$$

where

$$m_{ij} = \int_{\mathcal{V}_i} f(r_i)\phi_j d\mathbf{q} \quad and \quad \mathbf{c}_{ij} = \frac{1}{m_{ij}} \int_{\mathcal{V}_i} f(r_i)\phi_j \mathbf{q} d\mathbf{q} \tag{3.9}$$

are modified mass and centroid respectively.

We will prove in the following section that the feedback laws (3.5) for $\kappa > 0$, maximize the coverage $\mathcal{H}$ in (3.4) with convergence of interceptors $\mathbf{x}_i$ to generalized Voronoi centroids $\tilde{\mathbf{c}}_i$ for $i = 1, ..., n$. According to equation (2.12), the centroids of Voronoi cells are affected by the risk, so they will be updated by changing the risk.

3.2.3 Asymptotic Convergence

For the time-varying case, in order to prove the stability of the agent's trajectory, we use Barbalat's lemma. This lemma says that if there exists a Lyapunov function $V(\mathbf{x}, t)$ satisfying the following conditions:

- $V(\mathbf{x}, t)$ is lower bounded
- $\dot{V}(\mathbf{x}, t)$ is negative semi-definite
- $\dot{V}(\mathbf{x}, t)$ is uniformly continuous in time (satisfied if $\ddot{V}(\mathbf{x}, t)$ is bounded)

then $\lim_{t\to\infty} \dot{V} = 0$. Therefore, the expression for $\dot{V}$ allows to establish asymptotic properties of the trajectories of the closed loop system.

Consider the candidate non-autonomous Lyapunov function

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

Functions f and ϕ as defined in equations (3.17) and (3.3) are positive definite, and therefore $\mathcal{H}$ and V are positive definite functions. Considering equation (3.4), it follows that V is continuous with respect to $\mathbf{x}$. The time derivative $\dot{V}$ is given by

$$\dot{V}(\mathbf{x}, t) = \frac{-\dot{\mathcal{H}}(\mathbf{x}, t)}{\mathcal{H}^2(\mathbf{x}, t)}, \tag{3.10}$$

where

$$\dot{\mathcal{H}}(\mathbf{x}, t) = \sum_{i=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}_i} \cdot \dot{\mathbf{x}}_i + \frac{\partial \mathcal{H}_i}{\partial t} \right), \tag{3.11}$$

and "." is the inner product between vectors.
By substituting the feedback law (3.5) in equation (3.11)

$$\dot{\mathcal{H}}(\mathbf{x}, t) = \sum_{i=1}^{n} \kappa \parallel \tilde{\mathbf{c}}_i - \mathbf{x}_i \parallel^2 . \tag{3.12}$$

Therefore $\dot{\mathcal{H}}$ is positive definite and it is bounded as long as the region Ω is bounded. $\dot{V}$ is negative definite and if we can prove that $\dot{V}$ is uniformly continuous, which holds if $\ddot{V}$ is bounded, then

$$\lim_{t\to\infty} \dot{V} = 0,$$

according to Barbalat's lemma. Considering the equation (3.10),

$$\ddot{V}(\mathbf{x},t) = \frac{-1}{\mathcal{H}^4(\mathbf{x},t)}(\ddot{\mathcal{H}}(\mathbf{x},t)\mathcal{H}^2(\mathbf{x},t) - 2\mathcal{H}(\mathbf{x},t)\dot{\mathcal{H}}^2(\mathbf{x},t)) = \frac{-1}{\mathcal{H}^3(\mathbf{x},t)}(\ddot{\mathcal{H}}(\mathbf{x},t)\mathcal{H}(\mathbf{x},t) - 2\dot{\mathcal{H}}^2(\mathbf{x},t)).$$

Hence in order to have a bounded $\ddot{V}(\mathbf{x},t)$, $\ddot{\mathcal{H}}(\mathbf{x},t)$ should be bounded, where

$$\ddot{\mathcal{H}}(\mathbf{x},t) = \sum_{j=1}^{n} \frac{\partial \dot{\mathcal{H}}}{\partial \mathbf{x}_j} \cdot \dot{\mathbf{x}}_j + \frac{\partial \dot{\mathcal{H}}}{\partial t} \tag{3.13}$$

Using equation (3.12),

$$\ddot{\mathcal{H}}(\mathbf{x},t) = \sum_{i=1}^{n}\sum_{j=1}^{n} \left(2\kappa(\tilde{\mathbf{c}}_i - \mathbf{x}_i)^{\mathsf{T}} \left(\frac{\partial \tilde{\mathbf{c}}_i}{\partial \mathbf{x}_j} - \mathbf{I}\delta_{ij} \right) \dot{\mathbf{x}}_j \right) + \sum_{i=1}^{n} \left(2\kappa(\tilde{\mathbf{c}}_i - \mathbf{x}_i)^{\mathsf{T}} \frac{\partial \tilde{\mathbf{c}}_i}{\partial t} \right), \tag{3.14}$$

where $\mathbf{I}$ and δ_{ij} are identity matrix and Kronecker delta respectively. According to [81] and [17] the term $(\frac{\partial \tilde{\mathbf{c}}_i}{\partial \mathbf{x}_j} - \mathbf{I}\delta_{ij})$ is bounded, also the term $2\kappa(\tilde{\mathbf{c}}_i - \mathbf{x}_i)$ is bounded as long as the domain Ω is bounded. Therefore, the boundedness of $\ddot{\mathcal{H}}(\mathbf{x},t)$ and consequently the stability of system depends on the boundedness of $\frac{\partial \tilde{\mathbf{c}}_i}{\partial t}$ and $\dot{\mathbf{x}}_j$. The expression for $\frac{\partial \tilde{\mathbf{c}}_i}{\partial t}$ is [81]

$$\frac{\partial \tilde{\mathbf{c}}_i}{\partial t} = \frac{\tilde{m}_i \int_{\mathcal{V}_i} \mathbf{q} \frac{\partial \tilde{\phi}_i}{\partial t}(\mathbf{q},t) d\mathbf{q} - \frac{\partial \tilde{m}_i}{\partial t} \int_{\mathcal{V}_i} \mathbf{q}\tilde{\phi}(\mathbf{q},t) d\mathbf{q}}{\tilde{m}_i^2}, \tag{3.15}$$

where, from (2.12) and (2.13)

$$\frac{\partial \tilde{m}_i}{\partial t} = \int_{\mathcal{V}_i} \frac{\partial \tilde{\phi}_i}{\partial t}(\mathbf{q},t) d\mathbf{q} \quad \textit{and} \quad \frac{\partial \tilde{\phi}_i}{\partial t}(\mathbf{q},t) = -2\frac{\partial \phi_i}{\partial t}(\mathbf{q},t)(\partial f(r_i)/\partial(r_i^2)). \tag{3.16}$$

In other words according to equations (3.5) to (3.7), and equation (3.16) the velocity of target should be bounded in order to bound $\dot{\mathbf{x}}_j$ and $\frac{\partial \tilde{\mathbf{c}}_i}{\partial t}$ respectively, and consequently to satisfy the third condition of Barbalat's lemma. In this work, the velocity of target has a finite value in all of the simulated scenarios.

Under these conditions, $\lim_{t\to\infty} \dot{V} = 0$ holds, and therefore by combining (3.10) and (3.12) we have that asymptotically the agents track the trajectories defined by the generalized Voronoi centroids and maximize the non-autonomous coverage metric $\mathcal{H}$.

3.3 Simulation Results

In this section, we present MATLAB simulation results that illustrate and implement various theoretical aspects discussed above. As an application, a harbor protection scenario in military domains was considered.

