Cooperative Multi Agent Search and Coverage in Uncertain Environments

Mostafa Mirzaei

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfilment of the Requirements

For the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

April 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mostafa Mirzaei**

Entitled: **Cooperative Multi Agent Search and Coverage in Uncertain Environments**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Mechanical Engineering)

complies with the regulations of the University and meet the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Adel Hanna

_____ External Examiner
Dr. Scott Nokleby

_____ External to Program
Dr. Krzysztof Skonieczny

_____ Examiner
Dr. Chun-Yi Su

_____ Examiner
Dr. Wen Fang Xie

_____ Supervisor
Dr. Brandon W. Gordon

_____ Supervisor
Dr. Camille-Allain Rabbath

Approved by

_____
Chair of Department or Graduate Program Director

April 2015 _____
Dean of Faculty of Engineering and Computer Science

Abstract

Cooperative Multi Agent Search and Coverage in Uncertain Environments

Mostafa Mirzaei, Ph.D.

Concordia University, 2015

In this dissertation, the cooperative multi agent search and coverage problem in uncertain environments is investigated. Each agent individually plans its desired trajectory. The agents exchange their positions and their sensors' measurement with their neighboring agents through a communication channel in order to maintain the cooperation objective.

Different aspects of multi agent search and coverage problem are investigated. Several models for uncertain environments are proposed and the updating rules for the probability maps are provided. Each of this models is appropriate for a specific type of problems. The cooperative search mission is first converted to a decentralized multi agent optimal path planning problem, using rolling horizon dynamic programing approach which is a mid-level controller. To make cooperation between agents possible, two approximation methods are proposed to modify the objective function of agents and to take into the account the decision of other agents. The simulation results show the proposed methods can considerably increase the performance of mission without significantly increasing the computation burden. This approach is then extended for the case with known communication delay between mobile agents. The simulation results show the proposed methods can compensate for the effect of known communication delay between mobile agents. A Voronoi-based search strategy for a team of mobile agents with limited range sensors is also proposed which combines both mid-level and low-level controllers. The strategy includes the short-term objective of maximizing the uncertainty reduction in the next step, the long-term objective of distributing the agents in the environment with minimum overlap in their sensory

domain, and the collision avoidance constraint. The simulation results show the proposed control law can reduce the value of uncertainty in the environment below any desired threshold.

For the search and coverage problem, we first introduce a framework that includes two types of agents; search agents and coverage agents. The problem is formulated such that the information about the position of the targets is updated by the search agents. The coverage agents use this information to concentrate around the more important areas in the environment. The proposed cooperative search method, along with a well-known Centroidal Voronoi Configuration method for coverage, is used to solve the problem. The effectiveness of the proposed algorithm is demonstrated by simulation and experiment. We then introduce the "limited turn rate Voronoi diagram" and formulate the search and coverage problem as a multi-objective optimization problem with different constraints which is able to consider practical issues like minimum fuel consumption, refueling, obstacle avoidance, and collision avoidance. In this approach, there is only one type of agents which performs both search and coverage tasks. The "multi agent search and coverage problem" is formulated such that the "multi agent search problem" and "multi agent coverage problem" are special cases of this problem. The simulation results show the effectiveness of the proposed method.

Acknowledgments

First of all, I would like to express my deep appreciation and gratitude to my supervisor, Dr. Brandon W. Gordon for his guidance, encouragement and advice. My research would not have been possible without his help. I would also like to thank my co-supervisor, Dr. Camille. A. Rabbath for his useful comments which have contributed greatly to the improvement of the thesis.

I would like to acknowledge Dr. Youmin Zhang for letting me use his laboratory facilities for the experimental test and for his constant assistance and advice. I would like to thank my colleagues at Concordia University, especially Dr. Farid Sharifi, Dr. Abbas Chamseddine, and Dr. Hojjat Izadi. I would also like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for providing financial assistance for this research.

Last but not least, I would like to thank my loving and caring family. My deepest gratitude goes to my beloved parents for their endless love, prayers and encouragement. I would also like to express my appreciation to my brother and sister who were always supporting and encouraging me with their best wishes. Words cannot express my appreciation and love for my wife. I would like to thank her for faithful support, patience, unconditional love and care. My world would not be the same without her.

Mostafa Mirzaei
April 2015

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The problem of cooperative multi agent decision making and control is to deploy a group of agents over an environment to carry out sensing, surveillance, data collection, or distributed servicing tasks. This topic covers a wide range of applications including search and rescue missions, air traffic control, automated highway systems, satellite networks, security systems, and many others. In each case, using a team of cooperative agents can be more efficient and reliable than using a single agent. With technological advances and developments of relatively inexpensive communication, computation, and sensing devices, this topic has received considerable attention over the last two decades [1-6].

The principles of search theory were introduced during World War II for the search of submarines [7-9] and furthered by the scientific community over the last decades [10-14]. The main objective of search theory is the distribution of search effort over an environment consisting of cells to maximize the probability of finding the object of interest. Typically, it is assumed that some prior knowledge about the distribution of the targets in environment is available. In the recent years, search missions with a team of mobile agents has received considerable attention in the search theory. Some studies in the literature have addressed the multi agent search problem in an uncertain environment, and a few approaches have been proposed accordingly. A probabilistic search formulation is used to describe the uncertainty in the environment and the problem is

usually converted to a multi agent path planning problem to find the optimal path of search agents. [15-19].

Cooperative coverage control studies the problem of covering a given domain using multiple agents. The objective of environment coverage is to distribute the agents across the environment while aggregating in more important areas. A distribution density function is usually defined that reflects a measure of relative importance of different regions in the terrain. The precise definition of the distribution density function depends on the desired application. Several solutions have been proposed to solve this problem which are usually based on Voronoi partitions and the Lloyd algorithm. However, in most of these studies it is assumed that the distribution density function in the environment is known *a priori* by all agents.

Multi agent search and coverage framework can provide a solution for the coverage problems in the uncertain environment where search agents are used to find the unknown distribution density function. By using a network of autonomous agents, the cooperative multi agent search and coverage framework can be used in many real-world applications involving distributed sensing and distributed actuation such as environmental monitoring and clean-up, forest fire detection and fighting or search and rescue [20-23].

Cooperative control is performed at three main levels: *high-level*, *mid-level* and *low-level* which are shown in Figure 1-1. The high-level design includes mission management, task assignment, timing/ scheduling, reconnaissance, and search algorithms. Some issues such as path planning, safety issues, collision avoidance, obstacle avoidance, formation keeping and trajectory following are designed in the mid-level. The low-level (or vehicle level) design discusses inner loop control, measurement noise and model uncertainty [24]. High-level decision making can be performed online or offline. In the case of online high-level decision making, it is necessary for

the central controller to communicate with each mid-level controller, either on a regular basis or on an event-based basis. When the high-level decisions are made offline, the mid-level controllers receive the required information from the high-level controller before starting their mission, and therefore there is no need for communication between the high-level controller and the mid-level controllers during the mission. The mid-level controllers should communicate with each other and with their corresponding low-level controller on a regular basis. In the search theory, the main focus is on the mid-level control. It is assumed that once the waypoints are defined by the mid-level controllers, there are appropriate low-level controllers available to guide the agents from one waypoint to the next one.

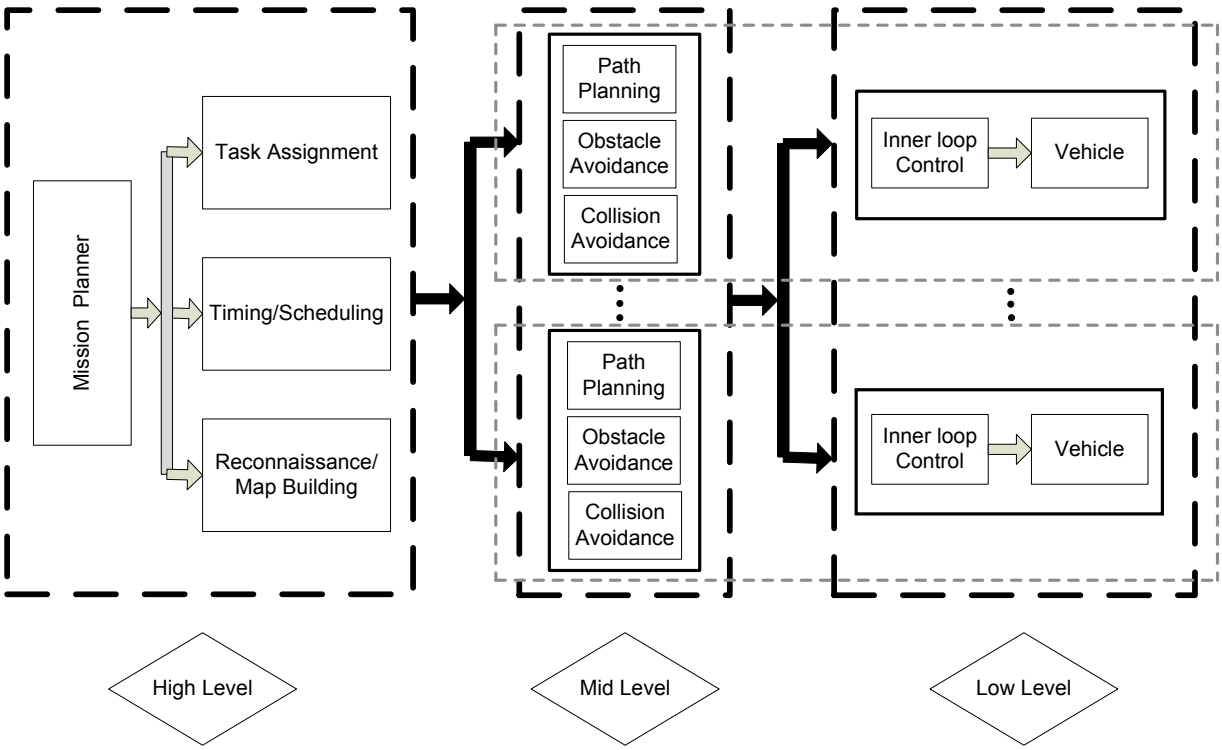


Figure 1-1. Hierarchy of Decentralized Cooperative Control

1.1 Literature Review

In this section, we first review the literature on cooperative control and decision making. The literature on probabilistic search and Bayesian update is reviewed in the next section. Then, different methods of path planning are reviewed. Finally, the recent developments in the field of coverage control are reviewed.

1.1.1 Cooperative Control and Decision making

Three different approaches were proposed for decision making and control of multi agent systems. The control of cooperative vehicles is traditionally performed in a centralized manner where the system as a whole is modeled and controlled by a single entity. For example, in [25] a centralized cooperative control strategy is presented for holonomic navigation in a planar world using an artificial potential function where a single controller constructs the collision free trajectories for all vehicles. The second approach is hierarchical which utilizes the distributed computational capacity of multiple platforms, and relies on a single facility to fuse information or resolve global constraints [26-30]. For example, in [30] a distributed hierarchical hybrid system is proposed for probabilistic pursuit-evasion games which emphasizes the autonomy of each agent while allowing for coordinated team efforts. Pursuit policy computation, map building and inter-agent communication are handled by central controller while individual agents are responsible for navigation, sensing, and control. Centralized decision making for a fleet of vehicle is not usually practical due to communication limits, robustness issues, and scalability. Using a hierarchical or distributed approach can mitigate many of these problems. In the hierarchical approach, the system consists of sub-teams using local communication networks to share information. Communication among the sub-teams is limited, although it is assumed to be available if necessary to exchange resources. The tasks can be selected by the sub-teams or by a coarse scheduling algorithm runs at

a higher level. Although hierarchical decision making and path planning reduces the dependency on a central planning system and increases the robustness of the overall mission to failure, its performance still depends on the central planner at the top hierarchy level. Thus, performance degradation is expected when the central planner fails.

The third approach is distributed or decentralized, which does not require any centralized facility, but instead relies on communication, consensus and negotiation [31-34]. In decentralized decision making, each agent decides on its own next action. However, it is essential that these control decisions be well coordinated among all the agents in order to maintain good overall performance. The objective of achieving tight coordination typically requires that the agents exchange large quantities of information about the environment, their current states, and their future intentions. The quantity of required information exchange depends on the level of coordination between agents. If it is possible to convert the multi agent decision making problem to a set of decoupled single agent decision making problems, the agents only need to share their information about the environment and their current states and then each agent simply solves its own decision making problem. However, usually multi agent decision making problems cannot be converted to sets of decoupled single agent decision making problems. In that case, one approach is that each vehicle determines its own mission by simultaneously choosing the path for all vehicles in the fleet. For instance, in [35], two decentralized recursive heuristics for multi agent Bayesian search problems are proposed, where each agent uses a centralized stochastic Dynamic Programming algorithm to find the set of all optimal actions of all agents in the team. When there are more than one optimal action, a uniform randomization operation is used by each agent to randomly choose its next action among all possible optimal. The other approach consists of a negotiation mechanism among vehicles to make proper decisions. For example, in [19], an agent

based negotiation scheme for multiple vehicles search is proposed where vehicles use negotiation as the decision making mechanism for obtaining their search paths. The negotiation schemes provide suboptimal solutions, but it is fast and scalable to large number of agents. To enable cooperation while complying with limited communication bandwidth, a decision system can be designed that does not require negotiation, but instead relies on the estimation of the next actions of other vehicles. In [34], a template is created around all vehicles which produces a grid of regions. The probability of existence of a vehicle in each cell in the near future is estimated based on the position of the cell in the surrounding template of the vehicle. Each vehicle then uses the estimated position of other vehicles in the future to modify its own objective function by decreasing the reward of searching a cell that may be searched by other vehicles. In [36], the possible paths of other vehicles are treated as soft obstacles. Each vehicle chooses its optimal path independently. To solve the optimization problem, an approximate dynamic programming method is developed where the cooperation among vehicles is achieved by using rivaling force approach. In [18], a cooperative search method is proposed where each vehicle uses feed-forward neural networks trained by a reinforcement learning to predict the states of other vehicles in its neighborhood and to utilize these predictions in its path planning process. The uncertainty map is updated using the Dempster-Shafer evidential method.

Impact of limited communications on a cooperative search algorithm for multiple unmanned aerial vehicles is studied in [37]. The results indicate that communication ranges has a significant impact on the group's ability to search an area. Achieving complete coordination among vehicles requires that all vehicles have complete information about the environment and current states of all vehicles contributing to the mission. This translates into a need to share a relatively large amount of information among vehicles. Limited bandwidth of communication channels and

potentially long distances between transmitter and receiver impose a delay in communication. This reduces the level of cooperation and might jeopardize the mission. If the vehicle can estimate the current states based on the delayed ones, then it can use estimated states to make decisions and compensate for the effects of communication delay, to some extent.

1.1.2 Probabilistic Search and Bayesian Update

Probabilistic Search theory studies the problem of searching for an object when the amount of searching effort is limited and only probabilities of possible position of the objects are known, where the problem is to find the optimal distribution of this total effort to maximize the probability of detection [38]. Most often, the search problem is mathematically formulated in discrete space (e.g. cells) and discrete effort (e.g. time space). Typical probabilistic search theory formulation describes the uncertainty in the environment by assigning probabilities of target existence to each cell of the environment which constructs the Probability Map of the environment [39,40]. The initial probability map is constructed based on the *a priori* information about the position of the targets. It is usually assumed that at each time step the probability of existence of the targets in each cell is a fixed value. However, in [41], this probability is described by a Beta distribution rather than a point estimate. Although Beta distribution is a very general distribution and theoretically it can be used to model almost any prior information about the presence of a target in the cell, finding appropriate parameters for the distribution function to model the prior information is generally very complicated. In fact, in most cases, it results in a simple uniform distribution. The sensor is generally assumed to be imperfect which means it is subject to false detection and missed detection errors. The probability distribution of the imperfect sensor is usually given by a Bernoulli distribution which is an appropriate model for a diverse array of sensor types, ranging from simple (e.g. bumper switch of robots) to sophisticated (e.g. visual object recognition) and is

applicable in variety of contexts. Therefore, the sensor measurement is subject to false-positive and false-negative errors. These errors are typically intrinsic to the specific sensing method that is utilized for detection and can be determined either empirically or by analytic approximation. Each observation made by the sensor provides new information about the environment which can be used to update the probability map. A Bayesian map building approach is usually used to update the probability map [42-46]. Much of the early research assumed no false-positive detection error [47, 48]. When the existence of different targets is highly correlated, the entire joint PDF of the targets is maintained [49]. However, this approach is intractable for large number of targets since the computational cost and memory usage exponentially increase with the number of targets [50]. In [52], the number of targets is known. Therefore, the probability of existence of the targets in different cells is highly correlated. To reduce the amount of computation, the relative probability values are defined such that only the value of the cell that is searched is changed. The footprint of sensor is usually equal to one cell which means at each time step, the sensor only observes one cell. The cell size depends on different factors including size of the targets, size of the environment, and computational capability of the mobile agents, while the sensor footprint is a property of the sensor and depends on different factors including the physical structure and the detection mechanism of the sensor. Therefore, in practice, the sensor footprint may contain several cells. In [51], the sensor has multiple cell footprint and the detection probabilities of the sensor depends on the range between the sensor and the observed point.

1.1.3 Multi Agent Path Planning

Path planning is responsible for moving the vehicles from one point to another [52]. The objective and approach of path planning differ depending on the application domain: surveillance [53, 54], search and rescue [55, 56], disaster monitoring [57], etc. Different methods are used to

design the paths based on the operating environment, physical limitation, and communication requirements. Application of multiple autonomous vehicles further increases the complexity of the path planning. There are a multitude of solution approaches available in the research literature and each approach has its own advantages and disadvantages.

In [58], a hierarchical path planning approach is pursued. A non-directional graph is used as the road map to represent the environment. The path for each vehicle is constructed using Voronoi diagram approach and the Dijkstra search algorithm to obtain minimum cost polygon path. This path is then modified by incorporating maneuverability constraints. In [59], a similar two step approach is used while considering the positional uncertainty of threat region. Instead of using the Voronoi diagram, the graph is based directly on the probability map. The probabilistic roadmap method is introduced in [60] which samples the given space for probable solution in the form of a network of graphs and connects the starting point to the goal point by adding successive trajectories to a pre-computed route. In [61], the rapidly exploring random trees approach is used where a tree of trajectory segments is extended from the start point to the goal point. Every successive trajectory is selected randomly by connecting to a closest point in the existing tree. In [62], the path planning is achieved by the rapidly exploring random trees and further enhanced by using Dijkstra search algorithm. The potential field method is used in [63] where the environment is presented as an artificial potential field. The destination is assigned an attractive potential, while the obstacles are assigned repulsive potential. The idea is that a vehicle moving in the field will be attracted towards the destination, while being repelled by the obstacles. In order to avoid getting trapped in a local maximum, authors in [64] use an adaptive potential field method with multiple auxiliary attraction points. The configuration of the optimum potential field is automatically determined by a genetic algorithm.

Many complete coverage path planning methods have been developed for the mobile robot coverage path planning problem, where one or more mobile robots are required to explicitly pass over all points in an unexplored region with obstacles to accomplish some tasks, such as floor cleaning, lawn mowing, and harvesting, [65-68]. The exhaustive search is a good strategy when the environment is uniform and static, and the agents have unlimited time and perfect target identification sensors. Optimal control is the most natural way to solve problems involving objective functions and constraints. However, the dimension and complexity of optimal control problems cause a heavy burden on computational time in the solution. Also, the nature of the problem may require either suboptimal or feasible solutions rather than the optimal one. Optimization techniques such as Dynamic Programming, Mixed Integer Linear Programming and Genetic Programming have been applied to path planning of vehicles. These techniques produce paths by optimizing certain cost function. The cost functions differ based on the applications, such as minimum time of arrival, optimizing fuel consumption, visiting more important areas, and coordinated motion. They are mostly search algorithm. The use of Mixed Integer Linear Programming for path planning applications can be found in [69-71]. Evolutionary algorithms are used in [72-74]. A *k*-shortest path algorithm based path generation methods is studied in [75]. The game theoretic approach is used in [76, 77]. In [78], health management is integrated with the cooperative path planning where the objective is to maximize the expected survival of the team of agents. Dynamic programming and heuristic technique are used to solve the problem. The foundation of Dynamic Programming is Bellman's equation [79] (also known as the Hamilton-Jacobi equations in control theory) which is most typically written in [80]. The gain for any time step is found by iterating enough times until the terminal gain is reached. However, as the dimension of the problem grows so does the computation time. To make the problem tractable,

and solvable in real-time, a limited look-ahead policy can be utilized [81, 82]. However, this solution is optimal with respect to the sub-problem, not in terms of the main problem. There is, therefore, a trade-off between optimality and computational complexity. In approximate Dynamic Programming, the objective function at the end of rolling horizon is approximate using different methods such as multilevel aggregation [83-86], basis functions [87], and neural networks [88].

Most of works in the field of multi vehicle search only consider the mid-level control. They convert the multi agent search problem into a multi agent path planning problem with some constraints. It is implicitly assumed that there is an appropriate low-level controller which is able to move the vehicles over their designated paths. However, there are a few studies that combine the low-level control of vehicles with path planning algorithm. In [89], a centralized gradient-type kinetic control strategy is proposed that guarantees each point in the domain is sampled by some agents in the network by any desired amount of effective coverage. The proposed control strategy is then modified for the case of partial communication between agents. A collision avoidance component is also added to the controller to guarantee that the agents do not collide. However, the proposed control strategy is centralized and not necessarily optimal. An individual state of awareness is defined for each vehicle in [90] which describes how aware the vehicles are of the events occurring over the entire domain. A decentralized control law is developed which guarantees a satisfactory state of awareness under dynamic communication structure and/or faulty sensors. A combined deployment and search strategy using Voronoi partitioning is proposed in [91]. The mobile agents deploy themselves to maximize reduction of uncertainty about the environment at each step, using a centralized Voronoi partitioning approach. The objective is to maximize single step search effectiveness and the results are only locally optimal. The range of sensors is assumed to be infinite and the collision avoidance issue is not addressed in this work.

1.1.4 Coverage Control

Cooperative Coverage control studies the problem of covering a given domain using multiple agents. In [92], the problem of fixed sensor network is investigated. The solution is based on Voronoi partitions and the Lloyd algorithm [93]. The algorithm can be calculated off-line and the optimal sensor location is the centroid of its Voronoi cell. Decentralized control laws based on both continuous and discrete-time versions of the classic Lloyd algorithms are designed in [94] such that the mobile sensor network covers an area partitioned into Voronoi region, in the sense that the system continually drives the agents toward the centroids of their Voronoi cells. The same problem is considered in [95] with a more realistic model for the sensors where their sensing ranges are restricted to a bounded region. In [96], Voronoi diagram is used to discover the existence of coverage holes, and different sensor deployment strategies are proposed to increase coverage. The idea of generalized Voronoi partition is used in [97] for the problem of area-constrained coverage. The area of the region assigned to each agent is assumed to be a pre-specified amount, and a Jacobi iterative algorithm is then used to assign the weights for generalized Voronoi partition that satisfies the area constraints. In [98] the generalized Voronoi partition is also used to adapt coverage to variable sensor performance and vehicle loss. The sensor health variable is added to the cost function and an iterative algorithm is proposed to adjust weights to satisfy cost constraints rather than area constraints. In [99], a deployment strategy is proposed for network of mobile agents such that the maximum traveling time the agents take to reach a place within the surveillance region is minimized.

Moreover, some research works consider more realistic environment. In [100], the non-convex environment is transformed to a proper convex region using a proper diffeomorphism where conventional Voronoi coverage can be applied. In [101], a discrete partitioning and

coverage control algorithm for a non-convex environment is presented. This method requires only unreliable short-range communication between pairs of robots. The problem of Voronoi coverage of non-convex regions when non-convexities can block visibility is investigated in [102]. The visible Voronoi diagram is defined and non-smooth optimization method is used to solve the problem. A space-partitioning algorithm is provided in [103] to address the problem of heterogeneous mobile sensors deployment to track a target in the field. In [104], a set of mobile sensors collaborates with a group of stationary sensors in order to detect an event. A path planning algorithm based on receding horizon optimization is presented to move the mobile sensors toward the areas that are least covered by the stationary sensors. The common assumption in the previous studies in the area of Voronoi-based coverage control is that the distribution of sensory information in the environment is known *a priori* by all agents. However, the problem of the online learning of the distribution density function is addressed in [105], and the density function is also estimated using neural networks in [106]. In [107], local interpolations are used to represent spatial fields as they are measured by a mobile sensor network which are able to take point measurements. A nonparametric estimate of the field is provided by two interpolation methods, which are refined via a Kalman filter-like recursion. In [108], an entropy-based metric is used to construct a map that determines the reachable regions of the environment. While the mobile robots explore the environment, they also use a centroid geodesic Voronoi tessellation to distribute themselves in such a way that the proper coverage is maintained, in the sense that mobile robots are distributed in the environment with more concentration around more important areas.

1.2 Motivations, Objectives and Contributions

This dissertation investigates distributed architectures for the multi agent search and coverage problem. Motivations, objectives and contributions of each chapter are as follows

- **Chapter 2**

Many studies about multi agent search consider perfect sensors which by one measurement can conclusively determine existence or non-existence of the target [109, 110]. Although this assumption simplifies the problem, it is not very practical. Much of the early research that consider imperfect sensors and use the Bayesian map building approach to update the probability map assumed no false-positive detection error. There are a few studies that use Bayesian map building approach to update the probability map and considered both false-positive and false-negative detection errors. Although some of these studies addressed issues like known number of targets in the environment, multi target scenarios, and sensors with multi cell footprints, they only consider special cases. To the best of author's knowledge there is no work that studies all of these issues including known number of targets in the environment, multi target scenarios, and sensors with multi cell footprints in a unified framework.

In chapter 2, we consider different conditions for the search problem in uncertain environments; different environment (with unknown number of targets or with known number of targets), different types of targets (distinguishable or indistinguishable), and different types of sensors (single-cell footprint or multiple-cell footprint). Any combination of these conditions makes a possible scenario for the search problem in uncertain environments and need a different probabilistic model. We develop probabilistic models for all possible scenarios and provide their probability map updating rules such that the models with single type of target are special cases of the models with multiple types of targets and the models with single-cell footprint are special cases of the models with multiple-cell footprint. Furthermore, for the models with known number of targets, we show that at each time step

the entire probability map should be updated which can be time consuming. In this case, the idea of relative probability is utilized which can significantly decrease the computational burden.

- **Chapter 3**

As we mentioned earlier, multi agent search problem in uncertain environment is an optimization problem in its nature. However, it is not possible to convert this optimization problem into multiple optimization problems for each agent. Therefore, most of studies in this area use centralized or hierarchical approach to solve the optimization problem. In some studies, each agent individually solves the global optimization problem which needs agent with high computational capability. There are a few works that use decentralized approach where each agent estimates the next action of other agents and uses that information in its decision making process. However, they need relatively high computation (when the estimation is online) or memory (when estimation is off line). It is important to notice that the mobile agents usually have limited computational capability. Therefore, reducing the required computation is a crucial task. Moreover, the mobile agents need to solve an optimization problem with dynamic programming. Thus, any extra available computation resource can be used to increase the look-ahead horizon of the dynamic programming method which can increase the performance of mission.

In chapter 3, a decentralized approach is used for the search mission where each mobile agent chooses its optimal action individually. To make cooperation between agents possible, two approximation methods are proposed to modify the objective function of agents and to take into the account the action of other agents. The first approach is a geometric estimation method which assume the position of an agents in the near future is on a moving arc with the

center of its current position. The second approach is a probabilistic estimation method which uses a Bayesian rule to find the position of agent in the near future. This approach requires more computation and more memory than the first one, but it can provide better performance. The simulation results show that the proposed methods can considerably reduce the computation burden with very little effect on the performance of the mission. This approach is then extended for the case with known commutation delay between mobile agents. The simulation results show that the proposed method can effectively compensate for the effect of communication delay.