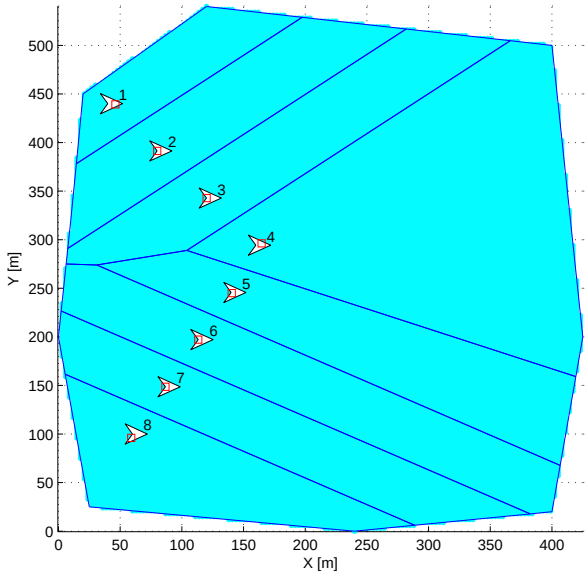


Figure 3.1: Initial configuration of agents.

3.3.1 Setup

We define a convex polygonal region with 9 vertices $(25, 25)$, $(240, 0)$, $(400, 20)$, $(425, 200)$, $(400, 500)$, $(120, 540)$, $(20, 450)$ and $(10, 320)$ as the harbour area. A priori risk is $\phi_c = 2 \times 10^{-2}, \forall \mathbf{q} \in \Omega$. In these simulations, the density function (3.3) with parameter $\sigma = 20$ was used. In a consistent system of units, eight agents ($n = 8$) are placed at initial positions $(40, 440)$, $(80, 392)$, $(120, 343)$, $(160, 295)$, $(140, 246)$, $(114, 198)$, $(87, 149)$ and $(60, 100)$ as shown in Figure 3.1. Initial Voronoi cells are calculated according to the generalized model (2.5) with strictly decreasing and differentiable sensor performance functions

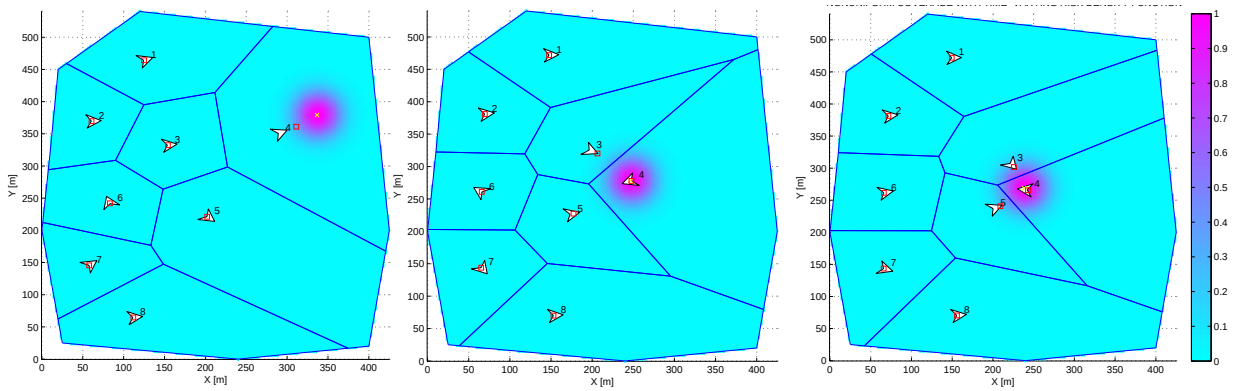
$$f(r_i) = \alpha e^{-\beta r_i^2}, \tag{3.17}$$

where $\alpha = 5$ and $\beta = 10^{-2}$. In all simulations the speed of targets are 1.5. In order to estimate the state of targets, a full- state estimator can be embeded on each agent, so that each agent is able to update its knowledge about the position and velocity of target(s) based on local measurements and on shared data. However, in this work we do not address the state estimation part, and we assume that all agents have knowledge of the state of the targets moving in the domain. Adding this additional feature is part of current and future work related to the present study. Additionally, we assume that each agent has full

knowledge of the state of the group (that is, the states of the other agents in the group) so that the computation of Voronoi cells at each time instant is standard. Generalized scenarios that involve space and time varying communication network topologies have been investigated [84] and are currently being developed. The following five scenarios are simulated:

I. Single Mobile Target:

The first scenario is single mobile target in which the target's initial and final positions are (410,450) and (4,6) respectively. Target enters the domain from one side and moves with a constant velocity on a diagonal path to the other side of the domain. Agents are supposed to chase the target and protect the domain (Figure 3.2).



(a) Time=8s (b) Time=18s (c) Time=19s

Figure 3.2: Agents configurations at different time instants for a single mobile target.

II. Two Mobile Targets:

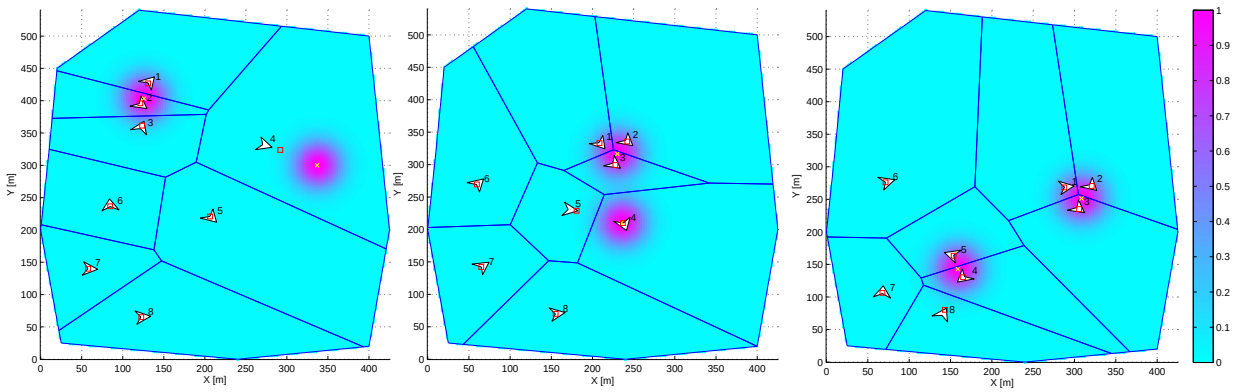
In the second scenario the initial positions of targets are (450,400) and (10,500) and their final positions are (4,6) and (250,300). The targets' enterance times are different. The agents' task is same as the previous case but here agents will be divided to two groups and each group will surround a target (Figure 3.3).

III. Two Mobile Targets (Special case):

In this scenario targets' initial positions are same as the previous case but the targets' trajectories are no longer straight lines inside the area but still agents' duty is to protect the area and surround the targets (Figure 3.4).

IV. Capturing a Mobile Target:

In the fourth scenario, the agents' task is to chase the target according to the proposed feedback law and then capture the target after a while. The target enters from the same



(a) Time=9s (b) Time=17s (c) Time=23s

Figure 3.3: Agents configurations at different time instants for two mobile targets.

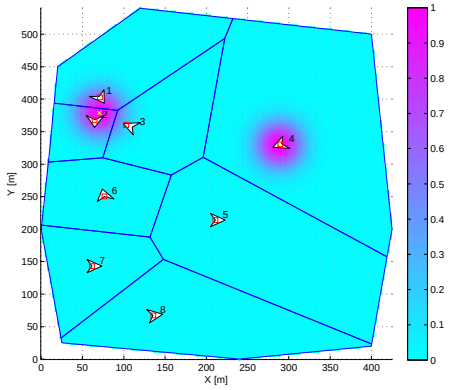
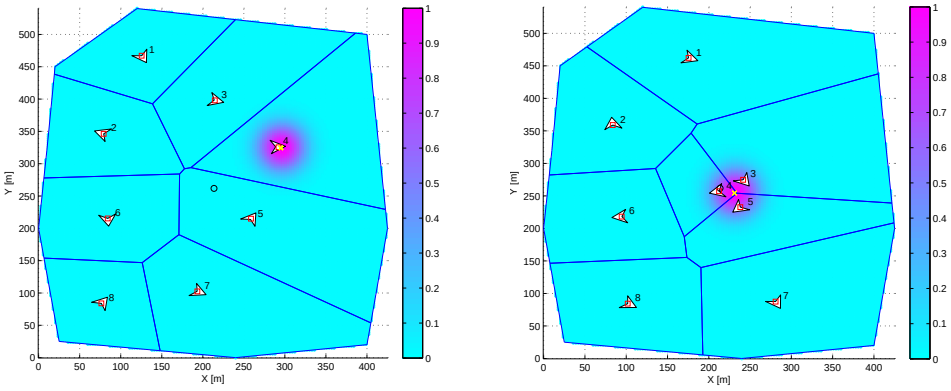


Figure 3.4: Final configuration of agents in the third scenario.

point as the first scenario but the target has a decreasing velocity with negative acceleration. In this scenario, after that the target is captured, the system will be switched to time-invariant case and the agents will follow the feedback law in equation (2.17) and they will converge to centroidal Voronoi tessellation configuration (Figure 3.5).

V. Neutralizing Three Mobile Targets:

Three targets enter the area with constant velocity from three different points at different times. Targets' initial positions are (400,-10), (350,550), and (0,250). Their final positions are (43,500), (4,6), and (407,257) respectively. The agents' duty is to protect the area, chase the targets and when an agent is 20 m far from a target, it neutralizes the target with probability one. Same as the previous case, the agents obey the feedback law (3.5) first, and then they will obey (2.17) when all targets are neutralized (Figure 3.6).



(a) Time=5s (b) Time=12s

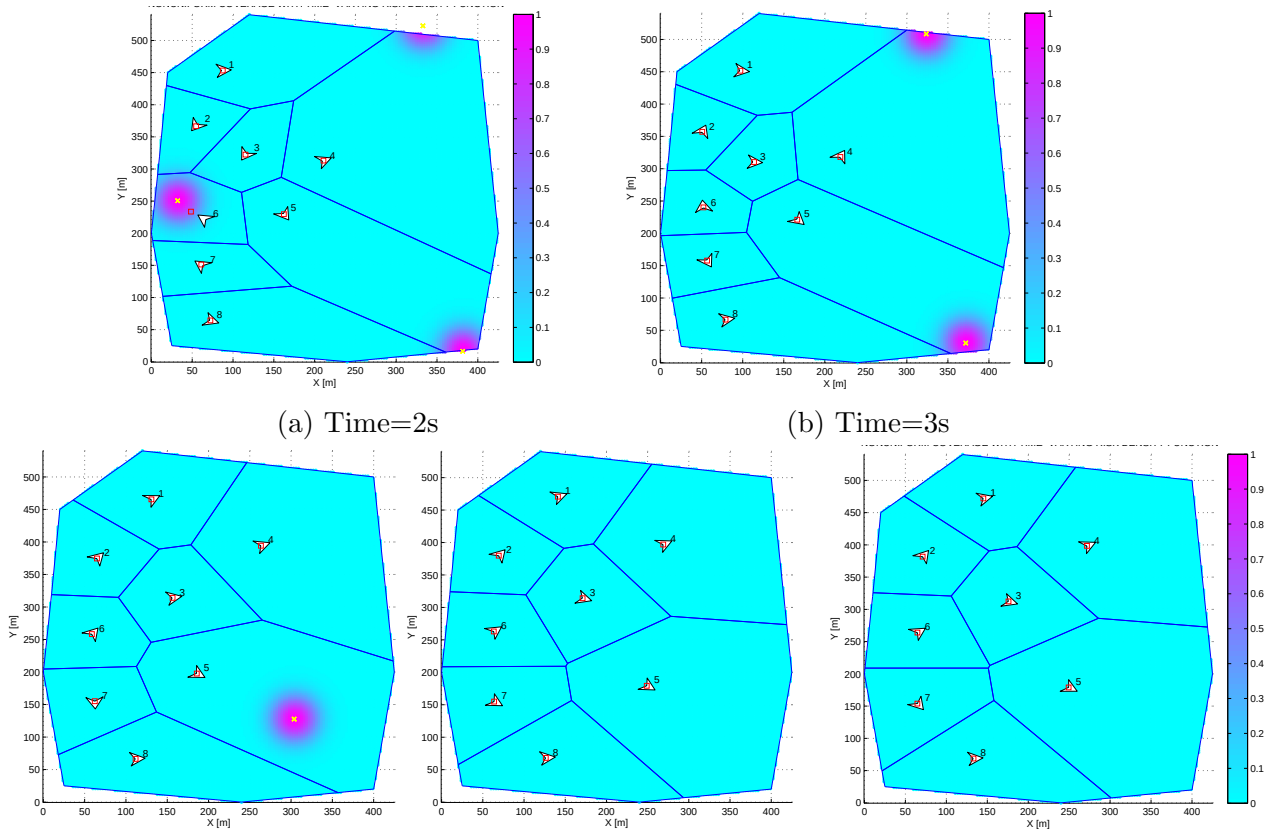
Figure 3.5: Agents configurations at different time instants for capturing a mobile target.

3.3.2 Coverage Performance

The target is represented by a purple circular region and the yellow cross shows the center of target. At each sampling time instant, $\phi = \phi_c$ and $\phi = 1$ reflects the fact that the area has lowest risk and highest risk respectively. The coverage of domain by eight agents at different time instants is shown in Figure 3.2 to Figure 3.6 where the color bar shows the relative density level with blue (purple) representing less (more) density level ($\phi = \phi_c$ for blue and $\phi = 1$ for purple). At each time the harbour's risk density is updated according to states of the targets as described by equation (3.3). The agents' motion is updated based on the kinematic model (2.15) with $\mathbf{u}(t)$ being defined in (3.5). The sampling time for the discrete-time implementation of (2.15) is chosen as $ts = 1$s so that $t = k \times ts$, for $k = 0, 1, 2, \cdots$. The non-uniform coverage by the team of eight agents subject to the time-varying risk density in the harbour at different time instants for all the scenarios are shown in Figure 3.2 to Figure 3.6. For clarity, the generalized Voronoi partitions are shown in all situations. The corresponding errors of different agents in the single mobile target scenario and two mobile target scenario are shown in the Figure 3.7 and Figure 3.8 respectively which represent the distance between each agent and its corresponding Voronoi centroid. The normalized coverage metric computed by equation (3.4) for all scenarios are shown in the Figure 3.9 to Figure 3.13.

3.4 Discussion

The spikes in Figure 3.7 and Figure 3.8 are due to the sudden approach of corresponding agents to the target at that time. In other words, after sensing the existence of targets, the shape of Voronoi cells around targets changes suddenly, so the centroids move toward the targets, resulting into a sudden increase in distance (error) between the interceptor and its corresponding centroid. In this situation, because of the feedback law (3.5) the agent starts converging to the centroid so the distance (error) between agent and its centroid will be decreased.