- **Chapter 4**

The common assumption in the most previous studies in the area of Voronoi-based coverage control is that the distribution of sensory information in the environment is known *a priori* by all agents. There are a few works that address the coverage control in unknown environment. However, in all these studies, it is assumed that the unknown density function can be measured by each agent at its position. But, in many real applications, the density function is not directly measurable at each point. To solve this problem, in chapter 4, a new distribution density model is introduced which is a function of position of some unknown targets in the environment. The problem is formulated such that the information about the positions of the targets is updated by some search agents. The cooperative search method developed in chapter 3, along with a well-known Centroidal Voronoi Configuration method for the coverage control, is used to solve the problem.

- **Chapter 5**

Gathering information about the environment and finding the targets are the main objectives of cooperative search problem. Covering the area around the detected targets is the

objective of cooperative coverage problem. There are also several other issues such as minimum fuel consumption, refuelling, obstacle avoidance, and collision avoidance which are important in multi agent tasks. In chapter 5, the "limited turn rate Voronoi diagram" is introduced and the search and coverage problem is formulated as a multi-objective optimization problem with different constraints. Despite the method used in the previous section, there is only one type of agents which perform both search and coverage tasks. The "multi agent search and coverage problem" is formulated such that the "multi agent search problem" and "multi agent coverage problem" are special cases of this problem.

- **Chapter 6**

In chapter 6, a Voronoi-based search strategy for a team of mobile agents with limited range sensors is presented which combines mid-level and low-level controllers. Our work has several advantages over the similar works in the literature. It considers sensors with limited range. The collision avoidance between agents is guaranteed. The control law is designed to balance between the myopic objective of maximizing uncertainty reduction in the next step and the long term objective of distributing the agent in the environment with minimum overlap in their sensory domain. The dynamic model of agents is also a double integral which can express the equation of motion of a broad class of vehicles.

In addition, several numeric simulations are provided in chapters 3, 4, 5, and 6 to show the effectiveness of proposed methods. Some experimental results are also presented in chapter 4.

CHAPTER 2

Probabilistic Models for Uncertain Environments

In this chapter, different probabilistic models for uncertain environments are presented. The environment is discretized in cells which are described by a probability of target existence. There is a *probability map*, which contains the probability of existence of all targets in each cell. It is assumed that there is at most one target in each cell. The probability map is initialized by the *a priori* knowledge about the environment. If there is no prior information about the status of cell (whether or not there is a target in the cell), its initial probability would be 0.5. Using Shannon entropy, this initial probability corresponds to maximum uncertainty about the status of that cell [117]. During a search mission, a mobile sensor can detect targets in its footprint. After each measurement, the probability map must be updated based on whether or not a target is detected by the sensor. The main objective of this chapter is to develop practical probabilistic models for the uncertain environments and provide the updating rules for their probability maps.

The probability distribution of the sensor is given by a Bernoulli distribution which is an appropriate model for a diverse array of sensor types, ranging from simple (e.g. bumper switch of robots) to sophisticated (e.g. visual object recognition). It is a simplified but common sensor model which is applicable in variety of contexts. It abstracts away any complexity in the sensor detection process and results in a binary decision [41]. The sensor measurement is subject to false-positive and false-negative errors. These errors are typically intrinsic to the specific sensing method that is utilized for detection and can be determined either empirically or by analytic approximation.

We consider search problems with different conditions; unknown number of targets in the environment or known number of targets in the environment, single type of targets or multiple types of targets, and sensors with single-cell footprint or sensors with multiple-cell footprint. Any combination of these conditions can construct a possible scenario for the probabilistic search in uncertain environments. In total, ten different scenarios are investigated and the updating rules for the probability maps are provided. Figure 2-1 shows different scenarios which are investigated in this chapter. Each of these scenarios is appropriate for a specific type of problems.

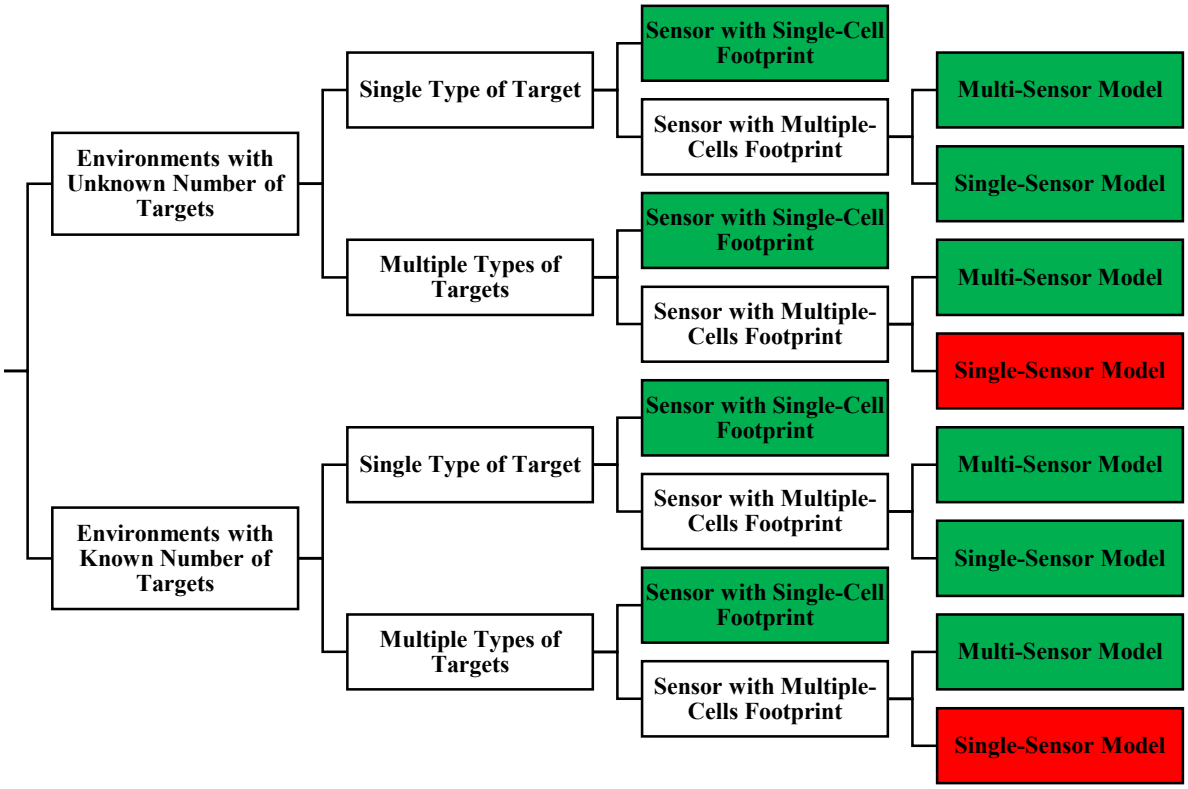


Figure 2-1. Different scenarios which are investigated in this chapter. Green: The updating rule for the probability map is provided. Red: The sensor provides little information about the environment; therefore, the probability map updating rule has not been discussed in this study.

2.1 Environments with Unknown Number of Targets

In this section, we study scenarios where the number of targets in the search domain is not known *a priori*.

2.1.1 Single Type of Target

In all scenarios in this section, there is an unknown number of similar targets in the environment.

2.1.1.1 Sensor with Single-Cell Footprint

In the first scenario, there is an unknown number of similar targets in the environment and the sensor has a single-cell footprint which can detect a target that resides in its current cell. Event E_q is the event that a target is in the cell q and D_q is the event that a target is detected in the cell q.

The probabilities of *true positive* and *false positive* measurement of sensors are assumed to be $\gamma = P(D_q|E_q)$ and $\varepsilon = P(D_q|\bar{E}_q)$ respectively, where γ is the probability of detecting a target and ε is the probability of reporting a target existence while it does not really exist. These two parameters are specification of sensors and assumed to be known *a* priori. When an agent enters a cell, its sensor can measure the cell which has two possible outputs; there is a target in the environment ($I = 1$) or there in not ($I = 0$).

When the sensor has not detected a target, the probability of existence of the target in the cell can be updated as follows, using the Bayes' Rule

$$P(E_q|\bar{D}_q) = \frac{P(E_q)P(\bar{D}_q|E_q)}{P(\bar{D}_q)} = \frac{P(E_q)\bar{\gamma}}{P(\bar{D}_q)}$$

where an overbar on the events represents the complement of the events.

Using a similar procedure, when a sensor has detected a target, the *posterior* probability of existence of the target in the cell can be computed as follows

$$P(E_q|D_q) = \frac{P(E_q)P(D_q|E_q)}{P(D_q)} = \frac{P(E_q)\gamma}{P(D_q)}$$

Therefore, we can update the probability of existence of the target in that cell, based on the output of the sensor as follows

$$p_q^* = \frac{p_q \cdot \bar{\gamma}}{P(\bar{D}_q)}(1 - I) + \frac{p_q \cdot \gamma}{P(D_q)} I$$

where p_q and p_q^* are the probabilities of existence of a target in the cell q before and after the visit respectively. Using law of total probability, $P(D_q)$ can be found as follows

$$P(D_q) = P(D_q|E_q)P(E_q) + P(\bar{E}_q)P(D_q|\bar{E}_q) = \gamma\, \mathrm{p_q} + \varepsilon(1 - p_q)$$

and

$$P(\bar{D}_q) = 1 - P(D_q) = \bar{\gamma}\, p_q + \bar{\varepsilon}(1 - p_q)$$

Therefore, the mobile sensor modifies the probability map by updating the probability of its current cell, q, using the following *Probability Map Updating Rule*

$$p_{\mathrm{q}}^* = \frac{p_{\mathrm{q}} \cdot \bar{\gamma}}{\bar{\gamma}\, p_{\mathrm{q}} + \bar{\varepsilon}(1 - p_{\mathrm{q}})}(1 - \mathrm{I}) + \frac{p_{\mathrm{q}} \cdot \gamma}{\gamma\, \mathrm{p_q} + \varepsilon(1 - p_{\mathrm{q}})} \mathrm{I} \qquad (2\text{-}1)$$

2.1.1.2 Sensor with Multiple-Cell Footprint

In this section, there is an unknown number of similar targets in the environment and the footprint of sensor consists of multiple cells. It is worth to mention that, in a search mission, even when the actual footprint of sensor is one cell, if the probability map is updated after several time steps, the updating rule for the multiple-cell footprint may need to be utilized.

2.1.1.2.1 Multi-Sensor Model

In the second scenario, there is an unknown number of similar targets in the environment. The footprint of sensor consists of multiple cells and the sensor can detect the target in each cell separately. In this scenario, updating rule (2-1) should still be used for each cell individually. In fact, one can replace such sensor with an array of several sensors with single-cell footprint and use (2-1) to update the probability of existence of the target in all cells in the sensor footprint.

Many types of sensors have some kind of quality deterioration or signal attenuation based on distance [111]. In these cases, a more realistic model can be used where the values of γ and ε are assumed to depend on the distance between the sensor and the cell. If we define $r = \|\mathbf{q} - \mathbf{p}\|$, where $\mathbf{p}$ is the location of sensor and $\mathbf{q}$ is the location of cell being observed and parameter r_γ as the range of sensor, function $\gamma(r)$ must have the following properties

- A peak value at the location of sensor, i.e. $\gamma(\mathbf{p}) > \gamma(\mathbf{q}), \forall \mathbf{p} \neq \mathbf{q}$
- A decreasing function of r, i.e. $\gamma(\mathbf{q_1}) > \gamma(\mathbf{q_2})$, if $\|\mathbf{q_1} - \mathbf{p}\| < \|\mathbf{q_2} - \mathbf{p}\|$
- Equal to 0.5 for all cells outside of sensor range, $\gamma(\mathbf{q})$ =0.5, if $\|\mathbf{q} - \mathbf{p}\| > r_\gamma$

This model indicates that the sensor's detection probability is maximum at its position and decreases with the distance. The probability is equal to 0.5 outside of sensing range which implies that it is equally likely for sensor to truly detect a target or miss it outside the sensing range. The *true negative* measurement, i.e. $1 - \varepsilon(r)$, must have similar properties. Therefore, the function $\varepsilon(r)$ has the following properties

- A bottom value at the location of sensor, i.e. $\varepsilon(\mathbf{p}) < \varepsilon(\mathbf{q}), \forall \mathbf{p} \neq \mathbf{q}$
- An increasing function of r, i.e. $\varepsilon(\mathbf{q_1}) < \varepsilon(\mathbf{q_2})$, if $\|\mathbf{q_1} - \mathbf{p}\| < \|\mathbf{q_2} - \mathbf{p}\|$
- Equal to 0.5 for all cells outside of sensor range, $\varepsilon(\mathbf{q})$ =0.5, if $\|\mathbf{q} - \mathbf{p}\| > r_\varepsilon$

where r_ε is the range of sensor.

Therefore, if we define Ω as the collection of cells inside the sensor footprint, the mobile sensor modifies the probability map by updating the probability of all cells inside Ω, i.e. $\forall q \in \Omega$, using the following *Probability Map Updating Rule*:

$$p_{\mathrm{q}}^{*} = \frac{p_{\mathrm{q}} \cdot \bar{\gamma}(\mathrm{r})}{\bar{\gamma}(\mathrm{r})\, p_{\mathrm{q}} + \bar{\varepsilon}(\mathrm{r})\left(1 - p_{\mathrm{q}}\right)} \left(1 - \mathrm{I}_{\mathrm{q}}\right) + \frac{p_{\mathrm{q}} \cdot \gamma(\mathrm{r})}{\gamma(\mathrm{r})\, p_{\mathrm{q}} + \varepsilon(\mathrm{r})\left(1 - p_{\mathrm{q}}\right)} \mathrm{I}_{\mathrm{q}} \tag{2-2}$$

where I_{q} is the output of sensor corresponding to the cell q.

A practical model for γ that we will use in the following chapters is a second-order polynomial function of r as follows

$$\gamma(\mathbf{q}) = \begin{cases} \frac{\gamma_0 - 0.5}{{r_\gamma}^2}\left({r_\gamma}^2 - r^2\right) + 0.5 & r \le r_\gamma \\ 0.5 & r > r_\gamma \end{cases} \tag{2-3}$$

where γ_0 $(0.5 \le \gamma_0 \le 1)$ is the peak value of γ at the observation point. Similarly, the model of *false positive* measurement ε is as follows

$$\varepsilon(\mathbf{q}) = \begin{cases} \frac{\varepsilon_0 - 0.5}{{r_\varepsilon}^2}\left({r_\varepsilon}^2 - r^2\right) + 0.5 & r \le r_\varepsilon \\ 0.5 & r > r_\varepsilon \end{cases} \tag{2-4}$$

where $\varepsilon_0 (0 \le \varepsilon_0 \le 0.5)$ is the bottom value of ε at the observation point and r_ε is the range of sensor. This is an appropriate model for electromagnetic or acoustic sensors [112].

Remark: It is a natural assumption that the sensor ranges for *true positive* and *true negative* measurements of the sensor are equal, i.e. $r_\gamma = r_\varepsilon$.

It should be noted that the sensor measurement does not change the probability of existence of the target outside its range, i.e. the cells with distance more than $r_\gamma = r_\varepsilon$ from the sensor.

Therefore, after each measurement, the probability map update is only performed for cells inside the sensor range.

2.1.1.2.2 Single-Sensor Model

In the third scenario, there is an unknown number of similar targets in the environment. The footprint of sensor consists of multiple cells and the sensor can only report the existence or non-existence of the target in its entire footprint. In this scenario, the target detection corresponds to detection of at least one target in the entire sensor footprint. We can still replace this sensor with several virtual sensors with single-cell footprint. However, in this case, if we define I_{q_i} as the output of the virtual sensor corresponding to the cell q_i, the output of the mobile sensor is $I = \bigvee_{q_i \in \Omega} I_i$, where Ω is the collection of cells inside the sensor footprint. Event E_{q_i} is the event that a target is in the cell $q_i \in \Omega$, event D_{q_i} is the event that the virtual sensor corresponding to the cell q_i detects the target in that cell, and N_Ω is the number of cells inside the footprint of sensor. Then, event D_Ω is the event that a target is detected by the mobile sensor and is equal to $D_\Omega = \bigcup_{q_i \in \Omega} D_{q_i}$. The $2N_\Omega$ parameters $\gamma_i = P(D_{q_i}|E_{q_i})$ and $\varepsilon_i = P(D_{q_i}|\bar{E}_{q_i})$ are specification of sensor and must be known *a priori*. In general, γ_i and ε_i can be functions of the distance between the location of the sensor and the position of the cell q_i.

When the sensor detects a target, by using the Bayes' Rule, the probability of existence of the target in any cell q_i inside the sensor footprint can be updated as follows

$$P(E_{q_i}|D_\Omega) = \frac{P(D_\Omega|E_{q_i}).P(E_{q_i})}{P(D_\Omega)} \tag{2-5}$$

where

$$P\left(D_\Omega \middle| E_{q_i}\right) = P\left(\cup_{\forall j} D_{q_j} \middle| E_{q_i}\right)$$

$$= + \sum_{\forall j_1} P\left(D_{q_{j_1}} \middle| E_{q_i}\right) - \sum_{\forall j_1, j_2} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \middle| E_{q_i}\right) + \sum_{\forall j_1, j_2, j_3} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \cap D_{q_{j_3}} \middle| E_{q_i}\right) - \cdots + (-1)^{N_\Omega} P\left(\cap_{\forall j} D_{q_j} \middle| E_{q_i}\right)$$

$$= + P\left(D_{q_i} \middle| E_{q_i}\right) + \sum_{\forall j_1 \neq i} P\left(D_{q_{j_1}} \middle| E_{q_i}\right) - \sum_{\forall j_1} P\left(D_{q_{j_1}} \cap D_{q_i} \middle| E_{q_i}\right) - \left(\frac{1}{2!}\right) \sum_{\substack{\forall j_1, j_2 \\ \neq i}} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \middle| E_{q_i}\right) + \left(\frac{1}{2!}\right) \sum_{\substack{\forall j_1, j_2 \\ \neq i}} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \cap D_{q_i} \middle| E_{q_i}\right) + \left(\frac{1}{3!}\right) \sum_{\substack{\forall j_1, j_2, j_3 \\ \neq i}} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \cap D_{q_{j_3}} \middle| E_{q_i}\right) - \cdots + (-1)^{N_\Omega} P\left(\cap_{\forall j} D_{q_j} \middle| E_{q_i}\right)$$

$$= + P\left(D_{q_i} \middle| E_{q_i}\right) + \sum_{\forall j_1 \neq i} P\left(D_{q_{j_1}}\right) - \sum_{\forall j_1} \left(P\left(D_{q_i} \middle| E_{q_i}\right) P\left(D_{q_{j_1}}\right)\right) - \left(\frac{1}{2!}\right) \sum_{\substack{\forall j_1, j_2 \\ \neq i}} \left(P\left(D_{q_{j_1}}\right) P\left(D_{q_{j_2}}\right)\right) + \left(\frac{1}{2!}\right) \sum_{\substack{\forall j_1, j_2 \\ \neq i}} \left(P\left(D_{q_i} \middle| E_{q_i}\right) P\left(D_{q_{j_1}}\right) P\left(D_{q_{j_2}}\right)\right) + \left(\frac{1}{3!}\right) \sum_{\substack{\forall j_1, j_2, j_3 \\ \neq i}} \left(P\left(D_{q_{j_1}}\right) P\left(D_{q_{j_2}}\right) P\left(D_{q_{j_3}}\right)\right) - \cdots + (-1)^{N_\Omega} P\left(D_{q_i} \middle| E_{q_i}\right) \prod_{\forall j_1 \neq i} P\left(D_{q_{j_1}}\right)$$

$$
\begin{aligned}
= \quad & +\gamma_i + (1-\gamma_i)\sum_{\forall j_1 \neq i}(\gamma_{j_1}.\mathrm{p}_{j_1} + \varepsilon_{j_1}.\bar{\mathrm{p}}_{j_1}) \\
& -(\tfrac{1}{2!})(1-\gamma_i)\sum_{\forall j_1,j_2 \neq i}(\gamma_{j_1}.\mathrm{p}_{j_1} + \varepsilon_{j_1}.\bar{\mathrm{p}}_{j_1}).(\gamma_{j_2}.\mathrm{p}_{j_2} + \varepsilon_{j_2}.\bar{\mathrm{p}}_{j_2}) \\
& +(\tfrac{1}{3!})(1-\gamma_i)\sum_{\forall j_1,j_2,j_3 \neq i}(\gamma_{j_1}.\mathrm{p}_{j_1} + \varepsilon_{j_1}.\bar{\mathrm{p}}_{j_1}).(\gamma_{j_2}.\mathrm{p}_{j_2} + \varepsilon_{j_2}.\bar{\mathrm{p}}_{j_2})(\gamma_{j_3}.\mathrm{p}_{j_3} + \varepsilon_{j_3}.\bar{\mathrm{p}}_{j_3}) \\
& - \cdots \\
& +(-1)^{N_\Omega}(1-\gamma_i)\prod_{\forall j_1 \neq i}(\gamma_j.\mathrm{p}_{j_1} + \varepsilon_i.\bar{\mathrm{p}}_{j_1}) \qquad (2\text{-}6)
\end{aligned}
$$

and

$$
\begin{aligned}
P(D_\Omega) &= P\left(\bigcup_{\forall j} D_{q_j}\right) \\
= \quad & +\sum_{\forall j_1} P\left(D_{q_{j_1}}\right) \\
& -\sum_{\forall j_1,j_2} P\left(D_{q_{j_1}} \cap D_{q_{j_2}}\right) \\
& +\sum_{\forall j_1,j_2,j_3} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \cap D_{q_{j_3}}\right) \\
& - \cdots \\
& +(-1)^{N_\Omega} P\left(\bigcap_{\forall j} D_{q_j}\right) \\
= \quad & +\sum_{\forall j_1} P\left(D_{q_{j_1}}\right) \\
& -(\tfrac{1}{2!})\sum_{\forall j_1,j_2} P\left(D_{q_{j_1}} \cap D_{q_{j_2}}\right) \\
& +(\tfrac{1}{3!})\sum_{\forall j_1,j_2,j_3} P\left(D_{q_{j_1}} \cap D_{q_{j_2}} \cap D_{q_{j_3}}\right) \\
& - \cdots \\
& +(-1)^{N_\Omega} P\left(\bigcap_{\forall j} D_{q_j}\right)
\end{aligned}
$$

$$= + \sum_{\forall j_1} P\left(D_{q_{j_1}}\right) - \left(\frac{1}{2!}\right) \sum_{\forall j_1, j_2} \left(P\left(D_{q_{j_1}}\right) P\left(D_{q_{j_2}}\right)\right) + \left(\frac{1}{3!}\right) \sum_{\forall j_1, j_2, j_3} \left(P\left(D_{q_{j_1}}\right) P\left(D_{q_{j_2}}\right) P\left(D_{q_{j_3}}\right)\right) - \cdots + (-1)^{N_\Omega} \prod_{\forall j_1} P\left(D_{q_{j_1}}\right)$$

$$= + \sum_{\forall j} (\gamma_j . p_j + \varepsilon_i . \bar{p}_j) - \left(\frac{1}{2!}\right) \sum_{\forall j_1, j_2} (\gamma_{j_1} . p_{j_1} + \varepsilon_{j_1} . \bar{p}_{j_1}) . (\gamma_{j_2} . p_{j_2} + \varepsilon_{j_2} . \bar{p}_{j_2}) + \left(\frac{1}{3!}\right) \sum_{\forall j_1, j_2, j_3} (\gamma_{j_1} . p_{j_1} + \varepsilon_{j_1} . \bar{p}_{j_1}) . (\gamma_{j_2} . p_{j_2} + \varepsilon_{j_2} . \bar{p}_{j_2}) (\gamma_{j_3} . p_{j_3} + \varepsilon_{j_3} . \bar{p}_{j_3}) - \cdots + (-1)^{N_\Omega + 1} \prod_{\forall j \neq i} (\gamma_j . p_j + \varepsilon_i . \bar{p}_j) \quad (2\text{-}7)$$

and $p_i = p_{q_i} = P\left(E_{q_i}\right)$. In deriving (2-6) and (2-7) we used the fact that, if $i \neq j$, the events E_{q_i} and D_j are independent and the events D_i and D_j are also independent, and $P\left(D_{q_j}\right) = P\left(D_j \middle| E_{q_j}\right) P\left(E_{q_j}\right) + P\left(D_j \middle| \bar{E}_{q_j}\right) P\left(\bar{E}_{q_j}\right) = \gamma_j . p_j + \varepsilon_j . \bar{p}_j$.

Similarly, when the sensor does not detect a target, the probability of existence of the target in any cell q_i inside the sensor footprint can be updated as follows

$$P\left(E_{q_i} \middle| \bar{D}_\Omega\right) = \frac{P(\bar{D}_\Omega | E_{q_i}) . P(E_{q_i})}{P(\bar{D}_\Omega)} \quad (2\text{-}8)$$

where

$$P\left(\bar{D}_{\Omega}\middle|E_{q_i}\right) = +\bar{\gamma}_i + (1-\bar{\gamma}_i)\sum_{\forall j\neq i}(\bar{\gamma}_j.p_{\mathrm{j}} + \bar{\varepsilon}_i.\bar{p}_{\mathrm{j}})$$
$$-(\tfrac{1}{2!})(1-\bar{\gamma}_i)\sum_{\substack{\forall j_1,j_2\\ \neq i}}\left(\bar{\gamma}_{j_1}.p_{j_1} + \bar{\varepsilon}_{j_1}.\bar{p}_{j_1}\right).\left(\bar{\gamma}_{j_2}.p_{j_2} + \bar{\varepsilon}_{j_2}.\bar{p}_{j_2}\right)$$
$$+(\tfrac{1}{3!})(1-\bar{\gamma}_i)\sum_{\substack{\forall j_1,j_2,j_3\\ \neq i}}\left(\bar{\gamma}_{j_1}.p_{j_1} + \bar{\varepsilon}_{j_1}.\bar{p}_{j_1}\right).\left(\bar{\gamma}_{j_2}.p_{j_2} + \bar{\varepsilon}_{j_2}.\bar{p}_{j_2}\right)\left(\bar{\gamma}_{j_3}.p_{j_3} + \bar{\varepsilon}_{j_3}.\bar{p}_{j_3}\right)$$
$$-\ \cdots$$
$$+(-1)^{N_{\Omega}}(1-\bar{\gamma}_i)\prod_{\forall j\neq i}(\bar{\gamma}_j.p_{\mathrm{j}} + \bar{\varepsilon}_i.\bar{p}_{\mathrm{j}}) \qquad (2\text{-}9)$$

and

$$P(D_{\Omega}) = +\sum_{\forall j}(\bar{\gamma}_j.p_{\mathrm{j}} + \bar{\varepsilon}_i.\bar{p}_{\mathrm{j}})$$
$$-(\tfrac{1}{2!})\sum_{\forall j_1,j_2}\left(\bar{\gamma}_{j_1}.p_{j_1} + \bar{\varepsilon}_{j_1}.\bar{p}_{j_1}\right).\left(\bar{\gamma}_{j_2}.p_{j_2} + \bar{\varepsilon}_{j_2}.\bar{p}_{j_2}\right)$$
$$+(\tfrac{1}{3!})\sum_{\forall j_1,j_2,j_3}\left(\bar{\gamma}_{j_1}.p_{j_1} + \bar{\varepsilon}_{j_1}.\bar{p}_{j_1}\right).\left(\bar{\gamma}_{j_2}.p_{j_2} + \bar{\varepsilon}_{j_2}.\bar{p}_{j_2}\right)\left(\bar{\gamma}_{j_3}.p_{j_3} + \bar{\varepsilon}_{j_3}.\bar{p}_{j_3}\right)$$
$$-\ \cdots$$
$$+(-1)^{N_{\Omega}+1}\prod_{\forall j\neq i}(\bar{\gamma}_j.p_{\mathrm{j}} + \bar{\varepsilon}_i.\bar{p}_{\mathrm{j}}) \qquad (2\text{-}10)$$

Therefore, the mobile sensor modifies the probability map by updating the probabilities of all cells in its footprint, i.e. $\forall q_i \in \Omega$, using the following *Probability Map Updating Rule*

$$p_{\mathrm{i}}^{*} = P\left(E_{q_i}\middle|\bar{D}_{\Omega}\right)(1-\mathrm{I}) + P\left(E_{q_i}\middle|D_{\Omega}\right)\mathrm{I} \qquad (2\text{-}11)$$

where $P\left(E_{q_i}\middle|\bar{D}_{\Omega}\right)$ and $P\left(E_{q_i}\middle|D_{\Omega}\right)$ are provided by (2-5) and (2-8), respectively.