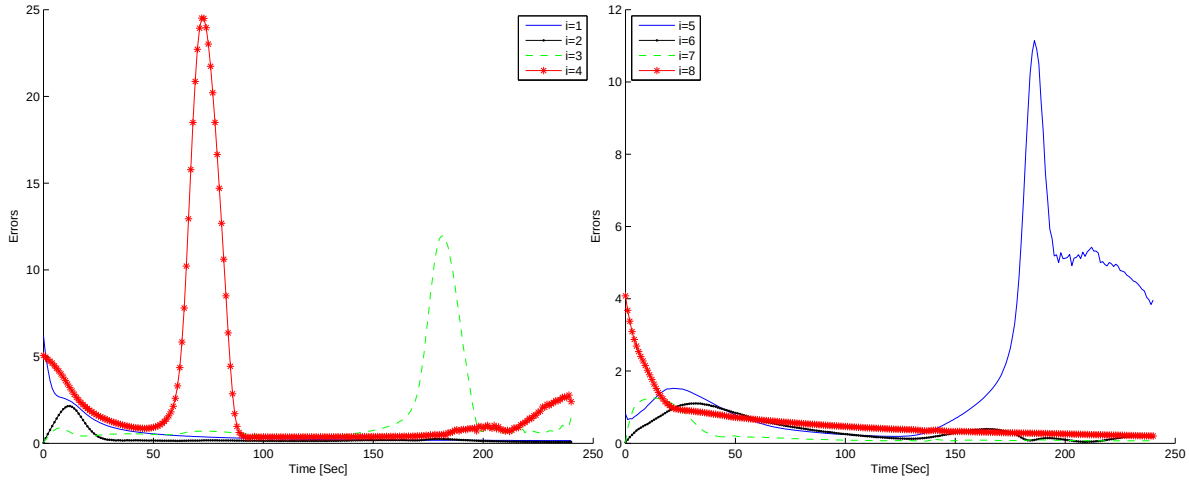


(a) Time=2s (b) Time=3s

(c) Time=10s (d) Time=14s (e) Time=16s

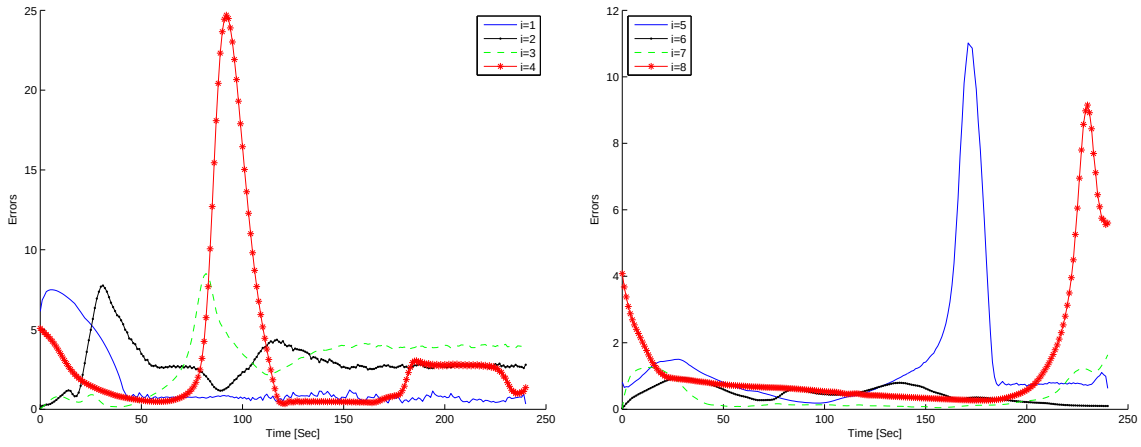
Figure 3.6: Agents configurations at different time instants for neuralizing three mobile targets.

The total coverage looks more like a staircase signal, which is expected since the density ϕ is updated according to the target's position. Nevertheless, if the density remains constant for a sufficient period of time, all the agents are guaranteed to configure themselves optimally and the coverage is maximized as desired. The spikes in Figure 3.13 are related to the time instants at which each target is being neutralized, where at that moment the agent reaches the centroid and we can observe a sudden increase in coverage but after target neutralization, a sudden decrease will be seen because of decrease in risk density at that moment. Figure 3.9 to Figure 3.13 show that in all scenarios the coverage metric is maximized and the Lyapunov function is minimized at the end of simulation time, hence, the agents are located at optimal configurations inside the domain Ω.



(a) Agents 1 to 4 (b) Agents 5 to 8

Figure 3.7: Errors (single mobile target).



(a) Agents 1 to 4 (b) Agents 5 to 8

Figure 3.8: Errors (Two mobile targets).

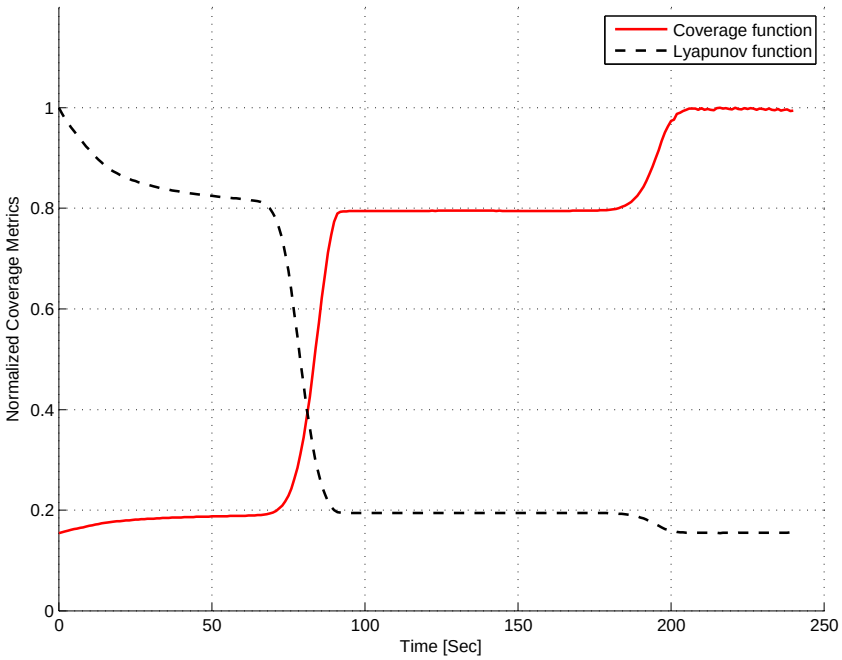


Figure 3.9: Single mobile target.

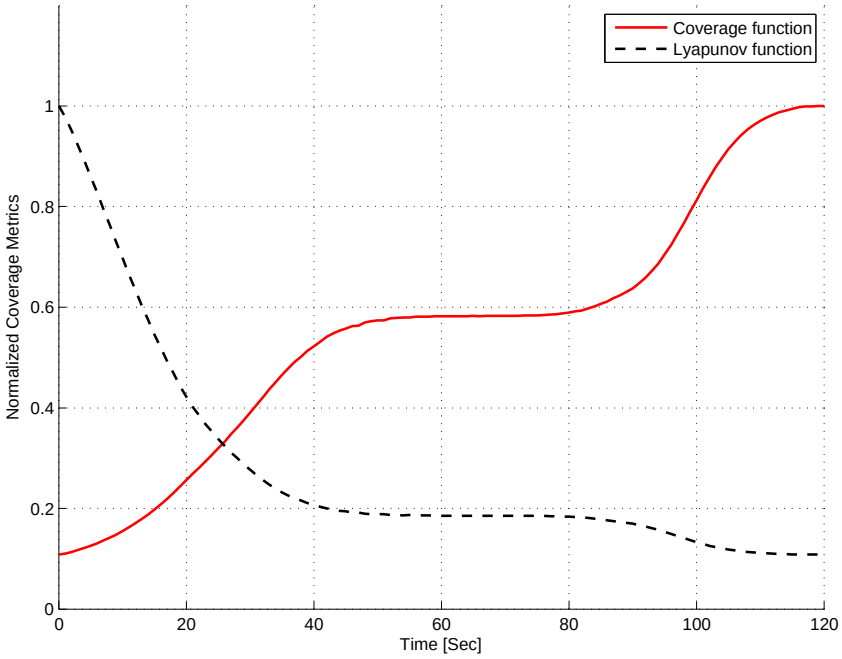


Figure 3.10: Two mobile targets.

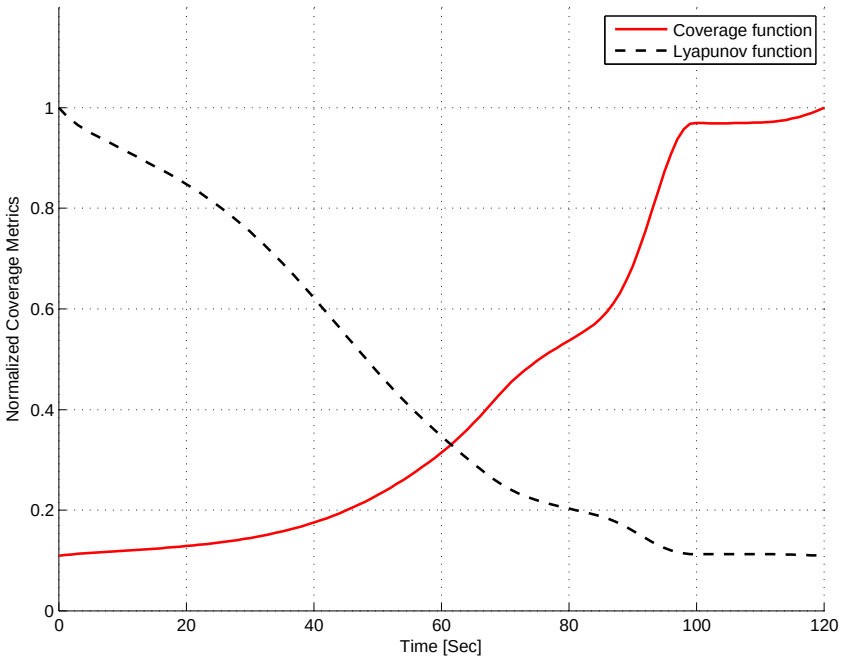


Figure 3.11: Two mobile targets (Special case).

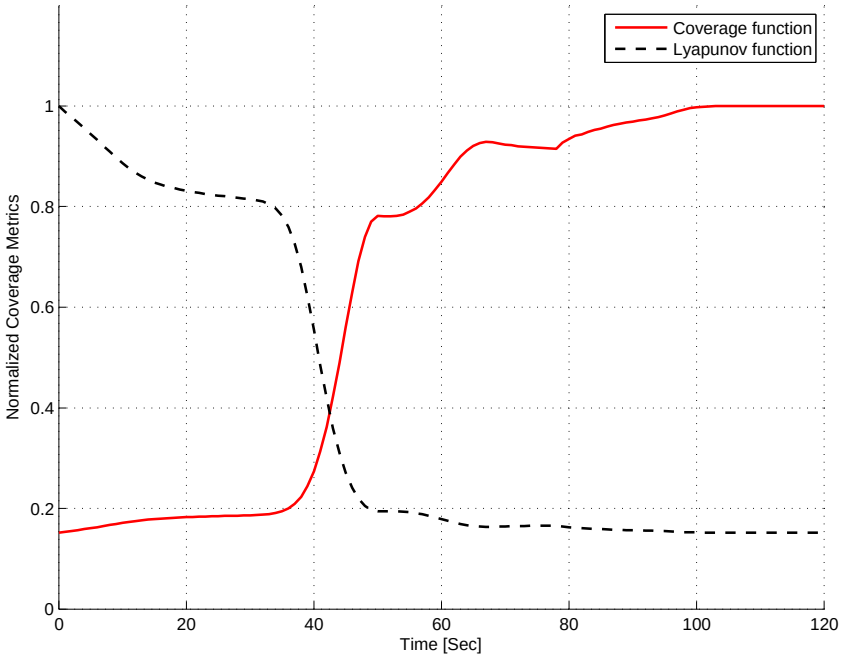


Figure 3.12: Capturing single mobile target.

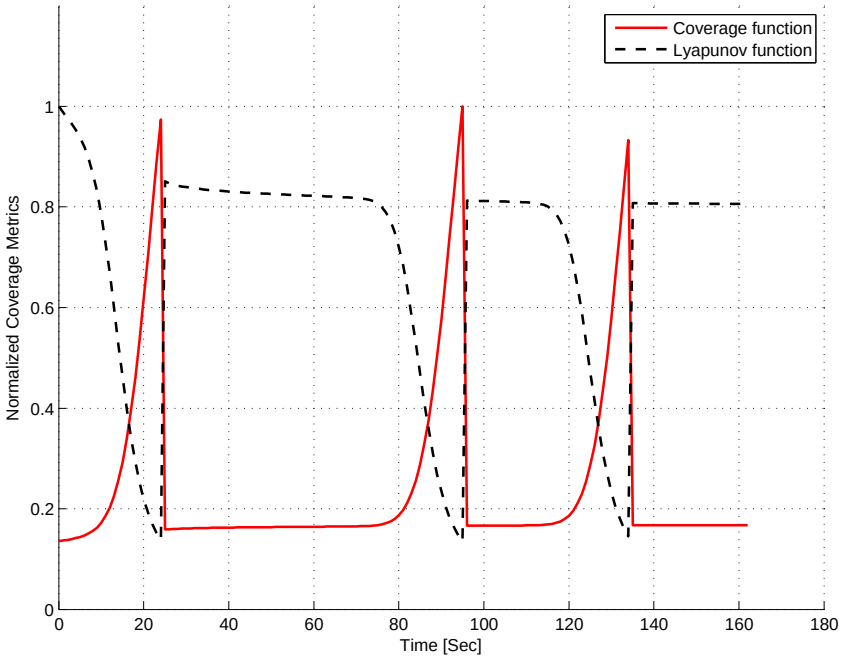


Figure 3.13: Neutralizing three mobile targets.

Chapter 4

Summary and Conclusion

Technological developments in fields such as autonomous systems technology, wireless communication, and sensors, allows to use networked multi-agent systems to cooperatively accomplish several missions in a distributed manner. Wireless sensor network is an important application of multi-robot systems, which a group of autonomous robots can cooperatively sense events of interest occuring in a domain. This thesis work has been dedicated to the study of a multi-robot system, where the states of robots are updated based on kinematic (velocity) feedback inputs. In this work a generalized area control problem was considered in which the coverage metric is non-autonomous, that is time varying independently of the states of the robots. This generalization was motivated by the study of coverage control problems in which the coordinated motion of a set of mobile robots accounts for the kinematics of objects penetrating from the outside. Barbalat's Lemma was used to study optimality and asymptotic convergence of the non-autonmous system, and connections with the kinematics of the moving intruders was established.

Several functions could be used to model risk density. The total risk density function in this work consists of two terms, in which the first one is the biased density that models a priori risk distribution in the environment, established to quantify the relative importance assigned to different subregions, and the second one is the risk density related to external threats (targets) intruding in the domain. Targets inside the domain introduce time varying components to the total risk density function, that in turn characterize the coverage metric as non-autonomous, since there is a time dependence not implicitly related to the motion of the agents. A suitable feedback law is introduced to account for the non-autonomous character of the coverage metric. By using the framework of Barbalat's lemma, it is shown that such feedback law generates optimal trajectories in the sense that they asymptotically maximize the coverage metric. Optimal trajectories have a component that attracts the agents towards the centroids of the corresponding Voronoi cells, as in the classic Lloyd's algorithm that is derived to solve coverage problems with autonomous metrics, and a component that attracts the agents towards external threats. Therefore this feedback law is suitable for tactical situations in which cooperating autonomous agents actively engage external threats.

Different scenarios were simulated in order to numerically illustrate theoretical pre-

dictions. In two of scenarios, we consider different mobile targets entering the domain and moving with constant velocities. The proposed feedback law generated interceptors' trajectories that resulted into convergence to Voronoi centroids with higher density of interceptors around targets, as a consequence of non-uniformity of the risk function. The coverage metric was asymptotically maximized. We have also simulated two scenarios in which targets move inside the domain with constant acceleration, and are respectively captured and neutralized by the interceptors, mimicking typical tactical situations in which interceptors engage and neutralize threats entering protected perimeters. After the disappearance of the targets, the motion control law reduces to the classic Lloyd's algorithm and therefore interceptor converge to Voronoi centroids that, in the absence of targets, are asymptotically time invariant.