Remark 1: Equation (2-1) is a special case of (2-11), when $N_{\Omega} = 1$.

Remark 2: In many cases, parameters $\gamma_i = P\left(D_{q_i}\middle|E_{q_i}\right)$ and $\varepsilon_i = P\left(D_{q_i}\middle|\bar{E}_{q_i}\right)$ are not known for every cell q_i inside the sensor footprint. In fact, often the only available information about the sensor is the probability of detecting the target given that a target is in the sensor footprint, i.e. $\gamma =$

$P(D_\Omega|E_\Omega)$, and the probability of detecting a target while there is no target in the sensor footprint, i.e. $\varepsilon = P(D_\Omega|\bar{E}_\Omega)$. In this case, in (2-6), (2-7), (2-9), and (2-10), γ_i must be replaced by γ and ε_i must be replaced by ε for all i.

Remark 3: When the size of cells is small with respect to the size of sensor footprint, N_Ω is a large number. Therefore, updating the probability map by using (2-11) needs too many calculations. However, in this case, the sensor does not provide much information about the cell in its footprint and it should be replaced with a sensor with finer resolution.

2.1.2 Multiple Types of Targets

In all scenarios in this section, there is an unknown number of different distinguishable targets in the environment. The sensors are able to detect different types of targets. In other words, they can detect an object and classify it as one of the possible targets. The probability map now contains the probability of existence of different types of targets in all cells.

2.1.2.1 Sensor with Single-Cell Footprint

In the fourth scenario, there is an unknown number of different distinguishable targets in the environment and the footprint of sensors is a single cell. Therefore, the sensor can detect and classify an object which resides in its current cell. The agents are equipped with imperfect sensors with categorical distribution which is the generalization of the Bernoulli distribution for the case with more than two possible outcomes. We define E_q^i as the event that the target i is in the cell q and D_q^i as the event the target i is detected in the cell q.

Parameter η_j^i is defined as the probability of detecting target i given the actual target is j, where $i, j \in [0, m]$ and m is the number of possible targets, i.e. $\eta_j^i = P(D_q^i|E_q^j)$. Index zero corresponds to the situation that there is no target. Therefore, η_j^0 is the probability of detecting no

target given target j exists in the cell, and η_0^j is the probability of detecting target j while there is no target in the cell. It is expected that the probability of *true positive* measurement of all targets is greater than 0.5, i.e. $\eta_j^j > 0.5$ for $\forall j \in [0, m]$. It is also expected that $\sum_{i=0}^{m} \eta_j^i = 1$ for $\forall j \in [0, m]$. The probability transition matrix $\mathbf{P}[\mathrm{P_{i,j}} = \eta_j^i]$ is obtained from technical specifications on the sensors, and is considered to be known *a priori*.

We define p_q^i as the probability of existence of the target i in the cell q. Since it is only possible to have at most one target in each cell, $p_q = \sum_{i=1}^{m} p_q^i$ is always less than or equal to one. Random variable T is defined to be equal to the output of the sensor, i.e. $T = i$ means that the sensor has detected target i in the cell. Again $T = 0$ means no target has been detected in the cell.

When $T = i$, the probability p_q^j can be updated as follows

$$p_q^{*j} = P(E_q^j | D_q^i) = \frac{p_q^j . \eta_j^i}{P(D_q^i)}$$

where the superscript $*$ indicates the updated value. Using the law of total probability,

$$P\left(D_q^i\right) = \sum_{k=0}^{m} \left(P\left(D_q^i | E_q^k\right) . P\left(E_q^k\right) \right) = \sum_{k=0}^{m} \left(\eta_k^i . p_q^k\right)$$

where $p_q^0 = 1 - p_q$ is the probability that no target exists in the cell. Therefore, when the target is measured as i, the posterior probability of existence of the target j in the cell can be updated by using the following equation

$$p_q^{*j} = \frac{p_q^j . \eta_j^i}{\sum_{k=0}^{n} \eta_k^i . p_q^k}$$

In general, the probability map can be updated using the following *Probability Map Updating Rule*

$$p_q^{*j} = \sum_{i=0}^{m} \left(\frac{p_q^j \cdot \eta_j^i}{\sum_{k=0}^{m} \eta_k^i \cdot p_q^k} \cdot \delta_{iT}\right) \tag{2-12}$$

where δ_{iT} is the Kronecker delta which is equal to one when $T = i$, and is equal to zero otherwise.

Remark 1: The updating rule (2-1) is a special case of (2-12) where $\eta_1^1 = \gamma$ and $\eta_0^1 = \varepsilon$.

The probability of detecting the target i given the actual target is j, i.e. $\eta_j^i = P(D_q^i | E_q^j)$ can be decomposed into two parts; the probability of classifying the target as i given the target j has been detected, i.e. $\mu_j^i = P(D_q^i | E_q^j \cap D_q)$, and the probability of detecting an object given the target j exists, i.e. $\gamma_j = P(D_q | E_q^j)$. Then

$$\eta_j^i = \mu_j^i . \gamma_j \tag{2-13}$$

If the probability of detecting all targets is equal, we define $\gamma = P\left(D_q \middle| E_q^j\right), \forall j \in [1, m]$, and $\varepsilon = P\left(D_q \middle| E_q^0\right)$. Therefore, the updating equation (2-12) can be written as

$$p_q^{*j} = \sum_{i=0}^{m} \left(\frac{\gamma . \bar{\eta}_j^i . p_q^j}{\gamma . \sum_{k=1}^{m} \left(\mu_k^i . p_q^k\right) + \varepsilon \mu_0^i . p_q^0} \cdot \delta_{iT}\right), \quad \forall j \neq 0 \tag{2-14}$$

and

$$p_q^{*0} = \sum_{i=0}^{m} \left(\frac{\varepsilon . \bar{\eta}_j^i . p_q^j}{\gamma . \sum_{k=1}^{m} \left(\mu_k^i . p_q^k\right) + \varepsilon \mu_0^i . p_q^0} \cdot \delta_{iT}\right) \tag{2-15}$$

Remark 2: When the classification error is equal for all targets, it is equally probable to falsely detect any target other than the one that really exists in the cell, which means μ_j^i is the same for $\forall i \in [0, m], i \neq j$.

2.1.2.2 Sensor with Multiple-Cell Footprint

In this section, there is an unknown number of different distinguishable targets in the environment and the footprint of sensor consists of multiple cells.

2.1.2.2.1 Multi-Sensor Model

In the fifth scenario, there is an unknown number of different distinguishable targets in the environment. The footprint of sensor consists of multiple cells and the sensor can detect the targets in each cell separately. In this scenario, updating rule (2-12) should still be used for each cell individually. In fact, one can replace such sensor with several sensors with one cell footprint and use (2-12) to update probability of existence of the target in all cells in the sensor footprint.

A more general model is the one that the value of η_j^i is assumed to depend on the distance between the sensor and the cell. We define $r = \|\mathbf{q} - \mathbf{p}\|$ as the distance between the location of sensor, $\mathbf{p}$, and the location of the cell being observed, $\mathbf{q}$, and parameter r_η as the range of sensors. Then the functions $\eta_i^i(r)$ must have the following properties

- A peak value at the location of sensor, i.e. $\eta_i^i(\mathbf{p}) \geq \eta_i^i(\mathbf{q}), \forall \mathbf{p} \neq \mathbf{q}$
- A non-increasing function of r, i.e. $\eta_i^i(\mathbf{q_1}) \geq \eta_i^i(\mathbf{q_2})$, if $\|\mathbf{q_1} - \mathbf{p}\| \leq \|\mathbf{q_2} - \mathbf{p}\|$

This model indicates that the capability of the sensor in detecting and classifying a target is maximum at its position and does not increase with the distance.

Similarly, a model of *false Classification*, i.e. $\eta_i^j, \forall j \neq i$, must have the following properties

- A bottom value at the location of sensor, i.e. $\eta_i^j(\mathbf{p}) \leq \eta_i^j(\mathbf{q}), \forall \mathbf{p} \neq \mathbf{q}$
- Anon-decreasing function of r, i.e. $\eta_i^j(\mathbf{q_1}) \leq \eta_i^j(\mathbf{q_2})$, if $\|\mathbf{q_1} - \mathbf{p}\| \leq \|\mathbf{q_2} - \mathbf{p}\|$

Additionally, for all cells outside the sensor footprint, i.e.$\|\mathbf{q} - \mathbf{p}\| > r_\eta$, the following property must hold

- $\eta_i^j(\mathbf{q}) = \eta_i^k(\mathbf{q})$ for $\forall i, j, k \in [0, m]$

which indicates that it is equally probable for the sensor to report existence of any target in a cell outside its footprint, regardless of the actual target. Therefore, $\eta_i^j(r) = \frac{1}{m+1}, \forall i,j \in [0,m]$, for $r > r_\eta$.

Therefore, if we define Ω as the collection of cells inside the sensor footprint, the mobile sensor modifies the probability map by updating the probability of all cells inside Ω, i.e. $\forall q \in \Omega$, using the following *Probability Map Updating Rule*

$$p_q^{*j} = \sum_{i=0}^{m} \left(\frac{p_q^j . \eta_j^i(r)}{\sum_{k=0}^{m} \eta_k^i(r) . p_q^k} . \delta_{i\mathrm{T_q}} \right) \tag{2-16}$$

where $\mathrm{T_q}$ is the output of sensor corresponding to the cell q.

Remark: The updating rule (2-2) is a special case of (2-16) where $m = 1$, $\eta_1^1(r) = \gamma(r)$ and $\eta_0^1(r) = \varepsilon(r)$.

It should be noticed that the sensor measurement does not change the probability of existence of the targets outside its footprint. Therefore, after each measurement, probability map updating is only performed for cells inside the sensor footprint.

2.1.2.2.2 Single-Sensor Model

The other possible case is when the sensor can only report the existence or non-existence of an object in its entire footprint and classify it as one of known possible targets. In this case, the sensor provides very little information about the probability of existence of the targets in each cell inside its footprint which is not useful from practical point of view. Therefore, we will not discuss this model in this study.

2.2 Environments with Known Number of Targets

In all scenarios that discussed in section 2.1, the actual number of targets in the search domain was unknown *a priori*. In this section, we study the search problem in uncertain environments where the number of targets in the entire domain is known *a priori*. In fact, it is known that there may be one target of each type in the environment but the exact location of the targets is unknown. Therefore, in this case, if a target is detected in a cell, it not only changes the probability of existence of the target in that cell (increases the value), but also changes the probability of existence of the target in the other cells (decreases the value).

2.2.1 Single Type of Target

In this case, it is known that there is at most one target in the environment but the exact location of the target is unknown. The probability map contains the probability of existence of the target in each cell. The probability map is initialized by the *a priori* knowledge about the environment. If it is known that the target can only exist in some parts of the environment (the uncertainty region), the initial probability map is constructed such that $p_q = 0$ for all cells q outside that particular area. The probability p_q for a cell q inside the uncertainty region is also assigned based on the *a priori* information. If there is no such information, the probability should be uniformly distributed between all cells which means it is equally probable for the target to be in any cell in the environment. In construction of initial probability map, it is important to make sure that the total probability of existence of the target in the environment is less than or equal to one, i.e. $\sum_{\forall q \in Q} p_q = 1$, where equality holds if we certainly know that there is a target in the environment, but we do not know its exact location.

2.2.1.1 Sensor with Single-Cell Footprint

In the sixth scenario, it is known that there is at most one target in the environment and the sensor has a single-cell footprint which can detect a target that resides in its current cell. Event E_q is the event that the target is in the cell q and event D_q is the event that the target is detected in the cell q. The probabilities of *true positive* and *false positive* measurement of sensors are assumed to be $\gamma = P(D_q|E_q)$ and $\varepsilon = P(D_q|\bar{E}_q)$ respectively, where γ is the probability of detecting the target and ε is the probability of reporting existence of the target while it does not really exist. These two parameters are specification of sensors and assumed to be known *a* priori. When a mobile sensor enters a cell, it measures the cell which has two possible outputs; the target is in the cell ($I = 1$) or the target is not in the cell ($I = 0$). When the sensor has not detected a target, the probability of existence of the target in the cell can be updated as follows, using the Bayes' Rule

$$P(E_q|\bar{D}_q) = \frac{P(E_q)P(\bar{D}_q|E_q)}{P(\bar{D}_q)} = \frac{P(E_q)\bar{\gamma}}{P(\bar{D}_q)}$$

Similarly, when a sensor has detected a target, the posterior probability of existence of the target in the cell can be computed as follows

$$P(E_q|D_q) = \frac{P(E_q)P(D_q|E_q)}{P(D_q)} = \frac{P(E_q)\gamma}{P(D_q)}$$

Using the law of total probability, $P(D_q)$ and $P(\bar{D}_q)$ can be found as follows

$$P(D_q) = P(D_q|E_q)P(E_q) + P(D_q|\bar{E}_q)P(\bar{E}_q) = \gamma\, \mathrm{p_q} + \varepsilon(1 - p_q)$$

and

$$P(\bar{D}_q) = 1 - P(D_q) = \bar{\gamma}\, p_q + \bar{\varepsilon}(1 - p_q)$$

Therefore, we can update the probability of existence of the target in that cell, based on the output of the sensor as follows

$$p_{\mathrm{q}}^{*}=\frac{p_q\cdot\bar{\gamma}}{\bar{\gamma}\,p_q+\bar{\varepsilon}\left(1-p_q\right)}\left(1-\mathrm{I}\right)+\frac{p_q\cdot\gamma}{\gamma\,p_q+\varepsilon\left(1-p_q\right)}\,\mathrm{I} \tag{2-17}$$

where p_q and p_q^* are the probability of existence of a target in the cell q before and after the visit respectively. Since $\gamma \geq 0.5$ and $\varepsilon \leq 0.5$, it is easy to verify that $p_q^* \geq p_q$ when the target is detected in the cell q, i.e. $I = 1$, and $p_q^* \leq p_q$ when the target is not detected in the cell q, i.e. $I = 0$.

When a mobile sensor visits a cell, not only the probability of existence of the target in that cell changes, but also the probability of existence of the target in the other cells changes. If the target has not been detected by the sensor in the cell q, the posterior probability of existence of the target in the cell $q' \neq q$ is as follows

$$p_{q'}^{*}=P\left(E_{q'}\middle|\overline{D}_q\right)=\frac{P\left(E_{q'}\right)P\left(\overline{D}_q|E_{q'}\right)}{P\left(\overline{D}_q\right)}=\frac{P\left(E_{q'}\right)\bar{\varepsilon}}{P\left(\overline{D}_q\right)} \tag{2-18}$$

where in deriving the above equation, we used the fact that existence of the target in the cell q' means there in no target in the cell q, thus $P\left(\overline{D}_q|E_{q'}\right)=P\left(\overline{D}_q|\bar{E}_q\right)=\bar{\varepsilon}$. Similarly, when the target has been detected by the sensor in the cell q, the posterior probability of existence of the target in the cell $q' \neq q$ is as follows

$$p_{q'}^{*}=P\left(E_{q'}\middle|D_q\right)=\frac{P\left(E_{q'}\right)P\left(D_q|E_{q'}\right)}{P\left(D_q\right)}=\frac{P\left(E_{q'}\right)\varepsilon}{P\left(D_q\right)} \tag{2-19}$$

where in deriving the above equation, we also used the fact that existence of the target in the cell q' means there in no target in the cell q, thus $P\left(D_q|E_{q'}\right)=P\left(D_q|\bar{E}_q\right)=\varepsilon$.

Therefore, the mobile sensor modifies the probability map by updating the probability of all cells in the environment, i.e. $\forall q' \in \mathrm{Q}$, using the following *Probability Map Updating Rule*

$$p^*_{q'} = \frac{p_{q'} \cdot \left(\bar{\gamma}\delta_{qq'} + \bar{\varepsilon}(1-\delta_{qq'})\right)}{\bar{\gamma}\, p_q + \bar{\varepsilon}(1-p_q)}(1 - \mathrm{I}) + \frac{p_{q'} \cdot \left(\gamma\delta_{qq'} + \varepsilon(1-\delta_{qq'})\right)}{\gamma\, p_q + \varepsilon(1-p_q)}\mathrm{I} \qquad (2\text{-}20)$$

where q is the cell which has been searched by the mobile sensor. It should be noted that the total probability of existence of the target in the environment can always be found using $p = \sum_{\forall q \in Q} p_q$.

In Figure 2-2, the probability maps for the first scenario and the sixth scenario are compared at different time steps. The only difference between two scenarios is the *a priori* information about the number of targets in the environment. In the first scenario, the number of targets in the environment is unknown *a priori* while in the sixth scenario it is known that there is at most one target in the entire environment. The left panels show the first scenario and the right panels shows the sixth scenario. Initial probability maps are the same for both scenarios (panel a and panel d). It is assumed that the uncertainty region of the target is a square area and the probability of existence of the target in other parts of the environment is zero. In the left panels, when the target is not detected in a cell, the probability of existence of the target in that cell decreases (panel b). When the target is detected in a cell, the probability of existence of the target in that cell increases (panel c). In both cases, the probability of existence of the target in other cells remain unchanged. In the right panels, when the target is not detected in a cell, the probability of existence of the target in that cell decreases and the probability of existence of the target in other cells increases (panel e). When the target is detected in a cell, the probability of existence of the target in that cell increases and the probability of existence of the target in other cells decreases (panel f).

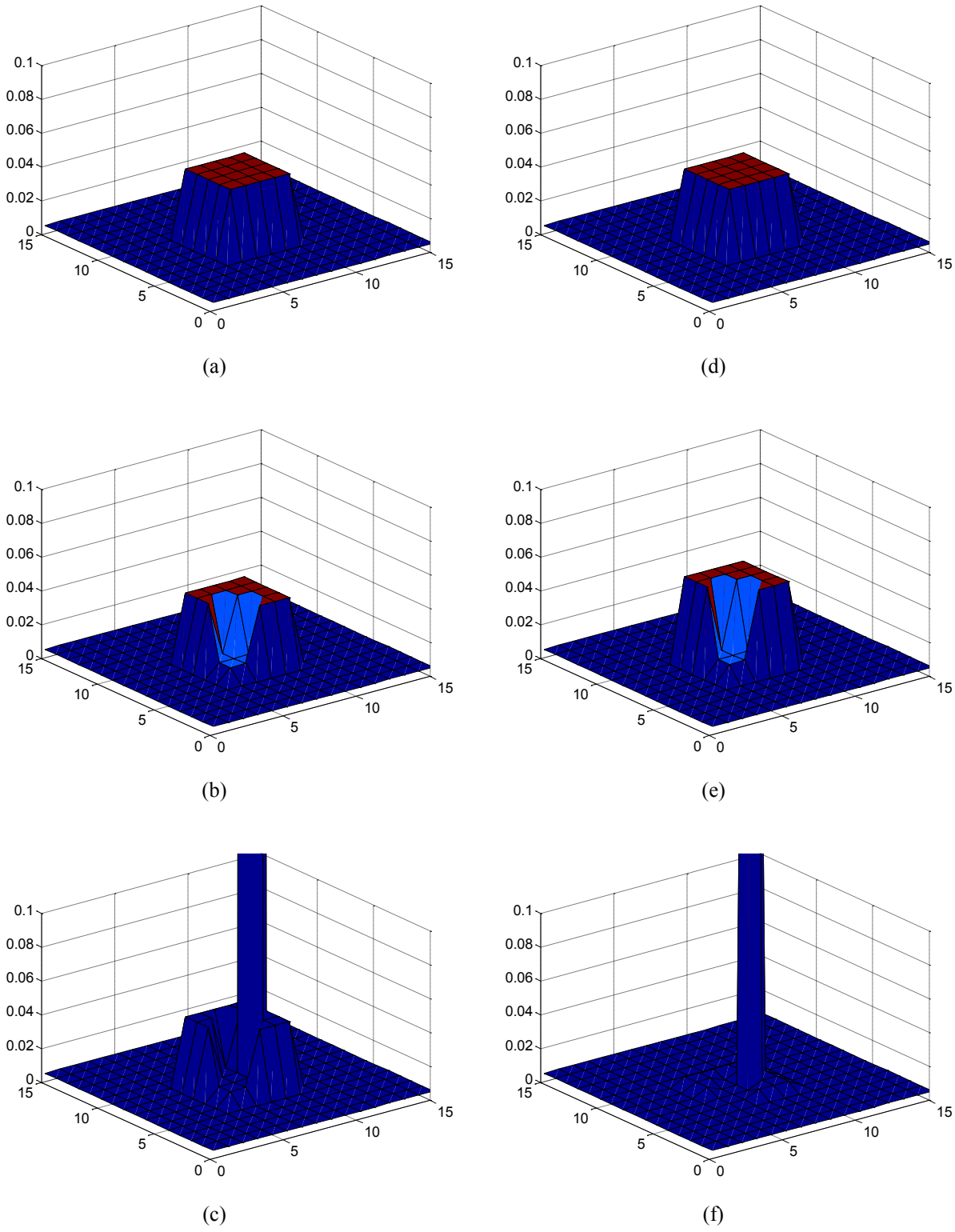


Figure 2-2. The probability maps at different time steps.
Left panels: the first scenario
Right panels: the sixth scenario

Although by using (2-20), one can update probability map after each measurement, it can be very time consuming to do so, especially when the number of cells in the environment is large. We use the concept of relative probability to resolve this issue. We define r_q as the relative probability of existence of the target in the cell q. The probability of existence of the target in the cell q at each time can be found by the following equation

$$p_q = \frac{r_q}{\sum_{\forall q' \in Q} r_{q'}} \tag{2-21}$$

At the beginning, the relative probability of existence of the target in each cell is initialized by $r_q = p_q$.

By defining $\Delta = \sum_{\forall q' \in Q} r_{q'}$, instead of storing and updating the probability map, the relative probability map and Δ are updated and stored. When a cell is visited by a mobile sensor, only the relative probability of existence of the targets in that cell is changed. By definition

$$\frac{r_q}{r_{q'}} = \frac{p_q}{p_{q'}}$$

If q is the cell which has been searched by the mobile sensor, the posterior relative probability of existence of the target in the cell q is as follows

$$r_q^* = \frac{p_q^*}{p_{q'}^*}.r_{q'}^* = \frac{\frac{\bar{\gamma}}{\bar{\gamma}\,p_q + \bar{\varepsilon}(1-p_q)}(1-\mathrm{I}) + \frac{\gamma}{\gamma\,p_q + \varepsilon(1-p_q)}\mathrm{I}}{\frac{\bar{\varepsilon}}{\bar{\gamma}\,p_q + \bar{\varepsilon}(1-p_q)}(1-\mathrm{I}) + \frac{\varepsilon}{\gamma\,p_q + \varepsilon(1-p_q)}\mathrm{I}}.\frac{p_q}{p_{q'}}.r_{q'}^*$$

$$= \left(\frac{\bar{\gamma}}{\bar{\varepsilon}}(1-\mathrm{I}) + \frac{\gamma}{\varepsilon}\mathrm{I}\right).\frac{p_q}{p_{q'}}.r_{q'}^{*j}$$

However, the relative probability of the other cells does not change after the visit, i.e. $r_{q'}^* = r_{q'}$. Therefore

$$r_q^* = \left(\frac{\bar{\gamma}}{\bar{\varepsilon}}(1-\mathrm{I}) + \frac{\gamma}{\varepsilon}\mathrm{I}\right).\frac{p_q}{p_{q'}}.r_{q'} = \left(\frac{\bar{\gamma}}{\bar{\varepsilon}}(1-\mathrm{I}) + \frac{\gamma}{\varepsilon}\mathrm{I}\right).r_q \tag{2-22}$$

We must also update Δ by adding the new value of relative probability of q and subtracting its old value as follows

$$\Delta^* = \Delta - r_q + r_q^* \tag{2-23}$$

Therefore, after a mobile sensor visits the cell q, we only need to update the relative probability of the cell q and the total relative probability of existence of the target, using (2-22) and (2-23), respectively.

If the number of cells in the environment is equal to N_q, updating the probability map requires updating N_q values, while updating the relative probability map and the total relative probability of existence of the target only requires updating 2 values.

2.2.1.2 Sensor with Multiple-Cell Footprint

In the previous section, we assumed that the footprint of sensor is only a single cell. In this section, we extend the results for the case that the footprint of sensor consists of multiple cells.

2.2.1.2.1 Multi-Sensor Model

In the seventh scenario, it is known that there is at most one target in the environment and the sensor has multiple-cell footprint which can detect the target in each cell separately. In this case, we can replace the sensor with an array of several virtual sensors with one cell footprint and use (2-20) to update the probability map or (2-22) and (2-23) to update the relative probability map and the total relative probability, respectively. It should be noted that each of these virtual sensors needs to update the entire probability map. Indeed, using these virtual sensors, there is multiple sensory information about all cells in the domain. In this situation, a sensor fusion algorithm may be used to combine the information derived from these different virtual sensors to achieve a better

result which is beyond the scope of this study [113]. The simplest solution to this problem is that the probability update is performed for all virtual sensors, in a pre-specified order.

A more realistic model for γ and ε can be used where they are functions of the distance between the sensor and the cell being observed, i.e. $r = \|\mathbf{q} - \mathbf{p}\|$. Parameters $\gamma(r)$ and $\varepsilon(r)$ must satisfy the properties described in section 2.1.1.2.1.

We define Ω as the collection of cells inside the sensor footprint and N^{Ω} as the number of cells in Ω. Therefore, for any cell inside the sensor footprint, i.e. $\forall q \in \Omega$, the mobile sensor modifies the probability map by updating the probability of all cells in the environment, i.e. $\forall q' \in Q$, using the following *Probability Map Updating Rule*

$$p_q^* = \frac{p_{q'}.\left(\bar{\gamma}(\mathrm{r})\delta_{qq'} + \bar{\varepsilon}(r)(1-\delta_{qq'})\right)}{\bar{\gamma}(\mathrm{r})\, p_q + \bar{\varepsilon}(\mathrm{r})\left(1-p_q\right)}\left(1 - \mathrm{I}_q\right) + \frac{p_{q'}.\left(\gamma(r)\delta_{qq'} + \varepsilon(r)(1-\delta_{qq'})\right)}{\gamma(\mathrm{r})\, p_q + \varepsilon(\mathrm{r})\left(1-p_q\right)}\mathrm{I}_q \qquad (2\text{-}24)$$

where I_q is the output of sensor corresponding to the cell q. This equation is the same as (2-20) where γ and ε are function of distance between the sensor and the cell.

If we define r_q as the relative probability of existence of the target in the cell q and Δ as the total relative probability, instead of storing and updating the probability map, we can store and update the relative probability map and the value of the total relative probability. In this case, for any cell inside the sensor footprint, i.e. $\forall q \in \Omega$, the mobile sensor modifies the relative probability map by updating the relative probability of the cell q, using the following rule

$$r_q^* = \left(\frac{\bar{\gamma}(\mathrm{r})}{\bar{\varepsilon}(\mathrm{r})}\left(1 - \mathrm{I}_\mathrm{q}\right) + \frac{\gamma(\mathrm{r})}{\varepsilon(\mathrm{r})}\mathrm{I}_\mathrm{q}\right).\, r_q \qquad (2\text{-}25)$$

and also modifies the total relative probability, using

$$\Delta^* = \Delta - r_q + r_q^* \qquad (2\text{-}26)$$

Equation (2-25) is the same as (2-22) where γ and ε are function of distance between the sensor and the cell.