The underlying assumption of this work is that agents have always the most updated knowledge of the state of the other agents in the group, so that Voronoi tessellations can be computed at every updating time. Moreover, it is assumed that inter-agent communications are noiseless, and that each agent knows the state of targets in the domain. The relaxation of these assumptions is part of current and future work, that include the study of the asymptotic properties of networked systems with time varying, stochastic communication network topologies, with neighbor rules that introduce space or general metric dependencies among communicating agents. Moreover, state estimators based on noisy measurements of targets' states have to be embedded on mobile agents to investigate the coupling with the proposed motion control feedback laws.

APPENDICES

Appendix A

Matlab Code

A.1 Main Code

```
close all
clear all
clc

bounds = [25, 240, 400, 425, 400, 120, 20, 10, 0;...
25, 0, 20, 200, 500, 540, 450, 320, 200]; %[m]

n = 8;  % number of interceptors

%%%%%%%% Divide the area of interest into a set of discrete point set
x = min(bounds(1,:)):3:max(bounds(1,:)); % x axis range
y = min(bounds(2,:)):3:max(bounds(2,:)); % y axis range
pts = zeros(length(x)*length(y),2); % total no. of possible discrete points
for i=1:size(x,2)
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
in = inpolygon(pts(:,1),pts(:,2),[(bounds(1,:))'; bounds(1,1)],...
[(bounds(2,:))'; bounds(2,1)]);
pts_harbour = [pts(in,1) pts(in,2)]; % discrete points in the harbour
npts = max(size(pts_harbour)); % number of points in the harbour area
%%%interceptors initial position%%%%%
yy = linspace(440,100,n);
xx = [linspace(40,160,floor(n/2)), linspace(140,60,ceil(n/2))];
intcpt_pos= [xx;yy];
intcpt_valcity =zeros(2,n);
```

```
init_state = [intcpt_pos;intcpt_valcity];
%%%%%%%%%%  Risk zone distribution   %%%%%%%%%%%%%%%%%%%%%
aviobj = avifile...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\SingleTNewFeedLaw.avi'...
,'compression','None','fps',10);
fig=figure;

alpha=5;
beta=0.001;

sigma = 20;

%1 Knot= 0.514 m/s

%risk zone centers
const = 1/50;  % constant
tf = 4*60; % total simulation time [sec]
ts = 1; % sampling time of interceptors [sec]
tin =  0:ts:tf; % time span
directions = zeros(1,n);
vel=2.5*0.514;
cxi = 410;
cyi = 450;
cxf=4;
cyf=6;
vx=vel*cos(atan2((cyf-cyi),(cxf-cxi)));
vy=vel*sin(atan2((cyf-cyi),(cxf-cxi)));
pointsize = 14;
Grad_H_Tdot_total=1;
% initialize some matric for state-space equations
C = [1 0 0 0;0 1 0 0];
Q1 = diag([1,1,1,1]);
Q2 = diag([1,1]);
C_bar = eye(4)-C'*((C*C')\C);
Q = C_bar'*Q1*C_bar + C'*Q2*C;

for time = 1: size(tin,2)
cx=cxi+vx*(tin(time));
cy=cyi+vy*(tin(time));
target=[cx cy];
for i=1:npts
temp1 = (pts_harbour(i,1) - cx)^2/(2*sigma^2)+...
(pts_harbour(i,2)-cy)^2/(2*(sigma)^2);
temp1_derriv=vx*(pts_harbour(i,1) - cx)/(sigma^2)+...
vy*(pts_harbour(i,2)-cy)/((sigma+0.15)^2);
```

```
rho(i,time)= const+ exp(-temp1);
rho2(i,time)=exp(-temp1)*temp1_derriv;
end
% Interceptor's system matrices
Phik= [1 0 ts 0;...
0 1 0 ts;...
0 0 1 0;...
0 0 0 1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf
scatter(pts_harbour(:,1),pts_harbour(:,2), pointsize, rho(:,time) ,...
'fill')
set(gca,'Color',[0.8 0.8 0.8]); % figure's background color
colormap(cool(64));
caxis([0 1]);
colorbar;
regions = voronoi_andrew(bounds, intcpt_pos);
drawRegions(bounds, regions, directions);
plot(target(1),target(2),'yx');
for i=1:n
pt = regions{i,1}; % interceptor position
tmp = regions{i,3};
vertices = tmp(1:end,1:2); % [(x1,y1),(x1,y1),...., (xm,ym)]'
%%%%%%%%%%%   find centroid of i th voronoi polygon  %%%%%%%%%
in_Vi = inpolygon(pts_harbour(:,1),pts_harbour(:,2),...
[vertices(:,1);vertices(1,1)],[vertices(:,2);vertices(1,2)]);
pts_Vi = [pts_harbour(in_Vi,1) pts_harbour(in_Vi,2)];
rho_Vi = rho(in_Vi,time);
rho2_Vi=rho2(in_Vi,time);
dis2=[(pts_Vi(:,1)-pt(1)).^2+(pts_Vi(:,2)-pt(2)).^2];
Phi_tilde= (2*alpha*beta*(exp(-beta.*dis2))).*(rho_Vi);
fr= alpha*(exp(-beta.*dis2));
Grad_H_Tdot(i)= sum(fr.*rho2_Vi);
% compute centroid
m_phitilde = sum(Phi_tilde); % sum of risk (density) of polygon Vi
x_Phi_tilde = (1/m_phitilde)*sum(pts_Vi(:,1).*Phi_tilde);
y_Phi_tilde = (1/m_phitilde)*sum(pts_Vi(:,2).*Phi_tilde);
c_phitilde = [x_Phi_tilde y_Phi_tilde];
plot(c_phitilde(1),c_phitilde(2),'rs');
%compute direction and error
errx= c_phitilde(1) - pt(1);
erry= c_phitilde(2) - pt(2);
Err(i,time) = sqrt(errx^2+erry^2);
directions(i) = atan2(erry,errx);
```

```
%Feedback law
x_tilde = C'*((C*C')\(c_phitilde'));
Gamax=(-Grad_H_Tdot(i)*errx)/(m_phitilde*((errx)^2+(erry)^2));
Gamay=(-Grad_H_Tdot(i)*erry)/(m_phitilde*((errx)^2+(erry)^2));
Gama=[Gamax,Gamay];
Gama_tilde=C'*((C*C')\(Gama'));
v_max=6*0.514;
u = (Q*(x_tilde-init_state(:,i)))+Gama_tilde;

init_state(:,i) = Phik*init_state(:,i) + ts*u;
intcpt_pos(:,i) = C*init_state(:,i);

b(i)=sum(fr.*rho_Vi);
end
H(time)=sum(b);
Error(time)=sum(Err(:,time));

%%%%%%%%%%%%%%%%%%%% Snapshots of Frames %%%%%%%%%%%%%%%%%%%%%%%%%
if time==1
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\Initial');
saveas(gcf, savefilename, 'pdf');
end
if time==75
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\8s');
saveas(gcf, savefilename, 'pdf');
end
if time==180
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\18s');
saveas(gcf, savefilename, 'pdf');
end
if time==190
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\19s');
saveas(gcf, savefilename, 'pdf');
end

F = getframe(fig);
aviobj = addframe(aviobj,F);

drawRegions(bounds, regions, directions);
end
```

```
aviobj=close(aviobj);
close(fig);

%%%%%%%%%%%%%%%%%%%%%%%%% Plots%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
hold on;
plot(tin,smooth(Err(1,:)));
plot(tin,smooth(Err(2,:)),'k.-');
plot(tin,smooth(Err(3,:)),'g--');
plot(tin,smooth(Err(4,:)),'-r*');
hold off;

h=legend('i=1','i=2','i=3','i=4',1);
set(h,'interpreter','none');
xlabel('Time [Sec]');
ylabel('Errors');
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\Error1-4');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');

figure;
hold on;
plot(tin,smooth(Err(5,:)));
plot(tin,smooth(Err(6,:)),'k.-');
plot(tin,smooth(Err(7,:)),'g--');
plot(tin,smooth(Err(8,:)),'-r*');
hold off;
h=legend('i=5','i=6','i=7','i=8',2);
set(h,'interpreter','none');
xlabel('Time [Sec]');
ylabel('Errors');
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\Error5-8');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');

figure;
title('H');
plot(tin,H);
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\CoverageH');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
```

```
figure;
title('error');
plot(tin,Error);
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\Error');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');

figure;
Error_new=Error/max(Error);
H_new=H/max(H);
plot(tin,H_new,'-ro',tin,smooth(Error_new),'-.b');
y=legend('Total Coverage, H', 'Total Error',2);
xlabel('Time [sec]');
ylabel('Normalized Metrics');
ylim([0, 1.05]);
set(y,'interpreter', 'none');
savefilename=...
('C:\Users\Arian\Documents\T\MATLAB\Code\SingleTNewFeedLaw\Error&H');
saveas(gcf, savefilename, 'pdf');
saveas(gcf, savefilename, 'fig');
```