If the number of the cells in the environment is equal to N_q, updating the probability map requires updating $N_q \times N^{\Omega}$ values, while updating the relative probability map and the total relative probability only requires updating $2N^{\Omega}$ values.

2.2.1.2.2 Single-Sensor Model

In the eighth scenario, it is known that there is at most one target in the environment. The footprint of sensor consists of multiple cells and the sensor can only report the existence or non-existence of the target in its entire footprint. We define Ω as the collection of cells inside the sensor footprint and N^{Ω} as the number of cells in Ω. Event E_{q_i} is the event that a target is in the cell $q_i \in \Omega$ and event D_{Ω} is the event that a target is detected by the mobile sensor. Parameters $\gamma_i = P\left(D_{\Omega}\middle|E_{q_i}\right)$ and $\varepsilon = P(D_{\Omega}|\bar{E}_{\Omega})$ are specification of sensor and must be known *a priori* where $E_{\Omega} = \cup_{\forall\, q_i \in \Omega} E_{q_i}$ is the event that there is a target in the sensor footprint. In general, γ_i can be a function of the distance between the location of the sensor and the position of the cell q_i.

When the sensor has not detected a target, the probability of existence of the target in the cell $q_i \in \Omega$ can be updated as follows

$$P\left(E_{q_i}\middle|\bar{D}_{\Omega}\right) = \frac{P\left(E_{q_i}\right)P\left(\bar{D}_{\Omega}|E_{q_i}\right)}{P(\bar{D}_{\Omega})}$$

$$= \frac{P\left(E_{q_i}\right)P\left(\bar{D}_{\Omega}|E_{q_i}\right)}{\sum_{\forall q_j \in \Omega}\left(P\left(\bar{D}_{\Omega}|E_{q_j}\right).P\left(E_{q_j}\right)\right) + P(\bar{D}_{\Omega}|\bar{E}_{\Omega}).P(\bar{E}_{\Omega})}$$

$$= \frac{p_{q_i}.\bar{\gamma}_i}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.p_{q_j}\right) + \bar{\varepsilon}.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}$$

Similarly, when a sensor has detected a target, the posterior probability of existence of the target in the cell $q_i \in \Omega$ can be computed as follows

$$P(E_{q_i}|D_\Omega) = \frac{P(E_{q_i}).P(D_\Omega|E_{q_i})}{P(D_\Omega)}$$

$$= \frac{P(E_{q_i}).P(D_\Omega|E_{q_i})}{\sum_{\forall q_j \in \Omega}\left(P\left(D_\Omega|E_{q_j}\right).P\left(E_{q_j}\right)\right) + P(D_\Omega|\bar{E}_\Omega).P(\bar{E}_\Omega)}$$

$$= \frac{p_{q_i}.\gamma_i}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.p_{q_j}\right) + \varepsilon.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}$$

When a mobile sensor visits a cell, the probability of existence of the target in the other cells is also changed. If the target has not been detected by the sensor in its footprint, the posterior probability of existence of the target in the cell $q' \notin \Omega$ is as follows

$$P(E_{q'}|\bar{D}_\Omega) = \frac{P(E_{q'}).P(\bar{D}_\Omega|E_{q'})}{P(\bar{D}_\Omega)}$$

$$= \frac{P(E_{q'}).P(\bar{D}_\Omega|\bar{E}_\Omega)}{\sum_{\forall q_j \in \Omega}\left(P\left(\bar{D}_\Omega|E_{q_j}\right).P\left(E_{q_j}\right)\right) + P(\bar{D}_\Omega|\bar{E}_\Omega).P(\bar{E}_\Omega)}$$

$$= \frac{p_{q'}.\bar{\varepsilon}}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.p_{q_j}\right) + \bar{\varepsilon}.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}$$

where in deriving the above equation, we used the fact that existence of the target in the cell q' means there in no target in the sensor footprint, thus $P(\bar{D}_\Omega|E_{q'}) = P(\bar{D}_\Omega|\bar{E}_\Omega) = \bar{\varepsilon}$. Similarly, when the target has been detected by the sensor in its footprint, the posterior probability of existence of the target in the cell $q' \notin \Omega$ is as follows

$$P(E_{q'}|D_\Omega) = \frac{P(E_{q'}).P(D_\Omega|E_{q'})}{P(D_\Omega)}$$

$$= \frac{P\left(E_{q'}\right).P(D_\Omega|\bar{E}_\Omega)}{\sum_{\forall q_j \in \Omega}\left(P\left(D_\Omega|E_{q_j}\right).P\left(E_{q_j}\right)\right) + P(D_\Omega|\bar{E}_\Omega).P(\bar{E}_\Omega)}$$

$$= \frac{p_{q'}.\varepsilon}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.p_{q_j}\right) + \varepsilon.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}$$

where in deriving the above equation, we also used the fact that existence of the target in the cell q' means there in no target in the sensor footprint, thus $P\left(D_\Omega|E_{q'}\right) = P(D_\Omega|\bar{E}_\Omega) = \varepsilon$.

Therefore, the mobile sensor modifies the probability map by updating the probability of all cells in the environment, i.e. $\forall q' \in \mathrm{Q}$, using the following *Probability Map Updating Rule*

$$p^*_{q'} = \begin{cases} \dfrac{p_{q'}.\bar{\gamma}_i.(I-1)}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.p_{q_j}\right) + \bar{\varepsilon}.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)} + \dfrac{p_{q'}.\gamma_i.I}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.p_{q_j}\right) + \varepsilon.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}, & \forall q' \in \Omega \\ \dfrac{\mathrm{p}_{q'}.\bar{\varepsilon}.(I-1)}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.p_{q_j}\right) + \bar{\varepsilon}.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)} + \dfrac{\mathrm{p}_{q'}.\varepsilon.I}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.p_{q_j}\right) + \varepsilon.\left(1 - \sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}, & \forall q' \notin \Omega \end{cases} \quad (2\text{-}27)$$

If we define r_q as the relative probability of existence of the target in the cell q and Δ as the total relative probability, Instead of storing and updating the probability map, we can store and update the relative probability map and the total relative probability. If $q_i \in \Omega$ and $q' \notin \Omega$, the posterior relative probability of existence of the target in the cell q_i is as follows

$$r^*_{q_i} = \frac{p^*_{q_i}}{p^*_{q'}}.r^*_{q'}$$

$$= \frac{\dfrac{\bar{\gamma}_i.(I-1)}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.\mathrm{p}_{q_j}\right) + \bar{\varepsilon}.\left(1 - \sum_{\forall q_j \in \Omega}\left(\mathrm{p}_{q_j}\right)\right)} + \dfrac{\gamma_i.I}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.\mathrm{p}_{q_j}\right) + \varepsilon.\left(1 - \sum_{\forall q_j \in \Omega}\left(\mathrm{p}_{q_j}\right)\right)}}{\dfrac{\bar{\varepsilon}.(I-1)}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.\mathrm{p}_{q_j}\right) + \bar{\varepsilon}.\left(1 - \sum_{\forall q_j \in \Omega}\left(\mathrm{p}_{q_j}\right)\right)} + \dfrac{\varepsilon.I}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.\mathrm{p}_{q_j}\right) + \varepsilon.\left(1 - \sum_{\forall q_j \in \Omega}\left(\mathrm{p}_{q_j}\right)\right)}}.\frac{p_q}{p_{q'}}.r^*_{q'}$$

The relative probability of the cells outside the sensor footprint does not change after the visit, i.e. $r^*_{q'} = r_{q'}$. Therefore

$$r^*_{q_i}$$

$$= \frac{\frac{\bar{\gamma}_i.(I-1)}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.p_{q_j}\right)+\bar{\varepsilon}.\left(1-\sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)} + \frac{\gamma_i.I}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.p_{q_j}\right)+\varepsilon.\left(1-\sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}}{\frac{\bar{\varepsilon}.(I-1)}{\sum_{\forall q_j \in \Omega}\left(\bar{\gamma}_i.p_{q_j}\right)+\bar{\varepsilon}.\left(1-\sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)} + \frac{\varepsilon.I}{\sum_{\forall q_j \in \Omega}\left(\gamma_i.p_{q_j}\right)+\varepsilon.\left(1-\sum_{\forall q_j \in \Omega}\left(p_{q_j}\right)\right)}} . \frac{p_q}{p_{q'}} . r_{q'}$$

$$= \left(\frac{\bar{\gamma}_i}{\bar{\varepsilon}}(1-\mathrm{I}) + \frac{\gamma_i}{\varepsilon}\mathrm{I}\right).r_q \qquad (2\text{-}28)$$

We must also update Δ by adding the new value of relative probability of all cells inside the sensor footprint and subtracting the old values as follows

$$\Delta^* = \Delta - \sum_{\forall q_j \in \Omega}\left(r_{q_j}\right) + \sum_{\forall q_j \in \Omega}\left(r^*_{q_j}\right) \qquad (2\text{-}29)$$

Therefore, after each visit of the mobile sensor, we only need to update the relative probability of cells inside the sensor footprint and the total relative probability of existence of the target, using (2-28) and (2-29), respectively.

Remark 1: In many cases, parameter $\gamma_i = P\left(D_\Omega \middle| E_{q_i}\right)$ is not known for every cell q_i inside the sensor footprint. In fact, often the only available information about the sensor is the probability of detecting a target given that the target is in the sensor footprint, i.e. $\gamma = P(D_\Omega|E_\Omega)$. In this case, in (2-27), (2-28) and (2-29), γ_i must be replaced by γ for any i.

Remark 2: In this section (2.2.1), we discussed the case where there is at most one target in the whole environment. All results are readily extendable to the case with multiple targets and disjoint uncertainty regions. In this case, the mobile sensor uses the probability updating rule only for the cells inside the uncertainty region of each target.

2.2.2 Multiple Types of Targets

In this section, we are interested in extending the results from the previous section for the case with different distinguishable targets. In this case, there is at most one target of each type in the entire environment. The sensors are able to detect different types of targets. The probability map contains the probability of existence of each target in each cell. The probability p_q^j is the probability of existence of the target j in the cell q, and the total probability of existence of the target in the whole environment is $p^j = \sum_{\forall q \in Q} p_q^j$. At the beginning, the probability map is constructed based on the *a priori* knowledge about the position of the targets. If it is known that the target j can only exist in some part of environment (its uncertainty region), the initial probability map is constructed such that $p_q^j = 0$ for all cells q outside of that particular area. The initial probability p_q^j for a cell q inside the uncertainty region of j is also assigned based on the *a priori* information. If there is no such information, the initial probability should be uniformly distributed between all cells. In construction of initial probability map, it is important to make sure that the total probability of existence of any target in the environment and the total probability of existence of all targets in any cell are less than or equal to one, i.e. $\sum_{\forall q \in Q} p_q^j \leq 1, \forall j \in [1, m]$ and $\sum_{j=1}^{m} p_q^j \leq 1, \forall q \in Q$. The probability $p^j = 1$ means that the target j definitely exists in the environment, however, its exact position is unknown.

2.2.2.1 Sensor with Single-Cell Footprint

In the ninth scenario, there are different distinguishable targets but it is known that there is at most one target of each type in the entire environment. The sensor has a single-cell footprint which can detect a target that resides in its current cell. The agents are equipped with imperfect sensors

with categorical distribution. We define E_q^i as the event that the target i is in the cell q and D_q^i as the event that the target i is detected in the cell q.

Parameter η_j^i is defined as the probability of detecting target i given the actual target is j, where $i, j \in [0, m]$ and m is the number of possible targets, i.e. $\eta_j^i = P(D_q^i | E_q^j)$. Index zero corresponds to the situation that there is no target. Therefore, η_j^0 is the probability of detecting no target given target j exists in the cell, and η_0^j is the probability of detecting target j while there is no target in the cell. It is expected that the probability of *true positive* measurement of all targets is greater than 0.5, i.e. $\eta_j^j > 0.5$ for $\forall j \in [0, m]$. It is also expected that $\sum_{i=0}^{m} \eta_j^i = 1$ for $\forall j \in [0, m]$. The probability transition matrix $\mathbf{P}[\mathrm{P_{i,j}} = \eta_j^i]$ is obtained from technical specifications on the sensors, and is considered to be known *a priori*.

Random variable T is defined to be equal to the output of the sensor, i.e. $T = i$ means that the sensor has detected target i in the cell. Again $T = 0$ means no target has been detected in the cell. If a mobile sensor visits a cell and detects the target i in that cell, i.e. $T = i$, the probability of existence of target j in that cell can be updated as follows

$$p_q^{*j} = P(E_q^j | D_q^i) = \frac{p_q^j \,.\, \eta_j^i}{P(D_q^i)}$$

Using the law of total probability

$$P\left(D_q^i\right) = \sum_{k=0}^{m} \eta_k^i . p_q^k \tag{2-30}$$

where $p_q^0 = 1 - p_q = 1 - \sum_{k=1}^{m} p_q^k$ is the probability that no target exists in the cell. Therefore, when the cell q is searched by a sensor, the posterior probability of existence of target j in that cell can be updated by using the following equation

$$p_q^{*j} = \sum_{i=0}^{m} \left(\frac{p_q^j . \eta_j^i}{\sum_{k=0}^{m} \eta_k^i . p_q^k} . \delta_{iT}\right) \tag{2-31}$$

When the probability of false detection of a target in a cell is independent of the real target in that cell, i.e. $\eta_j^i = \varepsilon_i,\ \forall j \neq i$, (2-39) can be further simplified as follows

$$P\left(D_q^i\right) = \sum_{\substack{k=0 \\ k\neq i}}^{m} \left(\varepsilon_i . p_q^k\right) + \gamma_i . p_q^i = \varepsilon_i . \left(1 - p_q^i\right) + \gamma_i . p_q^i \tag{2-32}$$

where $\gamma_i = \eta_i^i$. Therefore, (2-40) can be written as follows

$$p_q^{*j} = \sum_{i=0}^{m} \left(\frac{p_q^j . \gamma_j . \delta_{ij} + p_q^j . \varepsilon_j . (1-\delta_{ij})}{\varepsilon_i . (1-p_q^i) + \gamma_i . p_q^i} . \delta_{iT}\right) \tag{2-33}$$

Since we expect $\gamma_i \geq 0.5$ and $\varepsilon_i \leq 0.5$, it is easy to verify that $p_q^{*j} \geq p_q^j$ when the j^{th} target is detected in the cell q, i.e. $\delta_{jT} = 1$, and $p_q^{*j} \leq p_q^j$ when the j^{th} target is not detected in the cell q, i.e. $\delta_{jT} = 0$.

When a mobile sensor visits a cell, not only the probability of existence of the targets in that cell changes, but the probability of existence of the targets in the other cells also changes. Given the target i has been detected by the sensor in the cell q, the posterior probability of existence of the target j in the cell $q' \neq q$ is as follows

$$p_{q'}^{*j} = P(E_{q'}^j | D_q^i) = \frac{p_{q'}^j . P(D_q^i | E_{q'}^j)}{P(D_q^i)} \tag{2-34}$$

By using the fact that existence of the target j in the cell q' means that the target cannot exist in the cell q

$$P\left(D_q^i \middle| E_{q'}^j\right) = P\left(D_q^i \middle| \bar{E}_q^j\right) = P\left(D_q^i \middle| \cup_{\forall k\neq j} E_q^k\right)$$

$$= \frac{P\left(D_q^i \cap \left(\cup_{\forall k\neq j} E_q^k\right)\right)}{P\left(\cup_{\forall k\neq j} E_q^k\right)} = \frac{P\left(\cup_{\forall k\neq j}\left(D_q^i \cap E_q^k\right)\right)}{P\left(\cup_{\forall k\neq j} E_q^k\right)}$$

$$= \frac{\sum_{\forall k \neq j} P\left(D_q^i \cap E_q^k\right)}{\sum_{\forall k \neq j} P\left(E_q^k\right)} = \frac{\sum_{\forall k \neq j} \left(P\left(D_q^i | E_q^k\right) P\left(E_q^k\right)\right)}{1 - P\left(E_q^j\right)}$$

$$= \frac{\sum_{\forall k \neq j} \left(\eta_k^i . p_q^k\right)}{1 - p_q^j} \tag{2-35}$$

and

$$P\left(D_q^i\right) = \sum_{\forall k} \left(P\left(D_q^i | E_q^k\right) P\left(E_q^k\right) \right) = \sum_{\forall k} \left(\eta_k^i . p_q^k\right) \tag{2-36}$$

Therefore, the mobile sensor can modify the probability map by updating the probability of existence of all targets, i.e. $\forall j \in [1, m]$, in all cells in the environment, i.e. $\forall q' \in Q$, using the following *Probability Map Updating Rule*

$$P_{q'}^{*j} = \left(\sum_{i=0}^{m} \left(\frac{p_{q'}^j . \eta_j^i}{\sum_{k=0}^{m} \eta_k^i . p_q^k} . \delta_{iT} \right) \right) . \delta_{qq'} + \left(\sum_{i=0}^{m} \left(\frac{p_{q'}^j . \left(\sum_{\forall k \neq j} \left(\eta_k^i . p_q^k\right)\right)}{\left(1 - p_q^j\right) . \left(\sum_{k=0}^{m} \eta_k^i . p_q^k\right)} . \delta_{iT} \right) \right) . \left(1 - \delta_{qq'}\right) \tag{2-37}$$

where q is the cell which has been searched by the mobile sensor. It should be noted that the total probability of existence of any target in the environment can always be found using $P^j = \sum_{\forall q \in Q} P_{q'}^j$.

Remark 1: Equation (2-20) is a special case of (2-46), where $m = 1$, $\eta_1^1 = \gamma$, and $\eta_0^1 = \varepsilon$.

Although by using (2-46), one can update the probability map after each measurement, it can be a very time consuming, especially when the number of cells in the environment is large. We use the concept of relative probability to resolve this issue. We define r_q^j as the relative probability of existence of j^{th} target the cell q. The probability of existence of target j in the cell q at each time can be found by the following equation

$$p_q^j = \frac{r_q^j}{\Sigma_{\forall q' \in Q} r_{q'}^j} \tag{2-38}$$

The initial relative probability of existence of each target in each cell is initialized by $r_q^j = p_q^j$.

By defining $\Delta^j = \sum_{\forall q' \in Q} r_{q'}^j$ as the total relative probability of j^{th} target, instead of storing and updating the probability map, the relative probability map and Δ^j are stored and updated. When a cell is visited by a mobile sensor, only the relative probability of existence of the targets in that cell is changed. By definition

$$\frac{r_q^j}{r_{q'}^j} = \frac{p_q^j}{p_{q'}^j}$$

If q is the cell which has been searched by the mobile sensor, the posterior relative probability of existence of target j in the cell q is as follows

$$r_q^{*j} = \frac{p_q^{*j}}{p_{q'}^{*j}} . r_{q'}^{*j} = \frac{\sum_{i=0}^{m} \left(\frac{\eta_j^i}{\sum_{k=0}^{m} \eta_k^i . p_q^k} . \delta_{iT} \right)}{\sum_{i=0}^{m} \left(\frac{\left(\sum_{\forall k \neq j} \left(\eta_k^i . p_q^k \right) \right)}{\left(1 - p_q^j \right) . \left(\sum_{k=0}^{m} \eta_k^i . p_q^k \right)} . \delta_{iT} \right)} . \frac{p_q^j}{p_{q'}^j} . r_{q'}^{*j}$$

However, the relative probability of the other cells does not change after the visit, i.e. $r_{q'}^{*j} = r_{q'}^j$. Therefore

$$r_q^{*j} = \frac{\sum_{i=0}^{m} \left(\frac{\eta_j^i}{\sum_{k=0}^{m} \eta_k^i . p_q^k} . \delta_{iT} \right)}{\sum_{i=0}^{m} \left(\frac{\left(\sum_{\forall k \neq j} \left(\eta_k^i . p_q^k \right) \right)}{\left(1 - p_q^j \right) . \left(\sum_{k=0}^{m} \eta_k^i . p_q^k \right)} . \delta_{iT} \right)} . \frac{p_q^j}{p_{q'}^j} . r_{q'}^j$$

$$= \frac{\sum_{i=0}^{m} \left(\frac{\eta_j^i}{\sum_{k=0}^{m} \eta_k^i . p_q^k} . \delta_{iT} \right)}{\sum_{i=0}^{m} \left(\frac{\left(\sum_{\forall k \neq j} \left(\eta_k^i . p_q^k \right) \right)}{\left(1 - p_q^j \right) . \left(\sum_{k=0}^{m} \eta_k^i . p_q^k \right)} . \delta_{iT} \right)} . r_q^j$$

$$= \frac{\left(1 - p_q^j\right).\eta_j^i}{\sum_{i=0}^{m}\left(\left(\sum_{\forall k \neq j}\left(\eta_k^i.p_q^k\right)\right).\delta_{iT}\right)}.r_q^j$$

$$= \frac{\left(1 - p_q^j\right).\eta_j^i}{\sum_{i=0}^{m}\left(\left(\eta_0^i\left(1 - \sum_{k=1}^{m} p_q^k\right) + \sum_{k=1,k\neq j}^{m}\left(\eta_k^i.p_q^k\right)\right).\delta_{iT}\right)}.r_q^j$$

$$= \frac{\left(1 - p_q^j\right).\eta_j^i}{\sum_{i=0}^{m}\left(\left(\eta_0^i + \sum_{k=1}^{m}\left(\left(\eta_k^i - \eta_0^i\right).p_q^k\right) - \eta_j^i.p_q^j\right).\delta_{iT}\right)}.r_q^j$$

$$= \frac{\left(1 - \frac{r_q^j}{\Delta^j}\right).\eta_j^i}{\sum_{i=0}^{m}\left(\left(\eta_0^i + \sum_{k=1}^{m}\left(\left(\eta_k^i - \eta_0^i\right)\frac{r_q^k}{\Delta^k}\right) - \eta_j^i\frac{r_q^j}{\Delta^j}\right).\delta_{iT}\right)}.r_q^j$$

$$= \frac{\left(1 - \frac{r_q^j}{\Delta^j}\right).\eta_j^i.r_q^j}{\sum_{i=0}^{m}\left(\left(\eta_0^i + \sum_{k=1}^{m}\left(\frac{\left(\eta_k^i - \eta_0^i\right).r_q^k}{\Delta^k}\right) - \frac{\eta_j^i.r_q^j}{\Delta^j}\right).\delta_{iT}\right)} \tag{2-39}$$

which is only a function of the relative probability of existence of the targets in the cell q and the total relative probability of existence of the targets in the environment. We must also update Δ^j by adding the updated value of relative probability of q and subtracting its previous value as follows

$$\Delta^{*j} = \Delta^j - r_q^j + r_q^{*j} \tag{2-40}$$

Therefore, after a mobile sensor visits the cell q, we only need to update the relative probability of all targets in the cell q, i.e. $r_q^j, \forall j \in [1, m]$, and the total relative probability of existence of all targets, i.e. $\Delta^j, \forall j \in [1, m]$, using (2-39) and (2-40), respectively.

Remark 2: Equation (2-22**)** is a special case of (2-39), where $m = 1, \eta_1^1 = \gamma$, and $\eta_0^1 = \varepsilon$.

If the number of cells in the environment is equal to N_q, updating the probability map requires updating $N_q \times m$ values, while updating the relative probability map and the total relative probability of existence of all targets only requires updating $m + 1$ values.

2.2.2.2 Sensor with Multiple-Cell Footprint

In the previous section, we assumed that the footprint of sensor is only a single cell. In this section, we extend the results for the case that the footprint of sensor consists of multiple cells.

2.2.2.2.1 Multi-Sensor Model

In the tenth scenario, there are different distinguishable targets but it is known that there is at most one target of each type in the entire environment. The sensor has multiple-cell footprint which can detect the target in each cell separately. In this case, the footprint of sensor is more than one cell and the sensor can detect the target in each cell inside its footprint separately. In fact, we can replace this sensor with an array of several sensors with one cell footprint. Each sensor, then, can use (2-37) to update the probability map or (2-39) and (2-40) to update the relative probability map and the total relative probability of all targets, respectively. It should be noted that each of these virtual sensors needs to update the entire probability map. Indeed, using these virtual sensors, there is multiple sensory information about all cells in the domain. In this situation, a sensor fusion algorithm may be used to combine the information derived from these different virtual sensors to achieve a better result which is beyond the scope of this study. The simplest solution to this problem is that the probability update is performed for all virtual sensors, in a pre-specified order.

A more general model is the one that the value of η_j^i is assumed to depend on the distance between the sensor and the cell being observed, i.e. $r = \|\mathbf{q} - \mathbf{p}\|$. Parameters $\eta_j^i(r)$ must satisfy the properties described in section 2.1.2.2.1.

We define Ω as the collection of cells inside the sensor footprint and N^{Ω} as the number of cells in Ω. Therefore, for any cell inside the sensor footprint, i.e. $\forall q \in \Omega$, the mobile sensor modifies the probability map by updating the probability of existence of all targets, i.e. $\forall j \in [1, m]$, in all cells in the environment, i.e. $\forall q' \in Q$, using the following *Probability Map Updating Rule*

$$P_{q'}^{*j} = \left(\sum_{i=0}^{m} \left(\frac{p_{q'}^{j} . \eta_j^i}{\sum_{k=0}^{m} \eta_k^i . p_q^k} . \delta_{iT}\right)\right) . \delta_{qq'} + \left(\sum_{i=0}^{m} \left(\frac{p_{q'}^{j} . \left(\sum_{\forall k \neq j} \left(\eta_k^i . p_q^k\right)\right)}{\left(1 - p_q^j\right) . \left(\sum_{k=0}^{m} \eta_k^i . p_q^k\right)} . \delta_{iT}\right)\right) . \left(1 - \delta_{qq'}\right) \tag{2-41}$$

where q is the cell which has been searched by the mobile sensor.

If we define r_q^j as the relative probability of existence of j^{th} target in the cell q and Δ^j as the total relative probability of j^{th} target, instead of storing and updating the probability map, we can store and update the relative probability map and the total relative probability of all targets. In this case, for any cell inside the sensor footprint, i.e. $\forall q \in \Omega$, the mobile sensor modifies the relative probability map by updating the relative probability of existence of all targets, i.e. $\forall j \in [1, m]$, in the cells q, using the following rule

$$r_q^{*j} = \frac{\left(1 - \frac{r_q^j}{\Delta^j}\right) . \eta_j^i . r_q^j}{\sum_{i=0}^{m} \left(\left(\eta_0^i + \sum_{k=1}^{m} \left(\frac{\left(\eta_k^i - \eta_0^i\right) . r_q^k}{\Delta^k}\right) - \frac{\eta_j^i . r_q^j}{\Delta^j}\right) . \delta_{iT}\right)} \tag{2-42}$$

and also modifies the total relative probability of all targets, using

$$\Delta^{*j} = \Delta^j - r_q^j + r_q^{*j} \tag{2-43}$$

If the number of cells in the environment is equal to N_q, updating the probability map requires updating $N_q \times m \times N^{\Omega}$ values, while updating the relative probability map and the total relative probability only requires updating $(m+1) \times N^{\Omega}$ values.

2.2.2.2.2 Single-Sensor Model

The other possible case is when the sensor can only report the existence or non-existence of an object in its entire footprint and classify it as one of known possible targets. In this case, the sensor provides very little information about the probability of existence of the targets in each cell inside its footprint which is not useful from practical point of view. Therefore, we will not discuss this model in this study.

2.3 Conclusion

In this chapter, we considered ten different scenarios for the probabilistic search in uncertain environments and developed the probability updating rule for all scenarios. It is worth to mention that the scenarios with a single target are special cases of the scenarios with multiple targets. Similarly, the scenarios with a single-cell footprint sensor are special cases of the scenarios with multiple-cell footprint sensors.

Remarks:

The size of the cell is an important parameter. It should be chosen based on the size of the environment, the size of the target, the size of the mobile agents, and computational and storage capability of the mobile sensor. Although there are some hard constraint on the size of cell, it is generally up to the engineer's experience and knowledge to choose the appropriate cell size. For example, size of the cell should be fine enough to include only one target, but making it too fine

with respect to the size of environment only increases the computational and storage requirement of the mobile agent [114].

In his chapter, we discussed a single agent search problem. However, the result are easily extendable for multi agents search problems. When multiple mobile sensors are involved in the search mission, they are required to maintain an identical probability map. Therefore, the mobile sensors must be able to communicate with each other. After each measurement, all mobile sensors transmit their outputs to the others. Then, each mobile sensor updates its probability map based on its own measurement and measurements of the other mobile sensors.

The probabilities of true positive and false positive measurements of the sensor and the size of footprint of the sensor can be functions of the time which may be used to model the sensor performance degradation over the time.

CHAPTER 3

Cooperative Search using Dynamic Programming

This chapter studies cooperative multi agent search problem using Dynamic Programming approach. First, general model of mobile agents is presented and Dynamic Programming formulation of problem is developed. Different objectives of the search problem are investigated and appropriate cognitive models for each objective are presented. Probabilistic models that discussed in the previous chapter are shown to be one of these cognitive models. A decentralized approach is used for the search mission where each mobile agent chooses its optimal action individually. To make cooperation between agents possible, two approximation methods are proposed to modify the objective function of agents and take into the account the action of other agents. This approach is then extended for the case with known communication delay between mobile agents. Each section is followed by a simulation part to evaluate the effectiveness of the presented approach.

3.1 General Model of Mobile Sensors

An appropriate model must incorporate the influence of the current control action on future states. In general, the model of mobile agents is of the form

$$x_{k+1} = f(x_k, u_k, w_k) + b_k \tag{3-1}$$

where x_k is the system state at the discrete time step k, u_k is the control input, w_k is a random variable that captures the stochastic elements in the system dynamics, and b_k is an external disturbance. The state of the system consists of the search status and the agent status. The

probability of existence of the targets and the level of the uncertainty in their location constitute the search status. The agent status is comprised of position and heading (orientation) of all agents. The agents can communicate with each other so they can form a comprehensive view of the state. However, it is possible that delayed communication makes the observed state of the system different from the actual one. In this case, the observed state can be described as follows

$$\hat{x}_k = x_k + d_k \tag{3-2}$$

where d_k is uncertainty in the measurement. It should be noted that this uncertainty is different from the measurement noise which is due to imperfections of sensors and can be captured by w_k. The control input, u_k comes from a set of possible assignments U such as: turn left, turn right, or go straight. Stochastic elements which are captured by w_k come from different sources, including unknown locations of the targets, unknown actions of other agents, and imperfect sensor information.

3.2 Dynamic Programming Formulation

The mobile sensors must choose a control signal such that it results in the best possible paths, in the sense that the team of mobile sensors identifies maximum number of targets or gather maximum information about the environment. In other words, each agent attempts to optimize the possibility of finding targets over the decision process planning horizon. This leads naturally to the idea of applying Dynamic Programming techniques [115]. Define $J_k(\boldsymbol{x}_k)$ as the "gain" at decision time step k which can represents the expected number of the targets identified by the agents as they travel from time step k to the end of the mission. Bellman's equations for this problem can be expressed as [115]

$$J_k(\boldsymbol{x}_k) = \max_{u_k \in \mathrm{U}} \left(E_{w_k}[g(\boldsymbol{x}_k, u_k, w_k) + J_{k+1}(f(\boldsymbol{x}_k, u_k, w_k) + b_k)] \right) \tag{3-3}$$

where the term $g(\boldsymbol{x}_k, u_k, w_k)$ is the *single step gain*. The optimal decisions can be found by taking the arguments of the maximization of the Dynamic Programming recursion.

3.2.1 Single-Step Gain

The first step to calculate the gain function is finding the expected value of single step gain or the gain that a vehicle will receive at one time step (specifically at time step *k*). This value can be written as

$$g(\boldsymbol{x}_k, u_k, w_k) = \lambda^k \delta_k \sigma_k + c_k \tag{3-4}$$

where σ_k is the search gain for the vehicle at time step k which can be the expected value of the number of targets detected during the mission from time step k to time step $k+1$, δ_k is the probability that the mobile sensor is operational at time k, $\lambda(0 \leq \lambda < 1)$ is the time discount factor, and c_k is the uncertainty in the cost. The search gain σ_k can be calculated by adding up the probabilities of existence of the targets (or the value of uncertainty) in all cells that the agent covers during its mission from time step k to time step $k+1$. Parameter δ_k is normally a decreasing function of time which means the probability that the agent is operational decreases as time goes on. With the time discount less than one, it is typically desirable to find the targets as soon as possible.

3.2.2 Future Gain

Assume that the mission duration is *N* time steps. Hence, an agent will make *N* decisions over the course of the mission. Thus the terminal gain of the search mission is $J_N(\boldsymbol{x}_N)$. The gain for any time step *k* is found by iterating enough times until the terminal gain is reached. This gives [115]

$$J_k(\boldsymbol{x}_k) = \max_{u_k \in \mathrm{U}} (E_{w_k}[g(\boldsymbol{x}_k, u_k, w_k) + \max_{u_{k+1} \in \mathrm{U}} (E_{w_{k+1}}[g(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1}) + \cdots$$

$$+\max_{u_{N-1}\in \mathrm{U}}\left(E_{w_{N-1}}[g(\boldsymbol{x}_{N-1},u_{N-1},w_{N-1})+J_N(f(\boldsymbol{x}_{N-1},u_{N-1},w_{N-1})+b_{N-1})]) \dots]\right) \quad (3\text{-}5)$$

However, as the dimension of the problem grows so does the computation time. The dimension of the problem is given by the possible states to be examined over the planning horizon of the entire mission. To make the problem tractable, and solvable in real-time, a rolling horizon limited look-ahead policy can be been utilized [115]. The price to pay for such approximation is a loss in performance (near optimality). This rolling horizon approximation defines a horizon of time steps T, and then replaces the value of final gain J_N with J_{k+T}. This gives [115]

$$J_k(\boldsymbol{x}_k) \cong \max_{u_k\in \mathrm{U}}(E_{w_k}[g(\boldsymbol{x}_k,u_k,w_k) + \max_{u_{k+1}\in \mathrm{U}}(E_{w_{k+1}}[g(\boldsymbol{x}_{k+1},u_{k+1},w_{k+1}) + \cdots$$

$$+\max_{u_{k+T-1}\in \mathrm{U}}\left(E_{w_{k+T-1}}[g(x_{k+T-1},u_{k+T-1},w_{k+T-1})\right.$$

$$\left.+J_{k+T}(f(\boldsymbol{x}_{k+T-1},u_{k+T-1},w_{k+T-1})+b_{k+T-1})]) \dots]\right) \quad (3\text{-}6)$$

This produces a much smaller problem space, so it has the benefit of always producing a tractable result. However, this solution is optimal with respect to the sub-problem, not in terms of the main problem. There is, therefore, a trade-off between optimality and computational complexity.

3.3 Search Objectives

Cooperative search problem may have two different objectives; gathering more information about the environment or locating more targets based on *a priori* information about the possible position of the targets. Of course, the objective of a search mission can be a combination of these two objectives. In this case, a mobile sensor updates its information about the environment while searching for the targets. Therefore, in a search problem, the first step is to define the model of the environment and the updating rule of this model based on the mobile sensor measurement. We will present different environment models and updating rules in this chapter. Each of these models

is appropriate for some problem structure and search objective. In this framework, each agent uses a "cognitive map" as its environment representation. Cognitive maps are Cartesian grids containing cells, where each cell is assigned a certain value representing the probability or the agents' belief in the corresponding region being occupied by a target or threat. An initial map of the environment, which is uncertain and incomplete, is created based on the *a priori* knowledge about the environment.

Each sensor measurement obtained during the search is a source of evidence about the state of that location. We consider a case of imperfect sensors in this study, that is, each sensor scan does not by itself provide 100% certainty about the state of the corresponding location. In the next sections, different objectives of search mission and their corresponding model of the environment are presented and the updating rule of each model is discussed.

3.3.1 Uncertainty Reduction

The objective of mission in this case is gathering more information about the environment and reducing the uncertainty about it. The environment is a bounded $L_x \times L_y$ grid area, where each position is a cell. Each cell $\mathbf{q} = (x, y)$ has an associated uncertainty value, $\zeta(\mathbf{q}, t) \in [0,1]$, representing the agents' uncertainty about the target distribution in that cell. If $\zeta(\mathbf{q}, t) = 1$, then cell $\mathbf{q}$ is a completely unknown location for the vehicles at time t. As the cell is searched repeatedly, $\zeta(\mathbf{q}, t)$ approaches 0. A cell $\mathbf{q}$ is said to be fully searched, if $\zeta(\mathbf{q}, t) \leq \zeta_0$, where ζ_0 is a threshold corresponding to a decision that the cell does not need to be searched any more. We can see that the uncertainty value associated with each cell could actually represent the undetected information in that location. The search gain σ_k can be calculated by adding up the value of uncertainty in all cells that the agent covers during its mission from time step k to time step $k+1$, i.e. $\sigma_k =$

$\sum_{\forall \mathbf{q} \in \Omega_k} \zeta(\mathbf{q}, k)$, where Ω_k is the collection of cells that the mobile sensor covers during its mission from time step k to time step $k+1$.

Each agent uses a cognitive map to store its knowledge about the uncertainties in the environment and continuously updates it using new sensor readings from its own or from other agents by communication. This kind of cognitive map is defined as an *uncertainty map* and is denoted as $\zeta_i(t)$. Each cell in the uncertainty map is initialized with a value belonging to $[\zeta_0, 1]$ to reflect the agent's *a priori* knowledge about that location. Parameter ζ_0 ($0 \leq \zeta_0 < 1$) is the threshold value that represents no uncertainty. A cell with $\zeta_i(\mathbf{q}, 0) = 1$ is a completely unknown location to the agent i and needs to be searched. A cell with $\zeta_i(\mathbf{q}, 0) = \zeta_0$ is a location with no interest for search (for example, a location in a lake would be initialized as ζ_0 if the targets are all land-based). In general, due to the information loss caused by communication failures and delays, the uncertainty map carried by different agents might be different. If we assume that the communication among the group of vehicles is reliable and the sensor information from any vehicle is available to the whole group immediately, the whole group of vehicles actually share the same uncertainty map, which is denoted as $\mathrm{Z}(t)$.

3.3.1.1 Dempster's Rule of Combination

Here, we define an *uncertainty reduction rate*, denoted as $\mu_i \in (0,1)$, to model the uncertainties and inaccuracies about the i^{th} mobile sensor. Mathematically, μ_i quantifies the belief of a sensor scan from agent i committed to reducing the uncertainty in that cell. Since the agents are identical in this study, we use μ to denote the uncertainty reduction rate for all the agents. Based on the defined sensor model and using Dempster's rule of combination [116], a visit by any agent to cell $\mathbf{q} = (x, y)$ at time t will reduce the uncertainty value associated with that cell at a rate μ, represented as

$$\zeta(\mathbf{q}, t+1) = \mu\zeta(\mathbf{q}, t) \tag{3-7}$$

It is easy to generalize that if *m* agents visit the cell at the same time, the cell's uncertainty value is updated as

$$\zeta(\mathbf{q}, t+1) = \mu^m\zeta(\mathbf{q}, t) \tag{3-8}$$

According to equation (3-7) and (3-8), it can be seen that the first scan of a cell results in the maximum reduction in uncertainty and further scans result in reduced benefit. For example, if μ =0.5, the uncertainty value of a cell (x, y) with $\zeta(x, y, 0) = 1$, changes as 1, 0.5, 0.25, 0.125, and 0.0625, if it is sequentially visited four times by possibly different vehicles. Therefore, this update rule is a simple way to track the number of useful "looks" each cell has had and captures the nature of diminishing returns with each look. This property is similar to that of the detection function used in search theory [10] and [11], where the detection function represents the probability that a search in a given cell for a specified duration of time will detect the target provided that the target is present in that cell. Furthermore, each incremental time spent in searching a cell produces a decreasing return on the probability of detection.

3.3.1.2 Entropy-based Rule

Each cell $\mathbf{q} = (x, y)$ has an associated *target probability*, $P(\mathbf{q}, t) \in [0,1]$, representing the agents' belief about the probability that a target presents in the cell $\mathbf{q}$ at time t as discussed in the previous chapter. The uncertainty associated with cell $\mathbf{q}$ can be defined as the Shannon entropy [117] of the target probability $P(\mathbf{q}, t)$

$$\zeta(\mathbf{q}, t) = H\big(\zeta(\mathbf{q}, t)\big) = -P(\mathbf{q}, t)\log_2 P(\mathbf{q}, t) - \big(1 - P(\mathbf{q}, t)\big)\log_2\big(1 - P(\mathbf{q}, t)\big) \tag{3-9}$$

In this way, the uncertainty value associated with each cell can be used to quantify how much information the vehicles have about that location at a certain time *t*. When a cell $\mathbf{q}$ has $P(\mathbf{q}, t)$=0.5, it has 1 bit of uncertainty, which indicates that the agents are completely ignorant of whether a

target is present in that cell, because the probability of a target present is equal to the probability of no target present. When a cell $\mathbf{q}$ has $P(\mathbf{q}, t)$=1 or $P(\mathbf{q}, t)$=0, it has an uncertainty of 0. In this case, the agents are completely sure about the target present or not. Thus, the uncertainty value is a measure of lack of knowledge about the existence of the target in the cell: the closer it is to zero, the greater our knowledge.

Note that the purpose of utilizing an uncertainty map is the same as the description given in the previous section, but the definition of the uncertainty is different from the definition used in that section. The entropy-based uncertainty definition used in this section is a stronger basis for quantifying the vehicles' knowledge about the target information, but it cannot track the number of visits per cell as the Dempster-Shafer based uncertainty definition used in the previous section.

In order to update the uncertainty map, the target probability, $P(\mathbf{q}, t)$ must be updated. Updating rule for $P(\mathbf{q}, t)$ depends on the environment, the type of the targets and the type of the sensor which has been discussed in chapter 2.

3.3.2 Locating the Targets

The objective of the mission in this case is to locate the maximum number of targets in the environment in the given search time. To direct the agents to achieve their objectives, we use a probability map which represents the agent's knowledge about the target distribution. The incremental map-building method introduced in chapter 2 is used to incorporate new sensor readings based on a Bayesian model that accounts for sensor errors. Therefore, in this case, the cognitive map is a probability map.

The search gain σ_k can be calculated by adding up the probabilities of existence of the targets in all cells that the agent covers during its mission from time step k to time step $k+1$, i.e. $\sigma_k =$

$\sum_{\forall i} \sum_{\forall \mathbf{q} \in \Omega_k} P(E_{\mathbf{q}}^i)$, where Ω_k is the collection of cells that the mobile sensor covers during its mission from time step k to time step $k+1$, and $P(E_{\mathbf{q}}^i)$ is the probability of existence of target i in the cell $\mathbf{q}$.

Since objective of the mission is to locate the maximum number of targets in the environment, the agents are looking for the cells with high probability of existence of the targets. However, when a target is located in a cell, it is desirable to remove it from the list of the targets in the cell to let the mobile sensors look for the other targets. We usually define a threshold for the probability and when the probability of existence of a target in a cell is above that value, the target is considered to be in that cell and, therefore, it is removed from the list of the targets in the cell. This threshold is chosen based on different factors including the sensor accuracy, the target importance, time constraints, and available search effort. It is worth to mention that by using a Bayesian updating rule, the probability never becomes one. In fact, if a sensors detects a target in a cell for several times, the probability of existence of the target in the cell gets very close to one, but never equals one. Therefore, choosing a threshold value is necessary.

3.4 Cooperative Decision Making

We are interested in the capability of mobile sensors working in a distributed unsupervised mode, i.e., the agents themselves determine where to search based on their knowledge of the environment and do not rely on external guidance. While agents can certainly search the environment without cooperation, the search can be made much more efficient by using cooperation to minimize duplicated effort where some agents may follow the same search path and waste search effort. Therefore, the key problem for multi agent cooperative search is to choose

different search paths for each individual agent such that they can simultaneously explore different areas of their environment.

In this chapter, the mission objective is to search the terrain to locate as many targets as possible. To achieve this goal, we propose a decentralized method where each agent makes a decision about its next action individually. Each agent is viewed as a self-interested decision maker. The proposed approach consists of optimizing a global objective function through autonomous agents that are capable of making individually rational decisions to optimize their own objective functions. In non-cooperative decision making, it is possible that two or more agents decide to search the same area. Although the decision of each agent may be individually optimal, but the overall gain will be less than if the agents search completely different areas addressing a team goal. In order to enable cooperation, a mechanism must be used to consider the effect of decision of other vehicles on the decision of planning vehicle. One approach is that each agent determines its own action by simultaneously choosing the path for all agents in the fleet, using a centralized planning algorithm [35]. It is typically assumed that each agent then executes its own plan. This approach is impractical due to computational complexity, especially when the limited processing ability of the moving agents is considered. The other approach is to use a negotiation mechanism [19]. The result might be sub-optimal but the computational burden is considerably less than the previous method. However, a negotiation mechanism is not applicable in the cases of limited communication bandwidth and delays.

It is desired to obtain localized objective function for each agent that aligns with the global objective function. In our approach, each agent uses a method to estimate the probability of different actions of other agents. These probabilities are utilized to modify the objective function of the planning agent to comply with the global objectives. Therefore, when the agents want to

make decisions on their next actions, they must simply optimize their own objective functions. However, if all agents optimize their own objective function, this also optimizes the global objective. To enable efficient cooperation, the planning for each agent should consider the influence of the decision of other teaming agents on its own decision. We mentioned that the overall gain of the mission will be reduced, if two or more agents search the same area of the environment. In the extreme case that all agents follow the same path, the performance of a multi agent search is the same as the performance of a single agent search. So we intend to modify the objective function of agents to prevent them from searching the same area.

It has been shown that when an agent searches an area and does not find a target in it, the probability of existence of that target in that area is reduced. Therefore, the gain (σ) of searching that area in the future will be decreased. This can prevent agents from searching the areas which have been already searched by other agents. If all agents know the future position of other agents, a similar method can be used to prevent them from searching the same areas in the future. The bandwidth of communication channel is not large enough to let agents negotiate about their actions, so they do not know the precise position of other agents in the future. However, they may be able to estimate the path of the others in the near future. Assume that each agent knows the probability of presence of other agents in each cell over the future look-ahead horizon. Then the modified search gain of the agents for the uncertainty reduction objective and the target locating objective is defined as

$$\hat{\sigma}_k = \sum_{\forall \mathbf{q} \in \Omega_k} \rho_k(\mathbf{q}).\zeta(\mathbf{q}, k) \tag{3-10}$$

and

$$\hat{\sigma}_k = \sum_{\forall i} \sum_{\forall \mathbf{q} \in \Omega_k} \rho_k(\mathbf{q}) P(E_{\mathbf{q}}^i) \tag{3-11}$$

respectively, where $\rho_k(\mathbf{q})$ is always between zero and one. This discount factor is a decreasing function of the probability that other agents also decide to search the same area in the near future. Now, we can modify the single step gain of agents in (3-4) by replacing $\sigma_k$with $\hat{\sigma}_k$, as follows

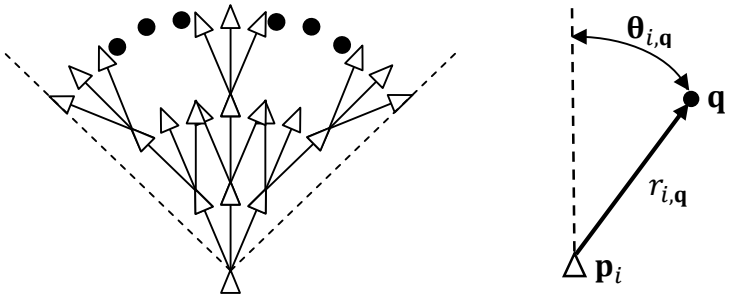
$$g(\mathbf{x}_k, u_k, w_k) = \lambda^k \delta_k \hat{\sigma}_k + c_k \tag{3-12}$$

The larger the value of g in an area, the more the probability that the planning agent decides to go to that area. Targets and agents act like opposite electrical charges. Targets attract agents while agents repel each other. Therefore, agents try to go to areas with the most probability of existence of unfound targets and the least probability of presence of other agents. It causes each agent to search the area that the other agents have not searched in the past and will not search in the future.

Exact evaluation of the probability of presence of other agents requires each agent to expand the planning tree of every other agent as shown in Figure 3.1-a. This probability then can be used to define the function ρ such that the single step gain decreases when the probability of presence of other agents increases. Although this method reduces the computational complexity of cooperative decision making compared to a centralized approach, it is still impractical when the number of vehicles or the search horizon increases. We propose two methods to estimate the planning tree of other agents. It is obvious that the performance of search mission is directly related to accuracy of these estimations.

3.4.1 Geometric Approach

In this section, we propose a geometric method to estimate the probability of different actions of other agents. The proposed function which approximately equals to the probability of presence of an agent in the near future is then used to find an appropriate discount factor ρ.



(a) (b)

Figure 3-1. The future position of a vehicle for three steps look-ahead

3.4.1.1 Geometric Estimation Method

At each decision time step k, we define Λ_τ^i as the set of all cells that the agent i may visit during the mission from the time step $k+\tau-1$ to the time step $k+\tau$. When a cell $\mathbf{q}$ is in both Λ_τ^i and $\Lambda_{\tau'}^j$, where $\tau' \leq \tau \leq T$ and T is the maximum look-ahead, the cell $\mathbf{q}$ will be probably searched by the agent j before the time step $k+\tau$. Therefore, an appropriate $\rho_{k+\tau}^i(\mathbf{q})$ must decrease the gain of searching the cell $\mathbf{q}$ during the τ^{th} look-ahead of the agent i. In order to find the discount factor, we need to know Λ_τ^i and $P(\mathscr{E}_{\mathbf{q}}^{j,\tau})$ which is the probability of presence of the agent i in the cell $\mathbf{q}$ from time step $k+\tau-1$ to the time step $k+\tau$ for $\forall i$ and $\tau \leq T$.

If the turning rate of the agent is relatively low, at the τ steps ahead, the agent is somewhere on an arc with the radius of r_τ, the angle of Θ_τ, and the center of the current position of the agent. Therefore, Λ_τ^i is the collection of cells which are enclosed by the arc with the radius of $r_{\tau-1}$, and the angle of $\Theta_{\tau-1}$, and the arc with the radius of r_τ and the angle of Θ_τ, and the center of the current position of the agent. The position of an agent between time step 2 and time step 3 is shown in the Figure 3-2.

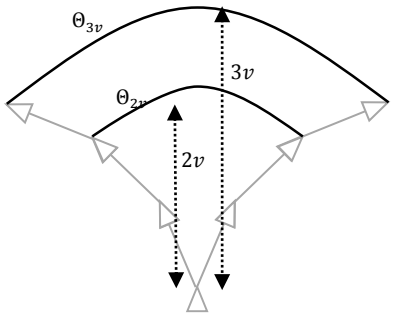


Figure 3-2. The position of the agent between time step 2 and time step 3

The value of r_τ is approximately equal to τv where v is the average velocity of the agent. The simplest approximation for Θ_τ is 2φ for all τ in the look-ahead horizon of the agent, where φ is the turning angle of the agents. This is a good approximation when the look-ahead horizon of the agent is small. A better approximation is one degree approximation

$$\Theta_\tau = 2\varphi(1 + \alpha\tau v) \tag{3-13}$$

where α is a scaling parameter.

We saw that the position of an agent in the near future is on a moving arc with the center of its current position. If we assume that the probability of presence of the agent on this arc is uniformly distributed, one can conclude that

$$P(\mathscr{E}_{\mathbf{q}}^{i,\tau}) \propto \begin{cases} \frac{1}{\|\mathbf{q}-\mathbf{p}_i\|.2\varphi(1+\alpha\|\mathbf{q}-\mathbf{p}_i\|)} & \mathbf{q} \in \Lambda_\tau^i \\ 0 & else \end{cases} \tag{3-14}$$

where $\|\mathbf{q} - \mathbf{p}_i\|$ and $2\varphi(1 + \alpha\|\mathbf{q} - \mathbf{p}_i\|)$ are the length and the angle of the moving arc when it crosses the cell $\mathbf{q}$, respectively. Therefore, the probability of presence of the agent i in the cell $\mathbf{q}$ during the τ^{th} look-ahead is as follows

$$P(\mathscr{E}_{\mathbf{q}}^{i,\tau}) = \begin{cases} \frac{\beta_\tau}{\|\mathbf{q}-\mathbf{p}_i\|.2\varphi(1+\alpha\|\mathbf{q}-\mathbf{p}_i\|)} & \mathbf{q} \in \Lambda_\tau^i \\ 0 & else \end{cases} \tag{3-15}$$

where β_τ is a scaling parameter that should be chosen in an appropriate way to ensure that the probability is always less than one. Since the length of the moving arc during the τ^{th} look-ahead is $r_\tau.\Theta_\tau = 2\varphi(1+\alpha\tau v)$, the scaling factor β_τ must be less than or equal to this value, i.e. $\beta_\tau \leq 2\varphi(1+\alpha\tau v)$. However, (3-15) is a discrete probability distribution and its support is the collection of cells in the environment. Thus, a less conservative condition for the value of β_τ is $\beta_\tau \leq 2\varphi(1+\alpha\tau v)/$number of cells in the moving arc. In the extreme condition when the moving arc is a straight line (an arc with the length of infinity), β_τ equals dimension of the cell. Therefore, β_τ must always be less or equal to the dimension of the cell. In deriving (3-15), we assumed that it is equally probable for the agent to be in any point on the moving arc. Figure 3-1-a shows that this assumption is not completely true. In fact, there are some places with more probability of presence of the agent than the other places which means the probability not only depends on the length of $\mathbf{q}-\mathbf{p}_i$, but also depends on the angle between $\mathbf{q}-\mathbf{p}_i$ and the current heading of the agent i to some extent. It is possible to use higher order approximation to consider this effect, at the cost of more computational complexity. However, the results shown in the simulation section demonstrate that how this relatively simple approximation can improve the performance of the mission.

Now, for each cell $\mathbf{q}$ in the sensor footprint of agent i at τ steps ahead, the discount value can be defined as follows

$$\rho^i_{k+\tau}(\mathbf{q}) = \max(0, 1 - K\sum_{\forall j\neq i}\sum_{t=1}^{\tau} P(\mathscr{E}^{j,t}_{\mathbf{q}})) \tag{3-16}$$

where K is a scaling parameter. It is clear that this discount factor is always between zero and one and is a non-increasing function of the probability that the cell will be visited by the other agents during the next τ steps. Since $P(\mathscr{E}^{j,\tau}_{\mathbf{q}})$ is non-zero only for $\mathbf{q} \in \Lambda^j_\tau$

$$\sum_{t=1}^{\tau} P(\mathscr{E}_{\mathbf{q}}^{j,t}) = \begin{cases} \frac{\beta_\tau}{\|\mathbf{q}-\mathbf{p}_j\|.2\varphi(1+\alpha\|\mathbf{q}-\mathbf{p}_j\|)} & \mathbf{q} \in \bigcup_{t=1}^{\tau} \Lambda_t^j \\ 0 & else \end{cases} \quad (3\text{-}17)$$

which makes the calculation of (3-16) very simple. In fact, in the worst case scenario when all agents may try to search the cell $\mathbf{q}$ during the next τ steps, we only need to calculate $n-1$ equations $\frac{\beta_\tau}{\|\mathbf{q}-\mathbf{p}_j\|.2\varphi(1+\alpha\|\mathbf{q}-\mathbf{p}_j\|)}$ for all agents j other than i, where n is the number of agent in the mission.

In the procedure of calculating the single step gain, first, the agent must recognize that whether in that step it is in the approximate future sector of the other agents, and if so, calculate the discount factor $\rho_{k+\tau}^{i}(\mathbf{q})$. Then, it can use this value to find the single step gain.