A.2 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).
%% each point has three different matrices.
function regionData = voronoi_andrew(bounds, points)
n = size(points, 2); %n_p
regionData = cell(n, 3); %n_p
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.3 Geometric Intersection of a Polygon and Halfplane

```
***** 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 positively 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); % n is the number of vertices
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.4 Voronoi Graphical Realization

```
   This function will take in a set of generator points and regions and
   will draw the associated graph

   ***** Inputs *****
   bounds - the bounding positively oriented polygon
   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}
       containing edge segments (from intersections with the boundary)

function drawRegions(bounds, regionData, orientations)
%clf
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_DDMR([pt;orientations(i)],axis(gca));
 % DDMR => Differential drive mobile robot
fill(Xa,Ya,'w');
%     lblintcpt = plot(pt(1), pt(2), 'r.', 'MarkerSize', 16);
% Label the agent
label = strcat(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(10, 520, 'Color', 'blue');
 axis equal
axis square
xlim([min(bounds(1,:)-0.5),max(bounds(1,:)+0.5)]);
ylim([min(bounds(2,:)-0.5),max(bounds(2,:)+0.5)]);
xlabel('X [m]');
ylabel('Y [m]');
title('NONUNIFORM COVERAGE WITH TIME-VARYING RISK DENSITY FUNCTION')

 legend([lblintcpt lblbounds], 'Interceptor', 'harbour boundary');
grid on
axis off
hold off
```

A.5 Drawing The Agent's Shape

```
function [X,Y] = plot_DDMR(Q,AX);
%Function that generates lines for plotting a unicycle.
%
%    PLOT_UNICYCLE(Q,AX) takes in the axis AX = axis(gca) and the unicycle
%    configuration Q and outputs lines (X,Y) for use, for example, as:
%       fill(X,Y,'b')
%    This must be done in the parent script file.