3.4.1.2 Simulation

In this section, we present some simulations to show the effect of cooperation on the performance of a multi agent search problem. It is desired to illustrate the capability of the proposed method to allow cooperation between agents which leads to a mission with higher performance. All simulations have been done in Matlab® R2010a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®.

The environment used in this simulation is a 80×80 square grid. There exists three targets known to be in 15×15 cells square areas (uncertainty regions) as shown in Figure 3-3 and 3-4, but their exact positions are unknown for the agents. The *a priori* probability of existence of these targets is uniformly distributed in their uncertainty region while their real positions are marked by the * marker. It is also considered that a virtual target exists in the environment and its uncertainty region is the entire terrain. Considering this target enforces the agents to search unexplored area of the environment. There are three agents in the environment that their starting positions are

shown by ► marker and their paths during the mission are shown by the solid, dotted, and dashed lines.

Each agent is equipped with a sensor that can detect the targets in its 2×2 cell footprint. The probabilities of true positive and false positive measurement of sensors are γ =0.9 and ε =0.05, respectively. If a sensor detects a target, the probability of existence of that target will increase. When the probability of existence of a target becomes greater than a specific threshold (0.9), then that target is considered as "found" target and it will be removed from the search list of the agents for the rest of the mission. The mission is terminated when all real targets marked as "found" or the maximum allowed mission duration is reached which is assumed to be 25 time steps. The probability updating rule is as presented by (2-24).

At each decision time step, the agents must decide to go straight, turn left 45 degrees or turn right 45 degrees. The agents can also ascend or descend to prevent the collision with others. However, this is usually not necessary because of the intrinsic collision avoidance capability of the discount factor ρ. We assume that once the agent has made a decision about its next action, that action can be performed immediately and then the agent continues its mission in a straight path until the next decision time step. The speed of agents assumed to be constant and is equal to three units per time step. In order to execute the simulation in a reasonable amount of time, we set the look-ahead horizon to four time steps. Therefore, at each time step, each agent chooses its optimal decision by using (3-6) where *T*=4 and the single step gain $g(\boldsymbol{x}_k, u_k, w_k)$ is calculated using (3-4). In calculating $g(\boldsymbol{x}_k, u_k, w_k)$, the values of λ and δ_k are assumed to be equal 1 for all time steps. The search gain σ_k is calculated by adding up the probabilities of existence of the targets in all cells that the agent covers during its mission from time step k to time step $k+1$, i.e. $\sigma_k = \sum_{\forall \mathbf{q} \in \Omega_k} \zeta(\mathbf{q}, k)$. The modified search gain $\hat{\sigma}_k$ is calculated by adding up the probabilities

of existence of the targets in all cells that the agent covers during its mission from time step k to time step $k+1$ times the discount factor, as shown by (3-11). The value of the discount factor is calculated by using (3-16) where scaling parameter K=0.25 and probability $P(\mathscr{E}_{\mathbf{q}}^{j,t})$ is calculated by using (3-15). In calculating $P(\mathscr{E}_{\mathbf{q}}^{j,t})$, the value of β_τ equals 0.5 for all τ.

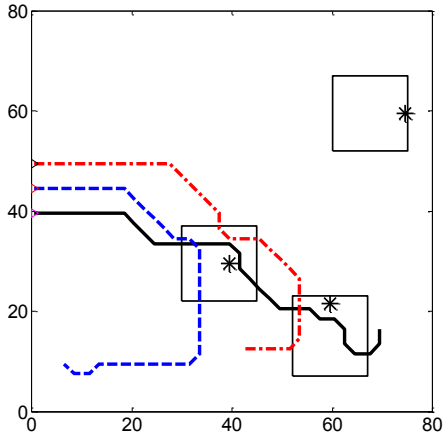


Figure 3-3. The agents do not cooperate in decision making.

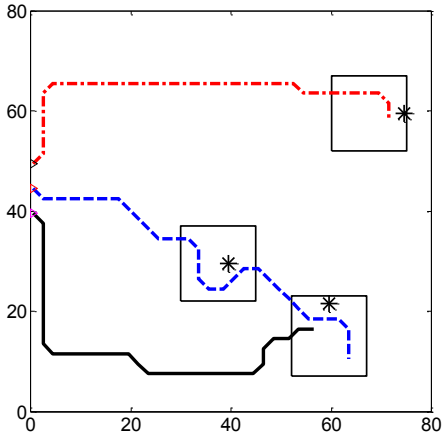


Figure 3-4. The agents cooperate in decision making.

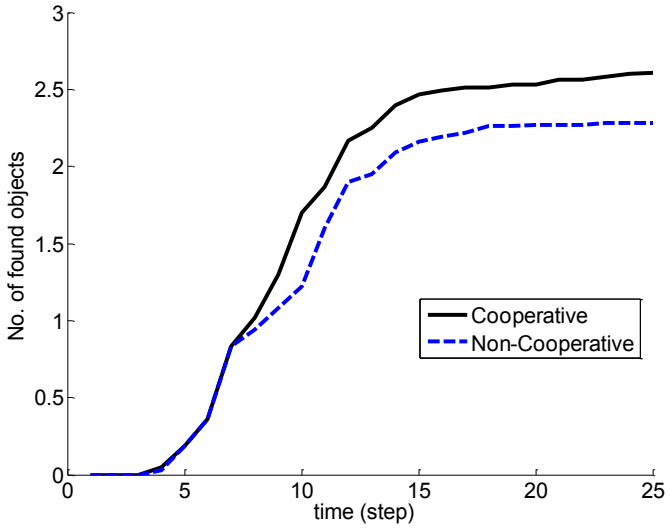


Figure 3-5. Comparison of the average number of found targets by cooperative and non-cooperative approaches for 75 random simulations.

In the scenario that is shown in Figure 3-3, the single step gain of σ_k prevents it from searching the areas that have been searched by the other agents in the past. But, because the agents do not try to take account of the future actions of the other agents, it is possible that all or some of agents decide to search the same area simultaneously, as seen in Figure 3-3.

In Figure 3-4, we use (3-11) to compute the single step gain of Dynamic Programming algorithm. Using the modified search gain to calculate the single step gain causes that each agent searches different parts of the terrain which increases the chance of finding different targets. As we can see in Figure 3-4, all targets have been found before the maximum duration time of mission reaches.

The average number of found targets during the course of different missions is shown in Figure 3-5. The simulations were repeated 75 times. In each simulation, the position of the targets and the uncertainty regions are randomly chosen while the starting position of the agents is fixed. The performance of the search mission with cooperation mechanism and without cooperation mechanism are compared in this figure. The results illustrate that using this cooperation mechanism can increase the total number of found targets and improve the performance of the mission. At the end of mission, the average number of found targets is increased about 25% by using the proposed cooperation mechanism.

We saw that the objective of a search mission could be gathering more information (reducing the uncertainty) or locating more targets in the environment. The search algorithms for both objectives are the same. The only difference is the definition of single step gain σ_k. We performed another simulation where the objective of the mission is to reduce the uncertainty in the environment. The structure of the environment is like the previous simulations which is a 80×80 square grid that includes three 15×15 cells uncertainty regions. The initial uncertainty map is

constructed as follows: for the cells inside an uncertainty region $\zeta(\mathbf{q}, 0)$ =0.1 and for the cells outside the uncertainty regions $(\mathbf{q}, 0)$ =0.01. Assigning a non-zero uncertainty value to the cells outside the uncertainty regions has the same effect as having a virtual robot in previous simulations. At each time step, the uncertainty value is updated using (3-7) where μ=0.1. The agents choose their optimal decision like the previous simulations. The only difference is that for the non-cooperative missions, the search gain σ_k is calculated by adding up the value of the uncertainty in all cells that the agent covers during its mission from time step k to time step $k+1$, i.e. $\sigma_k = \sum_{\forall \mathbf{q} \in \Omega_k} \zeta(\mathbf{q}, k)$. Similarly, for the cooperative missions, the modified search gain $\hat{\sigma}_k$ is calculated using (3-10). The simulations were repeated 75 times. In each simulation, the position of the uncertainty regions is randomly chosen while the starting position of the agents is fixed. The performance of the search mission with cooperation mechanism and without cooperation mechanism is compared in Figure 3-6. The results illustrate that using this cooperation mechanism can decrease the total uncertainty faster than a non-cooperative approach.

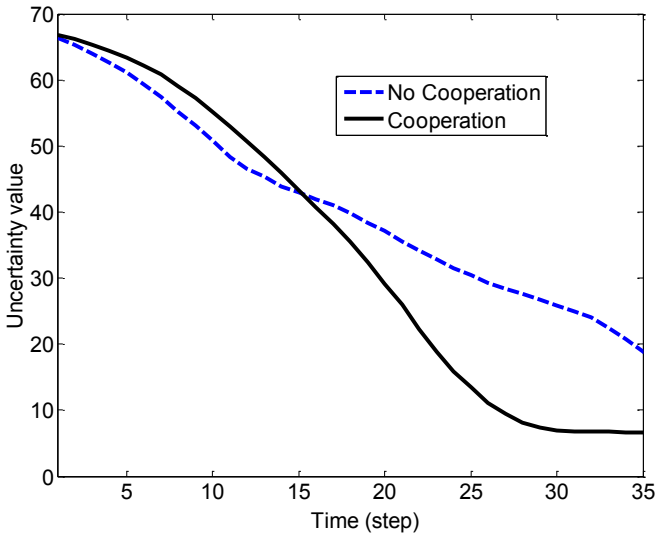


Figure 3-6. Comparison of the average uncertainty value by the cooperative and non-cooperative approaches for 75 random simulations.

3.4.2 Probabilistic Approach

In this section, we propose a probabilistic method to estimate the probability of different actions of other vehicles.

3.4.2.1 Probabilistic Estimation Method

At each decision time step k, assume that $\mathscr{E}_{\mathbf{q},\phi}^{i,\tau}$ is the event that agent i is in the cell $\mathbf{q} = (x, y)$ and its heading is equal to ϕ at the τ^{th} look-ahead. In order to decrease computational complexity, the heading angle is also discretized in to n equal sectors and $0 \leq \phi < n$. When there is no delay in communication, all agents know the exact position and heading of the others at each time step. This means that $P(\mathscr{E}_{\mathbf{q},\phi}^{i,0})$ is equal to one for a specified point $(\mathbf{p}_i, \phi_i)$ which is the current position of agent i at decision time step k and is equal to zero for the other points. Knowing the probability of presence of agent i in all cells $\mathbf{q}$ with any heading ϕ at τ steps ahead, this probability at $\tau + 1$ steps ahead can be found as follows

$$P(\mathscr{E}_{\mathbf{q},\phi}^{i,\tau+1}) = \sum_{\forall \mathbf{q}_0 \in Q} \sum_{\forall \phi_0 \in \Phi} \left[P(\mathscr{E}_{\mathbf{q},\phi}^{i,t+1} | \mathscr{E}_{\mathbf{q}_0,\phi_0}^{i,t}) P(\mathscr{E}_{\mathbf{q}_0,\phi_0}^{i,\tau}) \right] \tag{3-18}$$

where Q and Φ are the sets of all cells and all possible angles respectively. If we assume maximum velocity of agents is less than one cell per step then the above equation can be modified as

$$P(\mathscr{E}_{\mathbf{q},\phi}^{i,\tau+1}) = \sum_{j=-1}^{1} \sum_{k=-1}^{1} \sum_{\phi_0=0}^{n-1} \left[P(\mathscr{E}_{x,y,\phi}^{i,t+1} | \mathscr{E}_{x-j,y-k,\phi_0}^{i,t}) P(\mathscr{E}_{x-j,y-k,\phi_0}^{i,\tau}) \right] \tag{3-19}$$

We then define

$$\eta(j, k, \phi | \phi_0) = P(\mathscr{E}_{x,y,\phi}^{i,t+1} | \mathscr{E}_{x-j,y-k,\phi_0}^{i,t})$$

as the probability that agent i goes (j, k) cells ahead and changes its heading from ϕ_0 to ϕ during one time step of the mission. This probability is dependent on the maximum velocity and the maximum turn rate of the agent and assumed to be known *a priori* for all j and $k(-1 \leq j, k \leq 1)$ and all ϕ and ϕ_0 $(0 \leq \phi, \phi_0 < n)$. Then (3-19) can be shown as

$$P(\mathscr{E}_{\mathbf{q},\phi}^{i,\tau+1}) = \sum_{j=-1}^{1}\sum_{k=-1}^{1}\sum_{\phi_0=0}^{n-1}\left[\eta(j,k,\phi|\phi_0)P(\mathscr{E}_{x-j,y-k,\phi_0}^{i,\tau})\right] \quad (3\text{-}20)$$

The probability of presence of agent i at time step $\tau+1$ in the cell $\mathbf{q}$ is equal to

$$P(\mathscr{E}_{\mathbf{q}}^{i,\tau+1}) = \sum_{\phi=0}^{n-1} P(\mathscr{E}_{\mathbf{q},\phi}^{i,\tau+1}) \quad (3\text{-}21)$$

which is the total probability of presence of agent i at time step τ+1 in that cell with different heading angles (orientation). Therefore, the total probability that cell $\mathbf{q}$ will be visited by the agent i during the next τ steps of its mission is $\sum_{t=1}^{\tau} P(\mathscr{E}_{\mathbf{q}}^{i,t})$. So, the total probability that the cell $\mathbf{q}$ is visited during the next τ steps by one of the agents other than agent i is $\sum_{\forall j\neq i}\sum_{t=1}^{\tau} P(\mathscr{E}_{\mathbf{q}}^{j,t})$. Now, for each cell $\mathbf{q}$ in the sensor footprint of agent i at τ-steps ahead, the discount value can be defined as follows

$$\rho_{k+\tau}^{i}(\mathbf{q}) = \max(0, 1 - K\sum_{\forall j\neq i}\sum_{t=1}^{\tau} P(\mathscr{E}_{\mathbf{q}}^{j,t})) \quad (3\text{-}22)$$

where K is a scaling parameter. It is clear that this discount factor is always between zero and one and is a non-increasing function of the probability that the cell will be visited by the other agents during the next τ steps.

3.4.2.2 Simulation

In this section, we present some simulations to show the effect of using the proposed probabilistic estimation method on the performance of a multi agent search problem. All simulations have been done in Matlab® R2010a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®.

The environment used in this simulation is a 20×20 square grid. There exist four targets known to be in a 5×5 square areas as shown in Figure 3-7, but their exact positions are unknown. The *a priori* probability of existence of these targets is uniformly distributed in their uncertainty region while their real positions are marked by the * marker. It is also considered that a virtual

target exists in the environment and its uncertainty region is the entire terrain. There are three UAVs in the environment. Figure 3-8 shows the probability map of a typical mission after nine time steps. The positions of UAVs at each step are shown by ► marker and their paths during the mission are shown by lines.

Each agent is equipped with a sensor that can detect the targets in its 2×2 cells footprint. The probabilities of true positive and false positive measurement of sensors are $\gamma = 0.9$ and $\varepsilon = 0.1$, respectively. When the probability of existence of a target becomes greater than a specific threshold which is equal to 0.9 in this simulation, then that target is considered as "found" target and it will be removed from the search list of the agents for the rest of the mission. The mission is terminated when all real targets marked as "found" or the maximum allowed mission duration is reached which is 30 time steps. The probability updating rule is as presented by (2-24). As it can be seen in Figure 3-8, some parts of the uncertainty region in the bottom-left quarter of environment have been searched with two of UAVs, but the target has not been found yet. Therefore, the probability of existence of the target in those cells is decreased while the probability for the other cells in that region is increased.

At each decision time step, UAVs must decide to go straight, turn 22.5 degrees left or turn 22.5 degrees right. Each UAV flies in a predefined constant level to avoid collision of UAVs. However, this is usually not necessary because of the intrinsic collision avoidance capability of the discount factor ρ. Therefore, we can consider this problem as a two dimensional search problem. The speed of UAVs assumed to be constant and is equal to a unit per time step. In order to execute the simulation in a reasonable amount of time, we set the look-ahead horizon to 5 time steps. Agents follow the same procedure as explained in section to find their optimal decision. The only difference is that, now the probability $P(\mathscr{E}_{\mathbf{q}}^{j,t})$ is calculated by using (3-21). *Transition*

Probability, $\eta(j,k,\phi|\phi_0)$, is calculated by offline simulations and is known *a priori*. To find $\eta(j,k,\phi|\phi_0)$, it is initialized by zero. An agent is assumed to be at the origin with one of 16 (=360/2.5) possible headings ϕ_0. The next decision of the agent is chosen randomly. If the agent is now at cell (j,k) with heading ϕ, $\eta(j,k,\phi|\phi_0) = \eta(j,k,\phi|\phi_0) + 1$. For any possible initial heading ϕ_0, the procedure is repeated several times (in our simulation 50 times). At the end $\eta(j,k,\phi|\phi_0) = \eta(j,k,\phi|\phi_0)$/number of repetitions.

Total probability of presence of all UAVs from the current position until 5-step ahead is shown in Figure 3-9 for a typical situation. Positions of UAVs at each step are shown by ► marker. Simulations have been done 50 times and the average number of found targets is reported. Actual positions of the targets are randomly chosen for each repetition of simulation but they are the same for all scenarios. All three UAVs start their mission from the south west corner of the terrain.

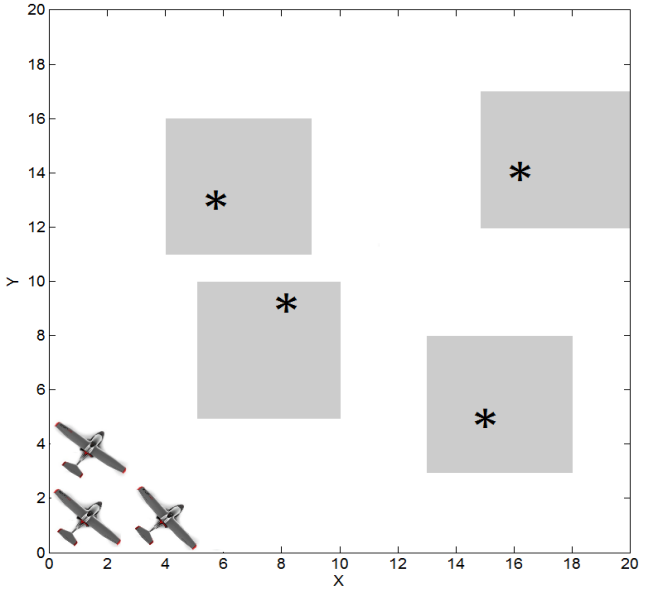


Figure 3-7. The problem environment. The grey rectangles are the uncertainty regions of different targets and * denotes the actual position of the targets which is unknown for the search UAVs.

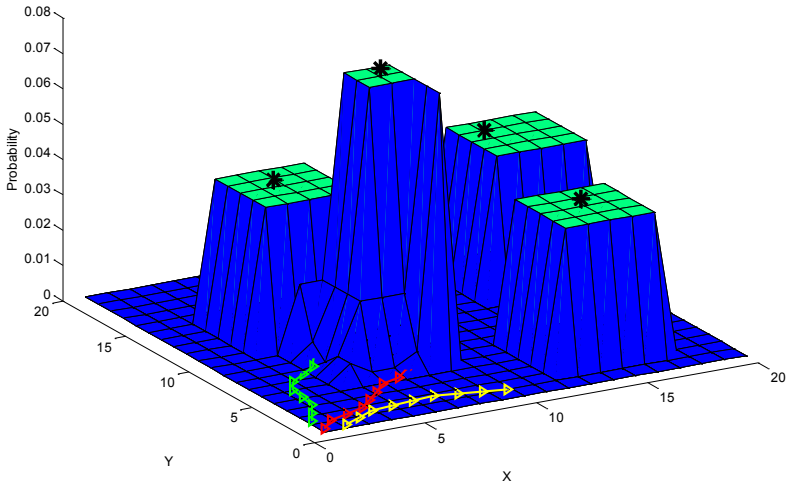


Figure 3-8. A typical search mission at the ninth time step. The actual position of the targets is shown by * and positions of vehicles at different time steps are shown by ►.

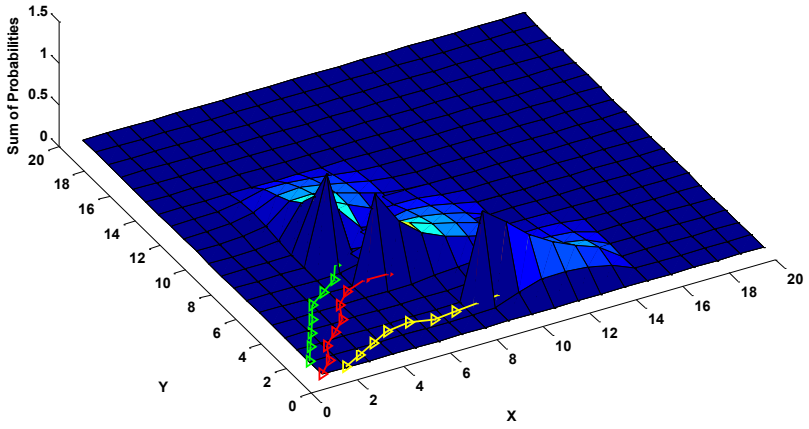


Figure 3-9. The total probability of presence of UAVs from the current position until 5-step ahead.

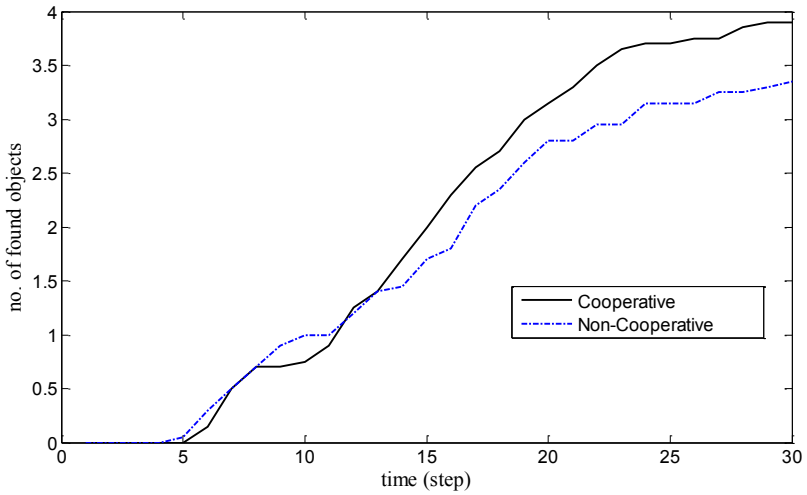


Figure 3-10. The average number of found targets for 50 random simulations.

In Figure 3-10, the performance of search mission with cooperation mechanism and without cooperation mechanism is compared. We can see that both methods have almost the same performance at the beginning of the mission, but the performance of cooperative method is dominant when the time grows up. At the end of the mission, the average number of found targets is increased about 20% by using the proposed cooperation mechanism. When there is no cooperation between UAVs, they may try to locate the same targets as shown in Figure 3-11, whereas in the cooperative method different UAVs try to search different parts of the terrain as shown in Figure 3-12.

In order to compare the performance of the proposed Geometric and Probabilistic methods with the performance of fully cooperative and non-cooperating methods, the same 50 simulation have been performed by using the geometric method, fully cooperative and non-cooperating methods and the average results are summarized in Table 3-1. In fully cooperated method, each agent expands the decision tree of all agents at each decision time step to find its optimal action.

In fact, each agent works like a central controller to find the optimal actions of all agents, but then it only performs its individual action.

All simulation have been done in Matlab® R2012a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®. It can be seen that both Geometric and Probabilistic methods can considerably increase the performance of the mission (decrease the average time steps to find all targets) with respect to the non-cooperative approach while increasing the computation time by less than 10%. The performance of fully cooperative approach is better than the performance of all other approaches but it takes about three times more computation time than the others. Calculating the discount factor ρ by Probabilistic method is about 6 times more time-consuming than the Geometric method. Therefore, when the computational capability of agents is relatively low, it may not be possible to use the Probabilistic method.

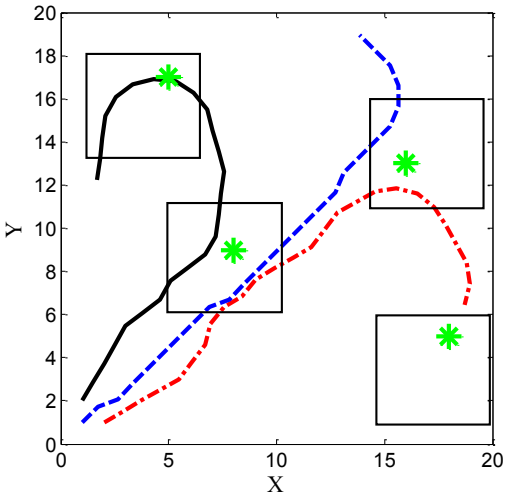


Figure 3-11. A typical mission without the cooperation mechanism.

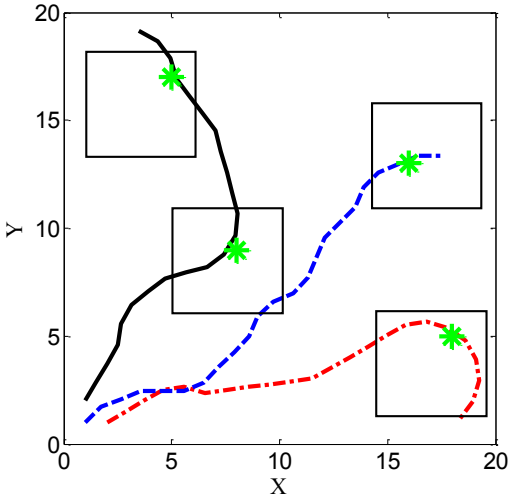


Figure 3-12. A typical mission with the cooperation mechanism.

TABLE 3-1. COMPARISON BETWEEN DIFFERENT METHODS FOR 50 SIMULATION

	Average time steps to find all targets	Average computation time of the discount factor	Average computation time of each decision
Geometric	48	8 ms	508 ms
Probabilistic	37	45 ms	545 ms
Fully Cooperative	29	-	1500 ms
Non-Cooperative	72	-	500 ms

3.5 Communication Delay

The communication between agents is important when a cooperative task is under consideration. It was mentioned that the state of system is comprised of the search status, and the agents' status. In each step, the agents receive the position and updated probability maps of the others. So they can update their states. Ideally, if there is no delay in communication, all agents observe the same state of the system in each step. But limited bandwidth of communication channels and the distance between the transmitter and receiver impose a delay in communication. This delay decreases the level of cooperation between agents and might worsen the performance of the mission. In the extreme case, when the delay goes to infinity, it means that there is no communication among agents. So agents disregard the other ones when they want to make decisions about their next actions. The performance of this mission is almost the same as the performance of a single agent mission. However, the redundant agent can improve the mission performance when the probability of damage is non-zero, i.e. $\delta < 1$.

If an agent can estimate the actual state of system from the delayed one, then the performance of mission will be improved. It is shown that the influence of cooperation between agents is that they do not try to search the areas that the other agents have searched in the past or may search in the future. Therefore, in the presence of communication delay, if an agent estimates the areas that might have been searched by other agents until now, then the objective function of the agent can

be modified to prevent it from searching those areas which are most probable to have been searched by the other agents before.

3.5.1 Geometric Approach

In this section, a geometric method is proposed to estimate and to compensate the effect of communication delay between agents.

3.5.1.1 Geometric Method for Delay Compensation

In section 3.4.1, we proposed a geometric method to estimate the effect of future actions of the other agents on the current decision of the planning agent. Now, we can use the same method to estimate the influence of past actions of the other agents on the current decision of the planning agent. In this case, each agent only knows the position and the angle of the other agents up to d steps earlier, where d is the amount of delay.

At each decision time step k, we define $\Lambda_{\tau|d}^{i}$ as the set of all cells that agent i may have been visited during the mission from the time step $(k-d)+\tau-1$ to the time step $(k-d)+\tau$, knowing its position at decision time step $k-d$. Using similar assumption as section 2.4.1, the probability of presence of agent i in the cell $\mathbf{q}$ during the time step $(k-d)+\tau$ is as follows

$$P(\mathscr{E}_{\mathbf{q}}^{i,\tau|d}) = \begin{cases} \frac{\beta_\tau}{\|\mathbf{q}-\mathbf{p}_i(k-d)\|.2\varphi(1+\alpha\|\mathbf{q}-\mathbf{p}_i\mathbf{p}_i(k-d)\|)} & \mathbf{q} \in \Lambda_{\tau|d}^{i} \\ 0 & else \end{cases} \tag{3-23}$$

where β_τ is a scaling parameter that should be chosen in an appropriate way to ensure that the probability is always less than one. The only difference between (3-23) and (3-15) is that $\mathbf{p}_i(k)$ is replaced by $\mathbf{p}_i(k-d)$ in (3-23). In fact (3-15) is a special case of (3-23) with $d=0$.

The agent can use (3-23) to estimate the probability of presence of other agents in different cells during the last d steps. Then, for each cell $\mathbf{q}$ in the sensor footprint of agent i, the discount factor that compensates for the effect of time delay can be calculated as follows

$$\rho^{i}_{0|d}(\mathbf{q}) = \max(0,1 - K \sum_{\forall j \neq i} \sum_{t=1}^{d} P(\mathscr{E}_{\mathbf{q}}^{j,t|d})) \tag{3-24}$$

where K is a scaling parameter. It should be noted that $P(\mathscr{E}_{\mathbf{q}}^{j,t|d})$ is non-zero only for $\mathbf{q} \in \Lambda^{i}_{\tau|d}$, therefore, calculation of $\rho^{i}_{0|d}$ using (3-24) is relatively simple. Agent i then uses $\rho^{i}_{0|d}$ to modify its search gain in the next T steps, i.e. $\hat{\sigma}_{k+t}, \forall t \leq T$.

The above approach can be generalized for the case of cooperative decision making in the presence of communication delay. In this case, the planning agent not only estimates the effect of past actions of the other agents on its current decision but also estimates the effect of their future actions. Thus, at each decision time step k, the value of discount factor of agent i at τ steps ahead that also compensates for the effect of d steps communication delay is as follows

$$\rho^{i}_{\tau|d}(\mathbf{q}) = \max(0,1 - K \sum_{\forall j \neq i} \sum_{t=1}^{\tau+d} P(\mathscr{E}_{\mathbf{q}}^{j,t|d})) \tag{3-25}$$

where K is a scaling parameter and $P(\mathscr{E}_{\mathbf{q}}^{j,t|d})$ can be calculated by using (3-23). It should be noted that $P(\mathscr{E}_{\mathbf{q}}^{j,t|d})$ is non-zero only for $\mathbf{q} \in \Lambda^{i}_{\tau|d}$, therefore, calculation of $\rho^{i}_{\tau|d}$ using (3-25) is relatively simple. The agent i then uses $\rho^{i}_{\tau|d}$ to calculate its modified search gain of the τ steps ahead, i.e. $\hat{\sigma}_{k+\tau}$.

It is obvious that the ability of (3-25) to accurately estimate the effect of actions of the other agents on the decision of the planning agent decreases when the amount of delay and the look-ahead horizon increase.

3.5.1.2 Simulation

In this section, we present some simulations to show how the proposed Geometric estimation method can compensate for the effect of known communication delay between agents. All simulation have been done in Matlab® R2010a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®.

The problem structure and parameters in this section are similar to 3.4.1.2. In Figure 3-13, 2 time steps communication delay causes that all agents choose almost the same path during the mission that means the performance of mission with three agents is like the performance of a single agent mission. Figure 3-14 shows how the proposed method is able to mitigate the impact of communication delay on the mission performance. The communication delay in this scenario is equal to that in Figure 3-13, but the single step gain is modified to alleviate the effect of delay. To modify the single step gain, the discount factor is calculated by using (3-24) where d=2. As expected, the result is very similar to Figure 3-3, where there is no communication delay. It means the effect of communication delay has been compensated by using the proposed method.

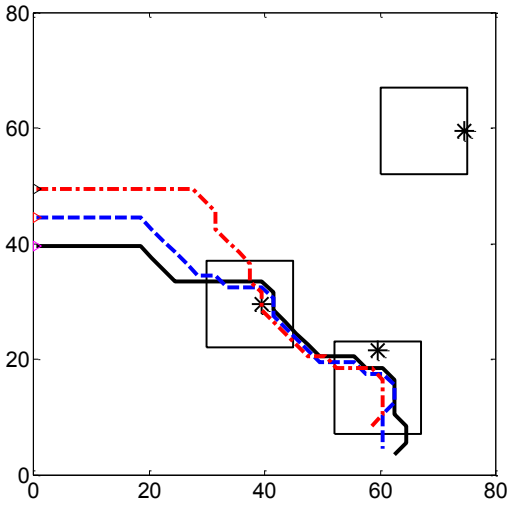


Figure 3-13. A typical mission with communication delay. The agents do not cooperate in decision making.

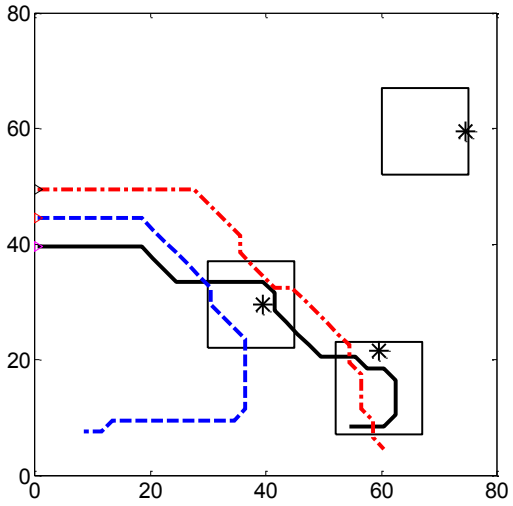


Figure 3-14. A typical mission with communication delay. The compensation method has been used to mitigate the effect of communication delay.

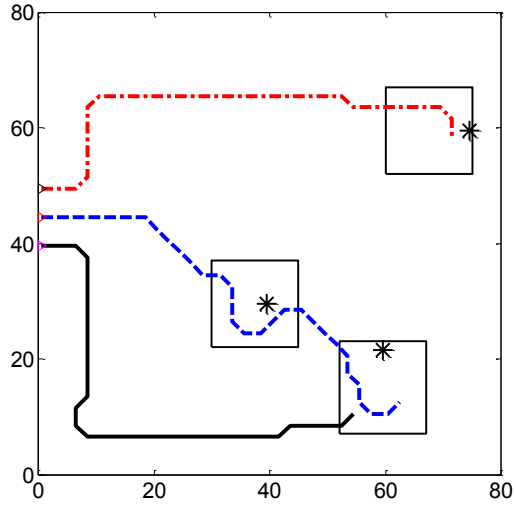


Figure 3-15. A typical mission with communication delay. The cooperation mechanism has been modified to consider the effect of future actions of other vehicles while mitigating the impact of communication delay.

In order to compensate for the impact of communication delay and enable cooperation between agents at the same time, we should use (3-25) to calculate discount factor. Figure 3-15 depicts the results. The result is very similar to Figure 3-4 where the agents make decision cooperatively in the absence of delay.

75 random simulations are performed, with five different amounts of delay. The simulation are similar to section 3.4.1.2 but there is communication delay among the agents. In Figure 3-16, the performance of the mission (average number of found objects) for different amounts of delay is compared. It is expected that the performance of mission declines when the amount of delay increases. Figure 3-17 and Figure 3-18 show how the communication delay deteriorates the performance of mission and how the compensation mechanism can mitigate it. The results demonstrate that this compensation mechanism is able to mitigate the influence of delay and improve the performance of mission. Amount of communication delay in Figure 3-17 and Figure 3-18 is equal to two and four steps, respectively. These figures show the average value of found targets for 75 random simulation. As it is expected, when the amount of delay increases, the ability of proposed method to compensate the impact of delay decreases.

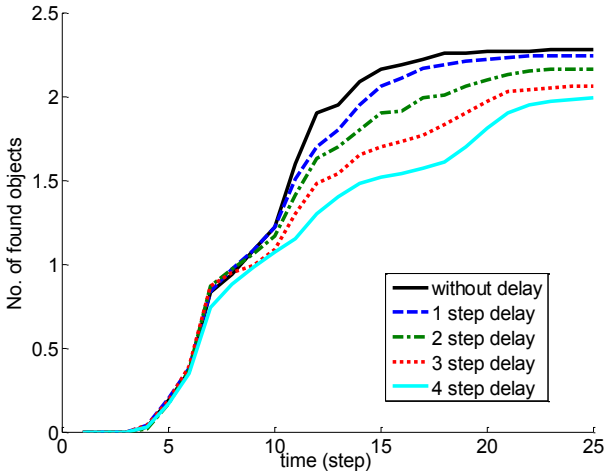


Figure 3-16. The average number of found targets for 75 random simulations with different amounts of delay.

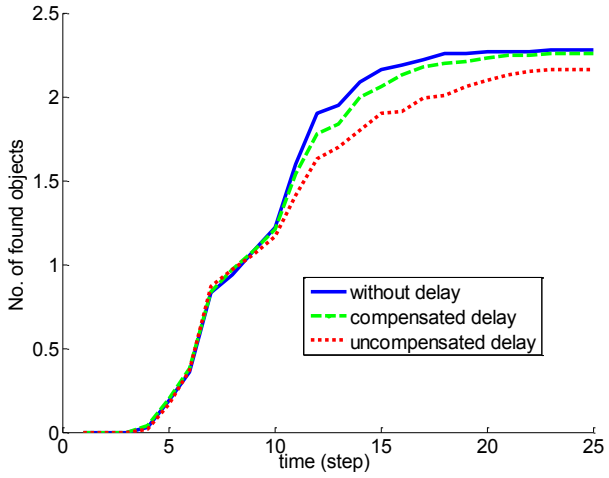


Figure 3-17. The average number of found targets for 75 random simulations with 2 steps communication delay.

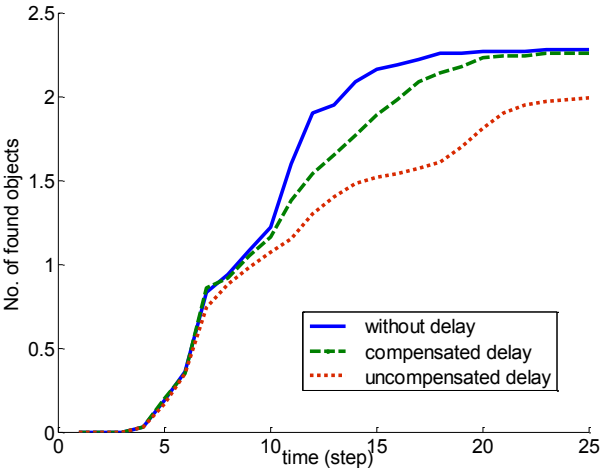


Figure 3-18. The average number of found targets for 75 random simulations with 4 steps communication delay.

3.5.2 Probabilistic Approach

In this section, a probabilistic method is proposed to estimate and to compensate the effect of communication delay between vehicles.

3.5.2.1 Probabilistic Method for Delay Compensation

In section 3.4.2.1, we proposed a probabilistic method to estimate the effect of future actions of other agents on the current decision of each agent. Now, we can use the same method to estimate the influence of past actions of other agents on the current decision of each agent. Amount of delay is assumed to be equal to d which means the most recent available information from other vehicles is related to d steps ago. Therefore, $P(\mathscr{E}_{\mathbf{q},\phi}^{i,0|d})$ is equal to one for the point $(\mathbf{p}_i(k-d), \phi_i(k-d))$ which is the position of agent i at decision time step $k-d$ and is equal to zero for the other points. Knowing the probability of presence of an agent in different cells with different angles at time step τ, the probability at the time step $\tau+1$ can be found using (3-20), then the total probability of presence of the agent at time step τ+1 in any given cell can be found using (3-21).

Therefore, if the most recent information about the agent i is from the decision time step $k-d$, the total probability that the cell $\mathbf{q}$ has been visited by the agent i during the mission from the decision time step $k-d$ to the current decision time step k is $\sum_{t=1}^{d} P(\mathscr{E}_{\mathbf{q}}^{i,t|d})$. So, the total probability that the cell $\mathbf{q}$ has been visited during the last d decision time steps by one of the agents other than agent i is $\sum_{\forall j\neq i}\sum_{t=1}^{d} P(\mathscr{E}_{\mathbf{q}}^{j,t|d})$. Then, for each cell $\mathbf{q}$ in the sensor footprint of agent i, the discount factor that compensates for the effect of time delay can be calculated as follows

$$\rho_{0|d}^{i}(\mathbf{q}) = \max(0, 1 - K\sum_{\forall j\neq i}\sum_{t=1}^{d} P(\mathscr{E}_{\mathbf{q}}^{j,t|d})) \tag{3-26}$$

where K is a scaling parameter. The agent i then uses $\rho_{0|d}^{i}$ to modify its search gain in the next T steps, i.e. $\hat{\sigma}_{k+t}, \forall t \leq T$. Indeed, knowing the exact position of the other agents at $k-d$, the agent i tries to estimate the cells that were visited by the other agents from then until d steps ahead which is the present time. Then, it uses $\rho_{0|d}^{i}$ to avoid searching those cells.

The above approach can be generalized for the case of cooperative decision making in the presence of communication delay. In this case, the agent not only estimates the effect of past actions of other agents on its current decision but also estimates the effect of their future actions. For this purpose, at each decision time step k, the value of discount factor for each cell in the sensor footprint of agent i at τ steps ahead should be modified as follows

$$\rho^{i}_{\tau|d}(\mathbf{q}) = \max(0,1 - K \sum_{\forall j \neq i} \sum_{t=1}^{\tau+d} P(\mathscr{C}_{\mathbf{q}}^{j,t|d})) \tag{3-27}$$

where K is a scaling parameter. The agent i then uses $\rho^{i}_{\tau|d}$ to calculate its modified search gain of the τ steps ahead, i.e. $\hat{\sigma}_{k+\tau}$.

It is obvious that the ability of (3-27) to accurately estimate the effect of other agents on the decision of each agent decreases when the amount of delay and the look-ahead horizon of Dynamic Programming algorithm increase.

3.5.2.2 Simulation

In this section, we present some simulations to show how the proposed probabilistic estimation method can compensate for the effect of known communication delay between agents. All simulation have been done in Matlab® R2010a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®.

The problem structure and parameters in this section are similar to 3.4.2.2. 50 random simulations are performed for different amounts of delay. Figure 3-19 shows how the communication delay deteriorates the performance of mission and how the compensation mechanism can mitigate it. In all cases, cooperation mechanism is used to increase the performance of mission. Solid line shows the number of found objects without delay (The discount factor is calculated by using (3-22)). The number of found objects in the presence of delay without using

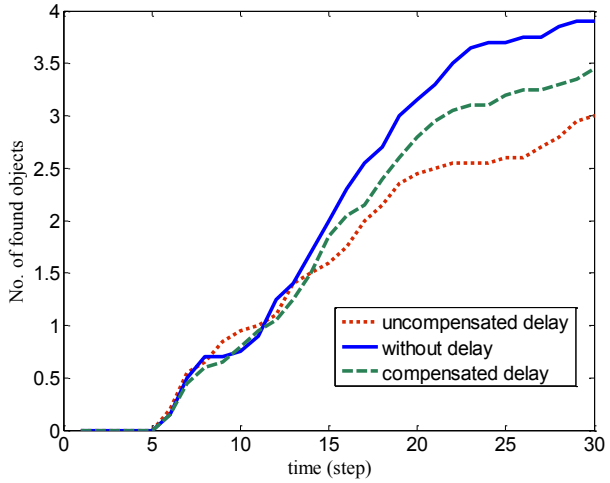


Figure 3-19. The average number of found targets for 50 random simulations with 4 steps communication delay.

the compensation mechanism (The discount factor is calculated by using (3-22)) and with using that mechanism (The discount factor is calculated by using (3-27)) are shown by dotted line and dashed line, respectively. The amount of delay is equal to 4 decision steps which is rather long in comparison to 5-step maximum look-ahead. The results demonstrate that this compensation mechanism is able to mitigate the influence of delay and improve the performance of mission.

3.6 Conclusion

In this chapter, a decentralized approach is used for the cooperative search mission where each agent individually chooses its optimal action. We argued that when the agent knows the future position of other agents, it should avoid those areas. This helps agents to explore different areas of the environment and gather more information. The search gain of the agent is modified to reflect this

However, predicting the exact position of other agents is computationally expensive. We proposed two methods to estimate the future position of other agents. Simulation results show that

both methods can considerably improve the performance of the mission with respect to a non-cooperative approach without significantly increasing the computation. The proposed probabilistic estimation method can provide a better performance than the proposed geometric estimation method, but it needs more computation.

Remarks:

As we discussed in chapter 1, probabilistic search approach is useful when there are some prior information about the position of the targets. Otherwise using an exhaustive search method provides the same performance with much less computation. Similarly, when there are multiple search agents and there is no prior information about the position of the targets, the cooperation mechanism is straightforward. The environment is divided between agents and each agent uses an exhaustive search method to find the targets in its own region.

In a multi agents search mission, number of agents is usually small relative to the size of environment. When there are relatively large number of search agents, using a probabilistic search approach does not provide significant improvement in the performance. Therefore, in this case, the environment can still be divided between agents and each agent can use an exhaustive search method to find the targets in its own region.

CHAPTER 4

Cooperative Search and Coverage Using Locational Optimization

In this chapter, the cooperative search and coverage problem is investigated. First, Voronoi partitioning and its different extensions are reviewed. Then, the search and coverage problem is introduced and formulated. A new distribution density model is introduced which is a function of position of some unknown targets in the environment. The cooperative search method that discussed in the previous chapter is used to update the distribution density function for the coverage task. A well-known Centroidal Voronoi Configuration method for the coverage is used to solve the coverage problem.

The cooperative multi agent search and coverage approach is useful for many applications involving distributed sensing and distributed actuation. For example, consider a team of Unmanned Aerial Vehicles (UAVs) charged with detecting and extinguishing multiple fires in a partially known environment like a forest. The fire detector UAVs with on-board sensors search the environment to find the centre of fires. Then, by using this information, the fire fighter UAVs aggregate in the perimeter of fires. Similarly, consider a group of water-borne vehicles which are in charge of monitoring and cleaning up an oil spill. The monitoring vehicles find the areas where the spill is most severe, while cleaning vehicles distribute themselves over the spill and concentrate their efforts on those most severe areas, without neglecting the areas where the spill is not as severe. In general, any application in which a group of automated mobile agents is required to

provide collective sensing and actuation over an environment can be considered as an example of this framework.

We consider the case in which some service agents deploy to cover an uncertain environment. They are expected to spread out over an environment while aggregating in areas of high service needs. Furthermore, the service agents are uncertain about the exact areas of service needs beforehand. In order to decrease the level of uncertainty, the environment is searched by some search agents which are equipped with sensors to detect the exact areas of service needs. As mission goes on, the service agents use the updated information of search vehicles to change their configuration and cover the environment more efficiently. A brief introduction to Voronoi partitioning and locational optimization using Voronoi tessellation for coverage problem is presented in the next two sections.

4.1 Voronoi Partitioning

The partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other is called Voronoi partitioning. A Voronoi diagram is sometimes also known as a Dirichlet tessellation [118].

Given a set of two or more but finite number of distinct points in the Euclidean plane, we associate all locations in that space with the closest member(s) of the point set with respect to the Euclidean distance. The result is a tessellation of the plane into a set of the regions associated with members of the point set. This tessellation is called the *planar ordinary Voronoi diagram* generated by the point set, and the regions constituting the Voronoi diagram are called *Ordinary Voronoi polygons*.

Let $P = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\} \subset \mathcal{Q} \subset \mathbb{R}^2$, where $\mathcal{Q}$ is a convex polytope, $n > 2$ and $\mathbf{p}_i \neq \mathbf{p}_j$ for $i \neq j$. We call the region given by

$$V_i = \left\{\mathbf{q} \in \mathcal{Q} | \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\|, \forall j \neq i, j \in \{1, \dots, n\}\right\}$$

the *planar ordinary Voronoi polygon* associated with $\mathbf{p}_i$, and the set given by

$$V = \{V_1, V_2, \dots, V_n\}$$

the *planar ordinary Voronoi diagram* generated by P. The point $\mathbf{p}_i$ is called the *generator point* of the i^{th} Voronoi polygon, and the set $P = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ is called the *generator set* of the Voronoi diagram [118].

We notice from the definition of the ordinary Voronoi diagram that an abstract idea for defining a Voronoi diagram is that every point in a space is assigned to at least one of the generator points according to a certain assignment rule, and that the resulting sets of points associated with the generator points are collectively exhaustive and mutually exclusive except for the boundaries. To generalize the Voronoi diagram, the Euclidean distance is replaced with a *distance metric* which is defined as a mapping, $d(\mathbf{q}, \mathbf{p}_i): \mathcal{Q} \times \mathcal{Q} \to \mathbb{R}_{\geq 0}$ satisfying the following four axioms: (i) $d(\mathbf{p}_i, \mathbf{p}_i) = 0$, (ii) $d(\mathbf{p}_i, \mathbf{p}_j) \leq d(\mathbf{p}_i, \mathbf{p}_k) + d(\mathbf{p}_k, \mathbf{p}_j)$, (iii) $d(\mathbf{p}_i, \mathbf{p}_j) = d(\mathbf{p}_j, \mathbf{p}_i)$, (iv) $d(\mathbf{p}_i, \mathbf{p}_j) > 0, \forall i \neq j$. In the ordinary Voronoi diagram, the distance metric is the Euclidean distance. Therefore, the formal definition of generalized Voronoi diagram is

$$V_i = \left\{\mathbf{q} \in \mathcal{Q} | \, d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j), \forall j \neq i, j \in \{1, \dots, n\}\right\}$$

where $P = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\} \subset \mathcal{Q} \subset \mathbb{R}^2$ is the set of generator points [118].

In the ordinary Voronoi diagram, we implicitly assume that generator points are identical. In some practical application, this assumption may not be appropriate. Rather, it is more appropriate to assume that generator points have different weights reflecting the variable property of the

generator points. In the case of multi agent tasks where the generator points usually correspond to the position of agents, it may reflect different capability of different agents. In the case of non-identical vehicles, we may use a weighted distance function $d_w(.)$ with a set of weight parameters $\boldsymbol{w} = \{w_1, w_2, \dots, w_n\}$. These weights can be a measure of coverage ability or energy (remaining fuel) of agents. Different weighted Voronoi diagrams have been used in literature. The most useful weighted Voronoi diagrams in multi agent problems are as follows [118]

- *Multiplicatively weighted Voronoi diagram:*

$$d_{mw}(\mathbf{q}, \mathbf{p}_i) = \frac{1}{w_i} \|\mathbf{q} - \mathbf{p}_i\|, \qquad w_i > 0$$

- *Additively weighted Voronoi diagram:*

$$d_{aw}(\mathbf{q}, \mathbf{p}_i) = \|\mathbf{q} - \mathbf{p}_i\| - w_i,$$

- *Compoundly weighted Voronoi diagram:*

$$d_{cw}(\mathbf{q}, \mathbf{p}_i) = \frac{1}{w_{im}} \|\mathbf{q} - \mathbf{p}_i\| - w_{ia}, \quad w_{ia} > 0$$

The notion of weighted Voronoi diagrams is not useful when the agents are not omnidirectional or when it is not always possible to reach from one point to another point using a straight line. However, it may be possible to use a weighted Voronoi distance as a good estimate of actual (non-Euclidian) distance.

Another implicit assumption in ordinary Voronoi diagram and even weighted Voronoi diagrams is that a point in the domain belongs to its closest generator point regardless of how far the point is. In multi agent problems, it is not an appropriate assumption. In fact, the agents have limited capabilities due to limited fuel in service problems or limited sensory range in search problems. The limited range Voronoi diagram is defined as follows [118]

$$V_i = \left\{ \mathbf{q} \in \mathcal{Q} \cap B(\mathbf{p}_i, r_i) |\, d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j), \forall j \neq i, j \in \{1, \dots, n\} \right\}$$

where $B(\mathbf{p}_i, r_i)$ is a circle with the center of $\mathbf{p}_i$ and the radius of r_i, and r_i is the range of ith point. Of course, limited range Voronoi diagram is not necessarily a partition and some points in the environment may not belong to any Voronoi region. Different Voronoi partitions and the algorithms to produce them are discussed in [118] in more detail.

4.2 Locational Optimization

The environment is denoted by $\mathcal{Q}$ which is a convex polytope in $\mathbb{R}^2$ including its interior. An arbitrary point in $\mathcal{Q}$ is denoted as $\mathbf{q}$, the position of the i^{th} service agent is denoted as $\mathbf{p}_i$, and the set of positions of all service agents is denoted as $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p_n}\}$. The function $\varphi: \mathcal{Q} \to \mathbb{R}_+$ is a distribution density function that defines a weight for each point. This function may reflect knowledge of the probability of occurrence of events in different regions, or simply a measure of relative importance of different regions in $\mathcal{Q}$. Therefore, the higher the value of $\varphi(\mathbf{q})$ the more attention the group has to pay to $\mathbf{q}$. A non-increasing and piecewise continuously differentiable function $f: \mathbb{R}_+ \to \mathbb{R}$ is defined as a performance function which describes the utility of placing an agent at a certain distance from a location in the environment. The smaller the distance, the better the performance is. In servicing problem, performance functions can encode the travel time or the energy expenditure required to service a specific destination.

Locational optimization problem is considered as the task of minimizing the following locational optimization function [120]

$$\mathcal{H}(\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p_n}) = -\int_{\mathcal{Q}} \max_{i \in \{1,2,\dots,n\}} f(\|\mathbf{q} - \mathbf{p}_i\|)\, \varphi(\mathbf{q}) d\mathbf{q} \tag{4-1}$$

which means for each $\mathbf{q} \in \mathcal{Q}$, consider the best coverage of $\mathbf{q}$ among those provided by each of the agents, then evaluate the performance by the importance $\varphi(\mathbf{q})$ of $\mathbf{q}$, and finally sum the resulting

quantity over all $\mathbf{q} \in Q$ to obtain $\mathcal{H}(\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_n})$ as a measure of the overall coverage provided by $\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_n}$.

If we define a partition of Q as a collection of n polytopes $W = \{W_1, W_2, \dots, W_n\}$ with disjoint interiors whose union is Q, (4-1) can be written as follows [120]

$$\mathcal{H}(\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_n}, W_1, W_2, \dots, W_n) = -\sum_{i=1}^{n} \int_{W_i} f(\|\mathbf{q} - \mathbf{p}_i\|)\, \varphi(\mathbf{q}) d\mathbf{q} \tag{4-2}$$

where it is assumed that the i^{th} service agent is responsible for the servicing over its *dominance region* W_i. Note that the function $\mathcal{H}(\mathcal{P}, W)$ is to be maximized with respect to both the location of service agents $\mathcal{P}$, and the assignment of the dominance regions W. The optimization is, therefore, to be performed with respect to the position of the agents and the partition of the space. This problem is referred to as a *facility location problem* [119].

Let $V = \{V_1, V_2, \dots, V_n\}$ be the Voronoi partition of $\mathcal{Q}$, for which the service agent positions are the generator points. The Voronoi region of a given service agent, V_i, is the region of points that are closer to that agent than to any other, that is [118]

$$V_i = \{\mathbf{q} \in \mathcal{Q} | \, \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\|, \forall j \neq i\}, i \in \{1, \dots, n\}$$

where norm 2 is used as the distance function. Two service agents V_i and V_j are (Voronoi) neighbors if $V_i \cap V_j \neq 0$. Since f is a non-increasing function, one can easily show that, at fixed location of service agents, the optimal partition of $\mathcal{Q}$ is the Voronoi partition that generates by the position of the service agents, i.e. $\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_n}$. Therefore, we are interested in minimizing $\mathcal{H}(\mathcal{P}, V)$ with respect to the position of the agents.