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-2*pi/3),x+l1*cos(theta),x+l1*cos(theta+2*pi/3),x];
Y = [y,y+l1*sin(theta-2*pi/3),y+l1*sin(theta),y+l1*sin(theta+2*pi/3),y];
```

References

[1] Y. Gu, "Introduction to robot planning," *Lecture note, West Virginia University.*

[2] D. Specht, "Gis, big data and lessons from john snow," 2015.

[3] R. B. Muhammad, "Incremental method," *Lecture note, Kent State University.*

[4] R. B. Muhammad, "Divide-and-conquer paradigm," *Lecture note, Kent State University.*

[5] D. Coeurjolly, "Computational geometry: Digital delaunay triangulation and application," *Laboratoire d'InfoRmatique en Image et Systmes d'information.*

[6] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *Automatic Control, IEEE Transactions on*, vol. 53, no. 11, pp. 2480–2496, 2008.

[7] A. Kansal, W. Kaiser, G. Pottie, M. Srivastava, and G. Sukhatme, "Reconfiguration methods for mobile sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 4, p. 22, 2007.

[8] K. Laventall and J. Cortés, "Coverage control by multi-robot networks with limited-range anisotropic sensory," *International Journal of Control*, vol. 82, no. 6, pp. 1113–1121, 2009.

[9] S. Susca, F. Bullo, and S. Martínez, "Monitoring environmental boundaries with a robotic sensor network," *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 288–296, 2008.

[10] Y. Cao and W. Ren, "Multi-vehicle coordination for double-integrator dynamics under fixed undirected/directed interaction in a sampled-data setting," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 9, pp. 987–1000, 2010.

[11] Y. Zou and P. R. Pagilla, "Distributed constraint force approach for coordination of multiple mobile robots," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1-2, pp. 5–21, 2009.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[13] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[14] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 427–438, 2013.

[15] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[16] J. Cortés and F. Bullo, "Coordination and geometric optimization via distributed dynamical systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, 2005.

[17] Q. Du and M. Emelianenko, "Acceleration schemes for computing centroidal voronoi tessellations," *Numerical linear algebra with applications*, vol. 13, no. 2-3, pp. 173–192, 2006.

[18] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the lloyd algorithm for computing centroidal voronoi tessellations," *SIAM journal on numerical analysis*, vol. 44, no. 1, pp. 102–119, 2006.

[19] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On centroidal voronoi tessellationenergy smoothness and fast computation," *ACM Transactions on Graphics (ToG)*, vol. 28, no. 4, p. 101, 2009.

[20] S. N. Simić and S. Sastry, "Distributed environmental monitoring using random sensor networks," in *Information Processing in Sensor Networks*, pp. 582–592, Springer, 2003.

[21] R. Carli and F. Bullo, "Quantized coordination algorithms for rendezvous and deployment," *SIAM Journal on Control and Optimization*, vol. 48, no. 3, pp. 1251–1274, 2009.

[22] M. E. Liggins, C.-Y. Chong, I. Kadar, M. G. Alford, V. Vannicola, S. Thomopoulos, *et al.*, "Distributed fusion architectures and algorithms for target tracking," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 95–107, 1997.

[23] T. H. Chung, V. Gupta, J. W. Burdick, and R. 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*, vol. 2, pp. 1914–1919, IEEE, 2004.

[24] S. MartíNez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[25] L. Ma and N. Hovakimyan, "Vision-based cyclic pursuit for cooperative target tracking," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 617–622, 2013.

[26] A. S. Willsky, M. Bello, D. A. Castanon, B. C. Levy, and G. C. Verghese, "Combining and updating of local estimates and regional maps along sets of one-dimensional tracks," *Automatic Control, IEEE Transactions on*, vol. 27, no. 4, pp. 799–813, 1982.

[27] B. Rao, H. Durrant-Whyte, and J. Sheen, "A fully decentralized multi-sensor system for tracking and surveillance," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 20–44, 1993.

[28] Y.-C. Chen and C.-Y. Wen, "Decentralized cooperative toa/aoa target tracking for hierarchical wireless sensor networks," *Sensors*, vol. 12, no. 11, pp. 15308–15337, 2012.

[29] Z. Wang and D. Gu, "Cooperative target tracking control of multiple robots," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 8, pp. 3232–3240, 2012.

[30] C. Zhang and S. Fei, "Energy efficient target tracking algorithm using cooperative sensors," *Systems Engineering and Electronics, Journal of*, vol. 23, no. 5, pp. 640–648, 2012.

[31] A. Mahapatra, K. Anand, and D. P. Agrawal, "Qos and energy aware routing for real-time traffic in wireless sensor networks," *Computer Communications*, vol. 29, no. 4, pp. 437–445, 2006.

[32] N. Chenglianq, L. Donqxin, Z. Tingxian, and L. Lihonq, "Distributed power control algorithm based on game theory for wireless sensor networks," *Systems Engineering and Electronics, Journal of*, vol. 18, no. 3, pp. 622–627, 2007.

[33] W.-L. Yeow, C.-K. Tham, and W.-C. Wong, "Energy efficient multiple target tracking in wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 2, pp. 918–928, 2007.

[34] A. Boukerche, H. A. Oliveira, E. F. Nakamura, and A. A. Loureiro, "Vehicular ad hoc networks: A new challenge for localization-based systems," *Computer communications*, vol. 31, no. 12, pp. 2838–2849, 2008.

[35] H. Lu, S. Zhang, X. Liu, and X. Lin, "Vehicle tracking using particle filter in wi-fi network," in *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pp. 1–5, IEEE, 2010.

[36] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *Information Fusion, 2008 11th International Conference on*, pp. 1–6, IEEE, 2008.

[37] S. Yousefi, M. S. Mousavi, and M. Fathy, "Vehicular ad hoc networks (vanets): challenges and perspectives," in *ITS Telecommunications Proceedings, 2006 6th International Conference on*, pp. 761–766, IEEE, 2006.

[38] H. S. Ramos, A. Boukerche, R. W. Pazzi, A. C. Frery, and A. A. Loureiro, "Cooperative target tracking in vehicular sensor networks," *Wireless Communications, IEEE*, vol. 19, no. 5, pp. 66–73, 2012.

[39] A. Suzuki and Z. Drezner, "The p-center location problem in an area," *Location science*, vol. 4, no. 1, pp. 69–82, 1996.

[40] R. Chandrasekaran and A. Daughety, "Location on tree networks: p-centre and n-dispersion problems," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 50–57, 1981.

[41] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *Automatic Control, IEEE Transactions on*, vol. 49, no. 8, pp. 1292–1302, 2004.

[42] R. Bachmayer and N. E. Leonard, "Vehicle networks for gradient descent in a sampled environment," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 1, pp. 112–117, IEEE, 2002.

[43] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed autonomous robotic systems 5*, pp. 299–308, Springer, 2002.

[44] T. Balch and M. Hybinette, "Behavior-based coordination of large-scale robot formations," in *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pp. 363–364, IEEE, 2000.

[45] R. Arkin, "ehavior-based robotics," 1998.

[46] J. Cortes, S. Martinez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 11, no. 04, pp. 691–719, 2005.

[47] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001.

[48] M. Porfiri, D. G. Roberson, and D. J. Stilwell, "Tracking and formation control of multiple autonomous agents: A two-level consensus approach," *Automatica*, vol. 43, no. 8, pp. 1318–1328, 2007.

[49] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 3500–3505, IEEE, 2005.

[50] R. M. Murray, "Recent research in cooperative control of multivehicle systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 2007.

[51] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, 2006.

[52] L. C. Pimenta, G. A. Pereira, M. M. Gonçalves, N. Michael, M. Turpin, and V. Kumar, "Decentralized controllers for perimeter surveillance with teams of aerial robots," *Advanced Robotics*, vol. 27, no. 9, pp. 697–709, 2013.

[53] G. Zhang, G. K. Fricke, and D. P. Garg, "Spill detection and perimeter surveillance via distributed swarming agents," *Mechatronics, IEEE/ASME Transactions on*, vol. 18, no. 1, pp. 121–129, 2013.

[54] M. K. Allouche and A. Boukhtouta, "Multi-agent coordination by temporal plan fusion: Application to combat search and rescue," *Information Fusion*, vol. 11, no. 3, pp. 220–232, 2010.

[55] Z. Kitowski, "Architecture of the control system of an unmanned surface vehicle in the process of harbour protection," *Solid State Phenomena*, vol. 180, pp. 20–26, 2012.

[56] Z. Kitowski, "Optoelectronic systems on board of unmanned surface vehicle'edredon'," in *Solid State Phenomena*, vol. 196, pp. 198–205, Trans Tech Publ, 2013.

[57] J. Veers and V. Bertram, "Development of the usv multi-mission surface vehicle iii," in *5th International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2006.

[58] U. Navy, "The navy unmanned surface vehicle (usv) master plan," *URL:¡ http://www. navy. mil/navydata/technology/usvmppr. pdf*, 2007.

[59] R. A. Freeman, P. Yang, and K. M. Lynch, "Distributed estimation and control of swarm formation statistics," in *American control conference*, vol. 7, Citeseer, 2006.

[60] Q. Du, M. Gunzburger, L. Ju, and X. Wang, "Centroidal voronoi tessellation algorithms for image compression and segmentation," 2006.

[61] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*, vol. 501. John Wiley & Sons, 2009.

[62] A. Hameurlain, J. Küng, and R. Wagner, eds., *Transactions on Large-Scale Data- and Knowledge-Centered Systems I*, vol. 5740 of *Lecture Notes in Computer Science*, Springer, 2009.

[63] M. Bock, A. K. Tyagi, J.-U. Kreft, and W. Alt, "Generalized voronoi tessellation as a model of two-dimensional cell tissue dynamics," *Bulletin of mathematical biology*, vol. 72, no. 7, pp. 1696–1731, 2010.

[64] H. Li, K. Li, T. Kim, A. Zhang, and M. Ramanathan, "Spatial modeling of bone microarchitecture," in *IS&T/SPIE Electronic Imaging*, pp. 82900P–82900P, International Society for Optics and Photonics, 2012.

[65] A. Sud, N. Govindaraju, and D. Manocha, "Interactive computation of discrete generalized voronoi diagrams using range culling," in *Proc. International Symposium on Voronoi Diagrams in Science and Engineering*, 2005.

[66] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1-4, pp. 153–174, 1987.

[67] P. J. Green and R. Sibson, "Computing dirichlet tessellations in the plane," *The Computer Journal*, vol. 21, no. 2, pp. 168–173, 1978.

[68] T. Ohya, M. Iri, and K. Murota, "Improvements of the incremental method for the voronoi diagram with computational comparison of various algorithms.," *J. OPER. RES. SOC. JAPAN.*, vol. 27, no. 4, pp. 306–336, 1984.

[69] K. Sugihara and M. Iri, "Construction of the voronoi diagram for one million generators in single-precision arithmetic," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1471–1484, 1992.

[70] M. I. Shamos and D. Hoey, "Closest-point problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pp. 151–162, IEEE, 1975.

[71] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

[72] M. J. Sabin and R. M. Gray, "Global convergence and empirical consistency of the generalized lloyd algorithm," *Information Theory, IEEE Transactions on*, vol. 32, no. 2, pp. 148–155, 1986.

[73] M. Emelianenko, L. Ju, and A. Rand, "Nondegeneracy and weak global convergence of the lloyd algorithm in r^d," *SIAM Journal on Numerical Analysis*, vol. 46, no. 3, pp. 1423–1441, 2008.

[74] X. Xiao, "Over-relaxation lloyd method for computing centroidal voronoi tessellations," 2010.

[75] O. Deussen, S. Hiller, C. Van Overveld, and T. Strothotte, "Floating points: A method for computing stipple drawings," in *Computer Graphics Forum*, vol. 19, pp. 41–50, Wiley Online Library, 2000.

[76] Q. Du and M. Gunzburger, "Grid generation and optimization based on centroidal voronoi tessellations," *Applied Mathematics and Computation*, vol. 133, no. 2, pp. 591–607, 2002.

[77] A. Secord, "Weighted voronoi stippling," in *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '02, (New York, NY, USA), pp. 37–43, ACM, 2002.

[78] A. Hausner, "Simulating decorative mosaics," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 573–580, ACM, 2001.

[79] M. Dickerson, D. Eppstein, and K. A. Wortman, "Planar voronoi diagrams for sums of convex functions, smoothed distance and dilation," in *Voronoi Diagrams in Science and Engineering (ISVD), 2010 International Symposium on*, pp. 13–22, IEEE, 2010.

[80] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2, pp. 1327–1332, IEEE, 2002.

[81] S. G. Lee and M. Egerstedt, "Controlled coverage using time-varying density functions," in *Estimation and Control of Networked Systems*, vol. 4, pp. 220–226, 2013.

[82] L. C. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. Pereira, "Simultaneous coverage and tracking (scat) of moving targets with robot networks," in *Algorithmic Foundation of Robotics VIII*, pp. 85–99, Springer, 2009.

[83] F. Lekien and N. E. Leonard, "Nonuniform coverage and cartograms," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, pp. 5518–5523, IEEE, 2010.

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