In particular, we are interested in minimizing [120]

$$\mathcal{H}(\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_n}) = \sum_{i=1}^{n} \int_{V_i} \frac{1}{2} \|\mathbf{q} - \mathbf{p}_i\|^2 \, \varphi(\mathbf{q}) d\mathbf{q} \tag{4-3}$$

as a measure of the performance of system, where $f(\|\mathbf{q}-\mathbf{p}_i\|) = -\frac{1}{2}\|\mathbf{q}-\mathbf{p}_i\|^2$. Note that $\mathcal{H}(\mathcal{P})$ measures the ability of the coverage provided by the network of service agents in $\mathcal{Q}$. Qualitatively, a low value of $\mathcal{H}(\mathcal{P})$ corresponds to a good configuration for coverage of the environment $\mathcal{Q}$. Therefore, it is desired to minimize it.

Each Voronoi region has mass M_{V_i}, and centroid C_{V_i} which are respectively defined as

$$M_{V_i} = \int_{V_i} \varphi(\mathbf{q}) d\mathbf{q} \tag{4-4}$$

$$\mathbf{C}_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} \mathbf{q}\varphi(\mathbf{q}) d\mathbf{q} \tag{4-5}$$

Remarkably, one can show that [120]

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = -\int_{V_i} (\mathbf{q}-\mathbf{p}_i)\varphi(\mathbf{q}) d\mathbf{q} = -M_{V_i}\left(\mathbf{C}_{V_i}-\mathbf{p}_i\right) \tag{4-6}$$

So the partial derivative of $\mathcal{H}$ with respect to the position of the i^{th} service agent only depends on its own position and the position of its Voronoi neighbors. Therefore, the computation of the derivative of $\mathcal{H}$ with respect to the agents' location is decentralized in the sense of Voronoi. It is clear that each partial derivative must be zero for a local minimum.

Clearly, the extremum points of $\mathcal{H}$ are those in which every agent is at the centroid of its Voronoi region, $\mathbf{p}_i = \mathbf{C}_{V_i}, \forall i$. The resulting partition of the environment is commonly called a Centroidal Voronoi Configuration. More thorough discussions were given in [120].

4.3 Problem Statement

This section addresses the cooperative multi agent search and coverage problem in an uncertain environment. Consider the scenario that some search agents are deployed to search and detect some targets in the terrain. There are also service agents that their duty is to spread out over the environment to provide coverage. The search agents broadcast their information about the environment to the service agents. This information allows the service agents to find where in the

environment they are mostly needed and to aggregate in those areas. For the search problem, the environment is discretized in cells that are described by a probability of target existence. There is an uncertainty region corresponding to each target. Each target is assumed to lie somewhere within its uncertainty region, but its exact position is unknown. Each search agent stores a probability map, which contains the probability of existence of all targets in each cell. During the mission, sensors of search agents can detect targets in their footprints. The probability map is updated during the mission based on whether or not the target is detected by the sensors. The objective of the cooperative search mission is to maximize the amount of information about the environment. Therefore, the search mission is the same as the one we discussed in the chapter 3.

The objective of service agents is to spread out over the area to cover the entire environment. However, in most cases, all points in the environment do not have the same level of importance. We can consider a density function which reflects a measure of relative importance of different points in the environment. The density of each point is a decreasing function of the distance between that point and position of the targets. Therefore, points closer to the targets have more value and more level of importance in the environment. Since the information about the position of the targets improves during the search mission, the density function is changed and get more accurate as mission goes on. The number of targets in the environment is known *a priori*. However, their exact position is unknown. Each agent is assigned a unique altitude, therefore, avoiding the need to consider collision avoidance, which is outside the scope of this paper. We assume that each search agent can communicate with all the other agents in the team. Moreover, all service agents can communicate with their neighboring vehicles.

4.3.1 Distribution Density Function

The precise definition of the distribution density function $\varphi(\mathbf{q})$ depends on the desired application. It defines a weight for each point in the environment which is a measure of relative importance of that point. In many applications, there are some critical points and the level of importance of each point in the terrain is inversely proportional to the distance between the point and the critical points. For instance, the critical points can be hotspots in a forest fire or the source of gushing in the oil spill. Let $\varphi(\mathbf{q}) = \sum_{i=1}^{n_c} \phi(\mathbf{q}, \mathbf{q}_c^i)$ where $\mathbf{q}_c^i$ is the i^{th} critical point and n_c is the number of critical points. Function $\phi(\mathbf{q}, \mathbf{q}_c^i)$ is known *a priori* and has a maximum at the critical point $\mathbf{q}_c^i$. Therefore, knowing the exact location of critical points, we can find the weight of all points, i.e. $\varphi(\mathbf{q})$.

In many cases, the location of critical points is not known precisely but it is known that they are located somewhere inside some uncertainty regions. Knowing the probability distribution of each critical point i in its uncertainty region, $P(\mathbf{q}_c^i)$, distribution density function $\varphi(\mathbf{q})$ can be obtained as follows

$$\varphi(\mathbf{q}) = \sum_{i=1}^{n_c} \int_{\Lambda_i} \phi\left(\mathbf{q}, \mathbf{q}_c^i\right) P\left(\mathbf{q}_c^i\right) d\mathbf{q}_c^i \tag{4-7}$$

where Λ_i is the uncertainty region of the i^{th} critical point. Indeed, $\int_{\Lambda_i} \phi(\mathbf{q}, \mathbf{q}_c^i)\, P(\mathbf{q}_c^i) d\mathbf{q}_c^i$ is the expected value of function $\phi\left(\mathbf{q}, \mathbf{q}_c^i\right)$ with respect to $\mathbf{q}_c^i$.

These critical points are in fact the targets of search problem. Since the search is done in a discrete environment, the probability of all points inside a cell is assumed to be equal. Therefore, (4-7) can be modified as follows

$$\varphi(\mathbf{q}) = \sum_{i=1}^{n_c} \sum_{\forall (x,y) \in \Lambda_i} P\left(E_{x,y}^i\right) \int_{q_c^i \in (x,y)} \phi\left(\mathbf{q}, \mathbf{q}_c^i\right) d\mathbf{q}_c^i \tag{4-8}$$

At the beginning of the mission, service agents have *a priori* information of probability distribution of critical points. They use this information to compute distribution density function $\varphi(\mathbf{q})$. Then, they spread out over the environment based on this distribution. During the mission, search agents update the probability maps of critical points and transmit this information to the service agents. Using these updated probability maps, service agents modify their configuration and change their position in the environment.

4.3.2 Distributed Coverage Controller

In this section the coverage control for a group of service agents is presented. Each service agent is modeled as a *double-integrator* point mass moving on a two-dimensional (2-D) plane as follows

$$\ddot{\mathbf{p}}_i = \mathbf{u}_i \tag{4-9}$$

where $\mathbf{u}_i$ is the control input of the i^{th} service agent. Equation of motion of a broad class of vehicles can be expressed by a double-integrator dynamic model. In addition, dynamics of many vehicles can be feedback linearized to double integrators. Following assumptions are used for derivation of distributed coverage controllers in this paper:

Assumption 1. Every agent has complete knowledge of its own dynamics.

Assumption 2. The service agents have the ability to compute their own Voronoi regions in a distributed manner.

Assumption 3. Each service agent can communicate with other service agents in its neighboring Voronoi regions, and all search agents as well.

For the purpose of coordinating multiple service agents to cover a planar environment, the position controller based on the Centroidal Voronoi Configuration is designed. Consider that the

position of the i^{th} service agent is denoted by $\mathbf{p}_i$, and $\mathbf{C}_{V_i}$ is the center of Voronoi that corresponds to the i^{th} service agent. We propose the following position control law for the i^{th} service agent

$$\mathbf{u}_i = k_1^i M_{V_i}\left(\mathbf{C}_{V_i} - \mathbf{p}_i\right) - k_2^i \dot{\mathbf{p}}_i \tag{4-10}$$

where k_1^i and k_2^i are the positive gains.

Theorem 4-1. Consider a group of n service agents whose dynamic models are described by (4-9). Let the Assumptions1 through 3 hold. Under control law (4-10), it is guaranteed that the whole system is asymptotically stable and the planar positions of service vehicles converge to a centroidal Voronoi configuration.

Proof*:* Consider the Lyapunov function candidate as

$$\vartheta = \sum_{i=1}^{n} k_1^i \mathcal{H}_i + \sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i \tag{4-11}$$

Since k_1^i is a positive value, and $\mathcal{H}_i$ is a strictly positive function, then the candidate Lyapunov function ϑ is lower-bounded by zero. Taking the time derivative of ϑ along the trajectories of systems gives

$$\dot{\vartheta} = \sum_{i=1}^{n} k_1^i \dot{\mathcal{H}}_i + 2\sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \ddot{\mathbf{p}}_i \tag{4-12}$$

By substituting $\dot{\mathcal{H}}_i = \dot{\mathbf{p}}_i^T \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}$ into the above equation, one obtains

$$\dot{\vartheta} = \sum_{i=1}^{n} \left[\dot{\mathbf{p}}_i{}^T \left(k_1^i \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} + 2\ddot{\mathbf{p}}_i\right)\right] \tag{4-13}$$

Now, if we substitute (4-6) into the above equation, we have

$$\dot{\vartheta} = \sum_{i=1}^{n} 2\left[\dot{\mathbf{p}}_i{}^T \left(k_1^i M_{\mathbf{V}_i}\left(\mathbf{p}_i - \mathbf{C}_{\mathbf{V}_i}\right) + \ddot{\mathbf{p}}_i\right)\right] \tag{4-14}$$

Finally, by substituting the model of each service agent (4-9) into (4-13) and using control input (4-10), the time derivative of Lyapunov function can be obtained as follows

$$\dot{\vartheta} = 2\sum_{i=1}^{n}\left(-k_2^i\, \dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i\right) \tag{4-15}$$

which is clearly non-positive. Let S be the set of all points in Q where $\dot{\vartheta} = 0$. Due to the convexity of the region Q, one can conclude that each of the Voronoi centroids $\mathbf{C}_{\mathbf{V}_i}$ lies in the interior of the i^{th} Voronoi region and so in the interior of the region Q. So the agents move toward the interior of the region Q and never leave it. Therefore, Q is a positive invariant set for the trajectories of the closed-loop system. Since this set is closed and bounded, one can make use of LaSalle's invariance principle to infer that the planar positions of service vehicles converge to the largest invariant subset of the set S. Suppose a trajectory belongs to the set S. By considering the model of service agents (4-9) and the control law (4-10), we have

$$\dot{\mathbf{p}}_i = 0 \quad \Rightarrow \quad \ddot{\mathbf{p}}_i = 0 \quad \Rightarrow \quad \mathbf{u}_i = 0 \quad \Rightarrow \quad \mathbf{p}_i = \mathbf{C}_{\mathbf{V}_i}, \forall i$$

Then, we can conclude that $\mathbf{p}_i = \mathbf{C}_{\mathbf{V}_i}, \forall i$ is the largest invariant set corresponding to the set of centroidal Voronoi configurations. Therefore, under control law (4-10), the closed-loop system is asymptotically stable and the planar positions of service vehicles converge to a set of centroidal Voronoi configuration.

□

It is worth to mention that although the controller of each service agent only depends on its Voronoi centroid, calculation of the center of Voronoi depends on the neighboring Voronoi region. So each service agent needs to communicate with other service agents in its neighboring Voronoi regions to compute its Voronoi region. The service agent applying the control law (4-10) will move towards the centroid of its Voronoi region. Due to the convexity of region, the centroid is always inside the Voronoi region. Therefore, the Voronoi approach has implicit collision avoidance. In addition, by designing a suitable controller for heights of multiple agents, they can fly at different levels. Then, the collision avoidance can be guaranteed in the entire mission even for the large dimension agents.

4.3.3 Simulation Results

The proposed distributed search and coverage algorithm has been demonstrated via numerical simulations. All simulation have been done in Matlab® R2011a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®. Multi-Parametric Toolbox (MPT) [121] is used to find the Voronoi diagrams. The dynamics are implemented in discrete time with a sampling time of 0.1 s.

The environment used in simulation is a 1 km×1 km square. Since the search problem has discrete nature, the environment is divided into 10000 cells which make a 100×100 square grid. Therefore, each cell is a 10 m×10 m square. There exist three targets known to be in 20×20 cells square areas (uncertainty region) as shown in Figure 4-1, but their exact positions are unknown. The *a priori* probability of existence of these targets is uniformly distributed in their uncertainty region while their real positions are marked by the * marker. It is also considered that a virtual target exists in the environment and its uncertainty region is the whole terrain. Considering this target enforces search UAVs to search the unexplored area of the environment.

A group of three fixed-wing search UAVs and ten quadrotor service UAVs are deployed to search and to cover the environment. Each search UAV is equipped with a sensor that can detect targets in its 4×4 cell footprint. The probabilities of true positive and false positive measurement of sensors are $\gamma = 0.9$ and $\varepsilon = 0.1$, respectively. All three search UAVs start their mission from the south west corner of the terrain, while all service UAVs start their mission from their individual bases which are located on the border of the environment as shown in Figure 4-1. For the purpose of collision avoidance, the UAVs fly in different levels.

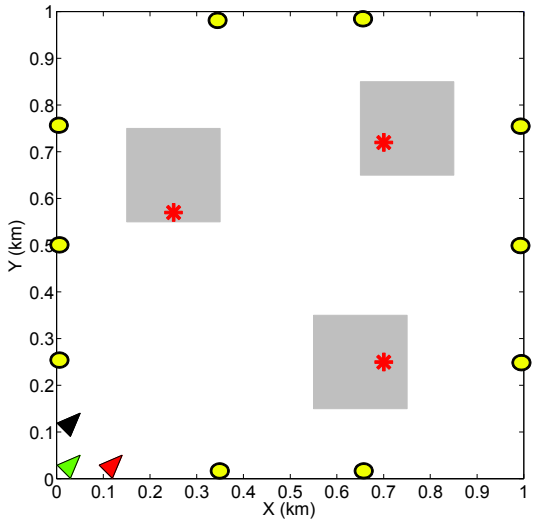


Figure 4-1. The problem environment; the grey rectangles are the uncertainty regions of different targets and * denotes the actual position of the targets. Search UAVs and service UAVs are shown by ⊳ and o markers respectively.

The decision time step for the search UAVs is 5 s. At each decision time step, the search UAVs must decide to go straight, turn 15 degrees left or turn 15 degrees right. We assume that once the search UAV has made a decision about its next action, that action can be performed immediately and then the search UAV continues its mission in a straight path until the next decision time step. The speed of the search UAVs is constant and equal to two units per time step. The search algorithm is similar to the one explained in section 3.4.1.2 and parameters are T=5, $\lambda = 1$, $\delta_k = 1$, K=0.25, and β_τ=5. The model of service UAVs are assumed to be a double integrator and their control law is denoted in (4-9). For all service UAVs, the gain of controllers are $k_1' = k_1 M_{V_i}$=1 and k_2=5.

In this simulation, the following Gaussian density function is used

$$\phi(q, q_c^i) = \frac{1}{\sigma\sqrt{2\pi}}\left(e^{-\frac{(q-q_c^i)^2}{2\sigma^2}}\right) \tag{4-15}$$

where $\sigma = 70\,m$. The initial and final probability maps and their corresponding distribution density functions are shown in Figure 4-2.

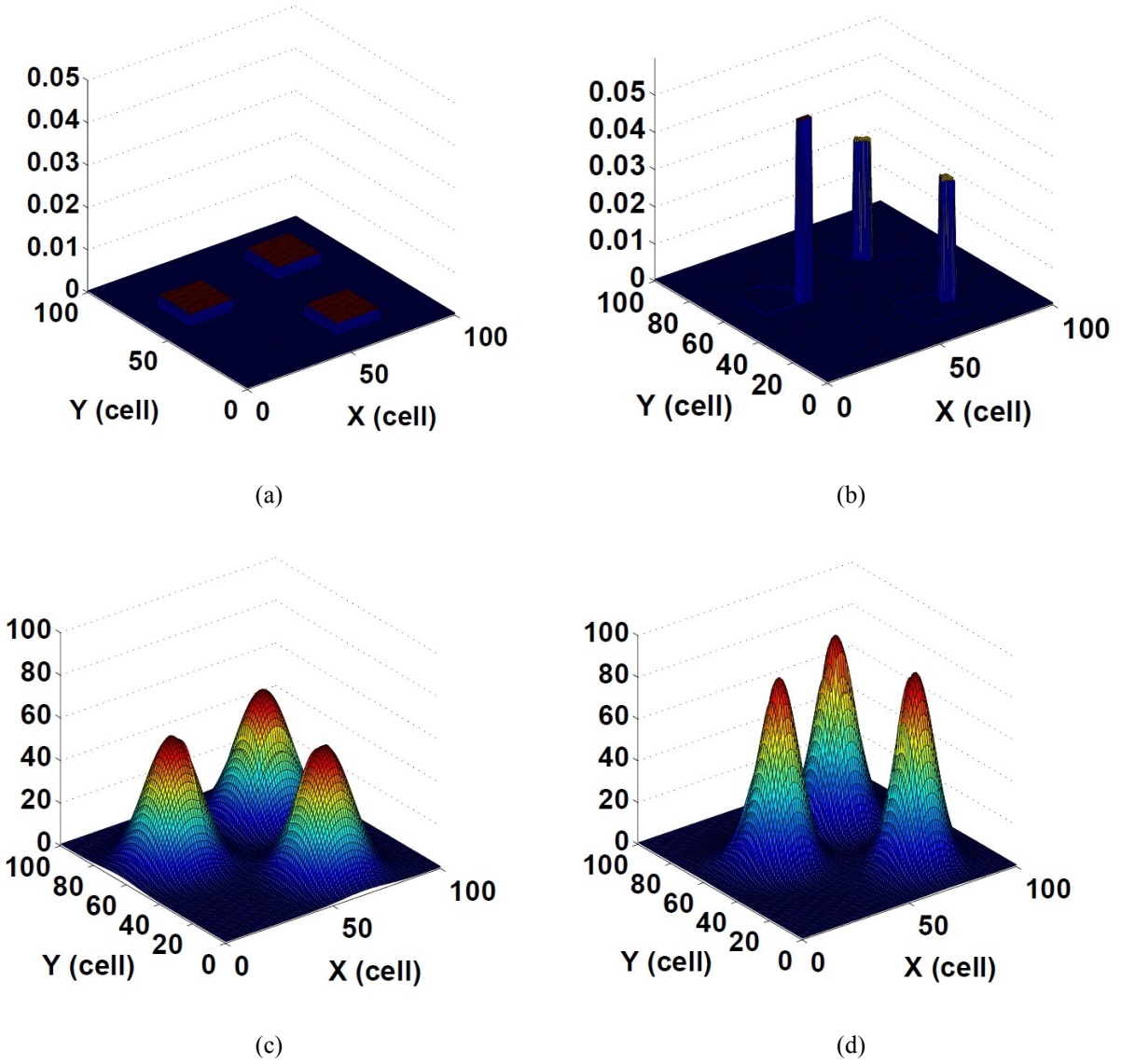


Figure 4-2. (a) the initial probability map, (b) the final probability map,
(c) the initial distribution density function, and (c) the final distribution density function.

We consider the scenario in which, at the beginning, the service agents spread over the terrain based on the imprecise initial distribution density function which is derived from the *a priori* probability maps. The final configuration of planar position and the trajectories of all UAVs are shown in Figure 4-3-a. The exact distribution density function is also shown in the figure. This distribution is calculated based on the actual position of critical points (targets). The color intensity

is proportional to the value of distribution density function at each point. Corresponding distribution density function based on the probability maps is depicted in the Figure 4-3-b. It can be seen from this figure that the configuration of service UAVs in the environment is optimal according to available density function.

Next, search UAVs start their mission to explore the terrain. During the mission, they update the probability maps of the targets and transmit these updated maps to the service agents on a regular basis. The position and trajectory of all UAVs and the exact distribution density function are shown in Figure 4-3-c, e, and g for three different time steps. The position of service UAVs and the corresponding distribution density function based on the most updated probability maps are shown in Figure 4-3-d, f, and h. It is worth to mention that as search UAVs explore the environment, the probability maps get more precise, and, therefore, the current distribution density function gets more similar to the exact one. Especially in Figure 4-3-g and h, the density functions are almost the same in both figures. As expected, deployment of search UAVs helps service UAVs to improve their performance to cover the most needed areas.

In the proposed algorithm, it is assumed that there is no limit on communication between neighboring service agents. To evaluate the effectiveness of the proposed method in more realistic situations where the communication is limited, the above simulation has been repeated with different communication ranges. As a measure of the performance of this method, the value of coverage function $\mathcal{H}$ is reported in Table 1 for five different times, using the exact density function. It can be perceived that the coverage performance is improved about 25% by using this approach in the case of no limit on the communication ranges. As expected, the coverage performance degrades when the communication between service vehicles is limited. However, it can be seen

that the performance degradation is insignificant and the proposed method still improves the coverage performance considerably.

In order to evaluate the average performance of the proposed approach, different simulations have been carried out 25 times and the average number of detected targets and the value of coverage function are depicted in Figure 4-4. The uncertainty regions and actual positions of the targets are randomly chosen for each repetition of simulation. As expected, the value of coverage function decreases and the coverage performance improves when the number of detected targets increases [122].

TABLE 4-1. THE VALUE OF COVERAGE FUNCTION AT DIFFERENT TIMES FOR THE SCENARIOS WITH DIFFERENT COMMUNICATION RANGES

Time (s) / Ranges (m)	*0*	*120*	*160*	*200*	*240*
No limit	2.4339	0.6771	0.6235	0.5948	0.5443
200	2.4339	0.6884	0.6294	0.5989	0.5468
100	2.4339	0.7801	0.6433	0.6127	0.5578
75	2.4339	1.4026	0.8925	0.7127	0.5872
Heterogeneous in [*75*, *200*]	2.4339	0.8810	0.6728	0.6321	0.5692

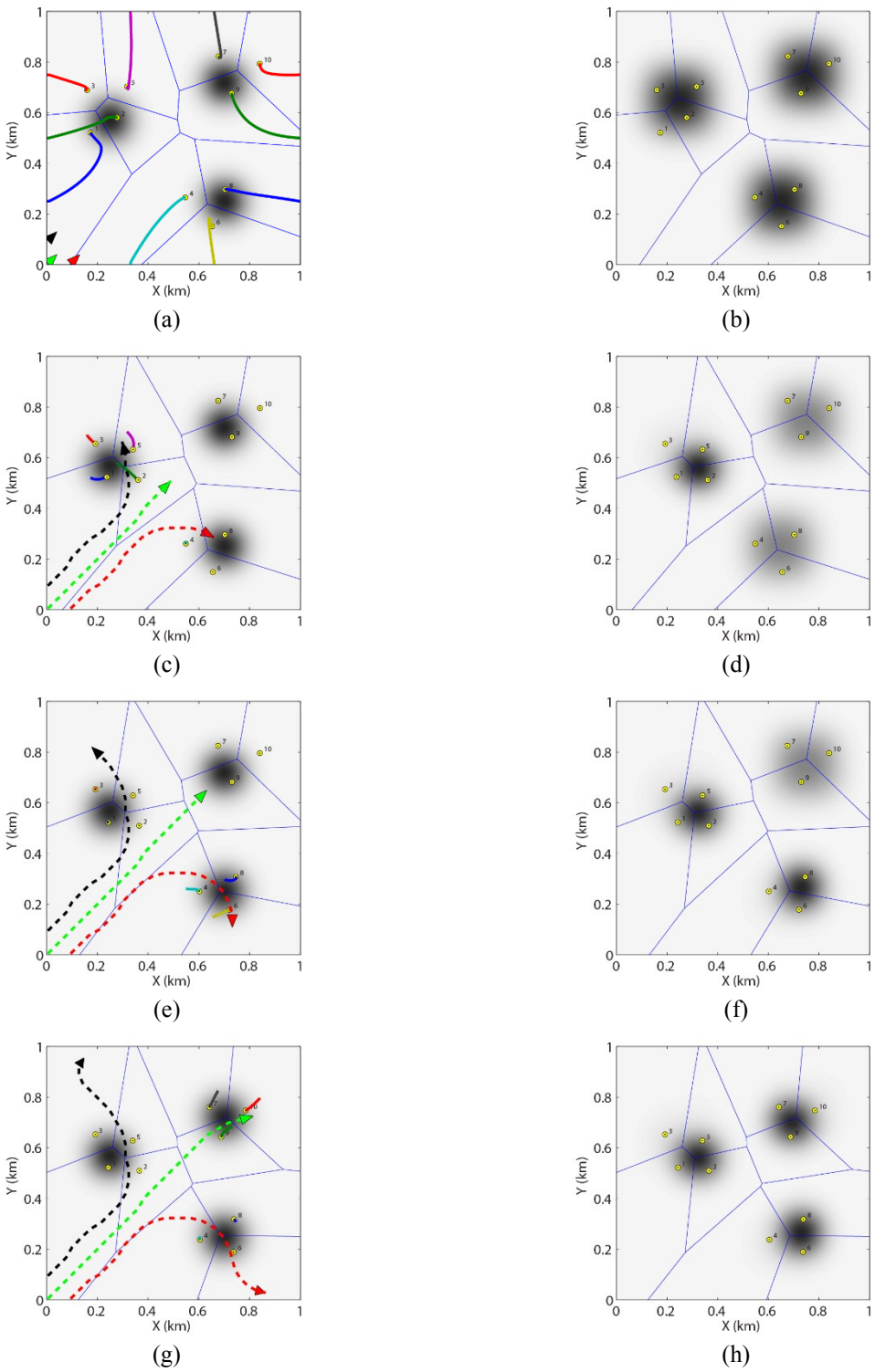


Figure 4-3. Left: The configuration and the trajectories of all UAVs and the exact distribution density function. The color intensity is proportional to the value of density function. Search UAVs and service UAVs are shown by ⊳ and ⊙ respectively. Right: The configuration of service UAVs and the corresponding distribution density function based on the probability maps.
(a) and (b) t=100 sec, (c) and (d) t=160 sec, (e) and (f) t=200 sec, (g) and (h) t= 240 sec

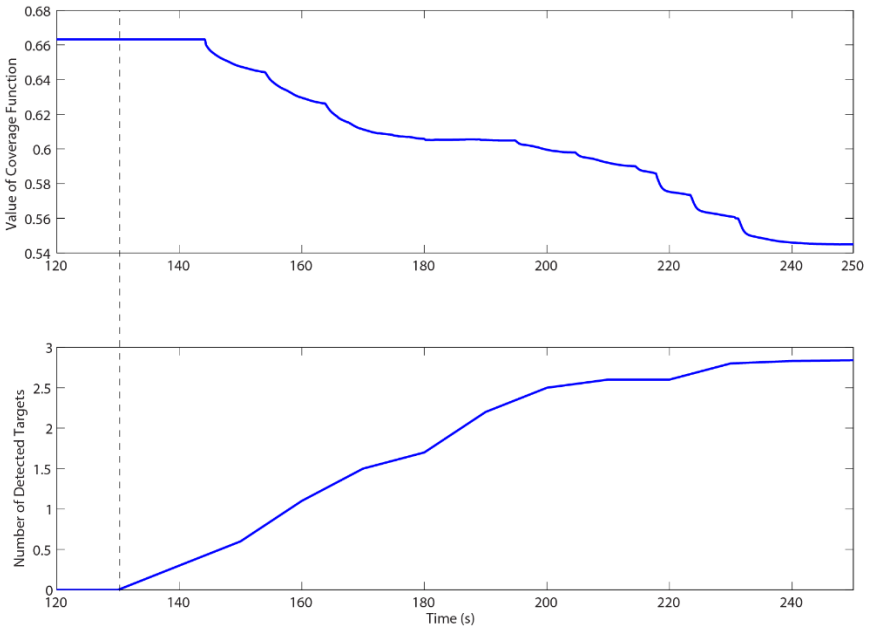


Figure 4-4. The average number of detected targets and the value of coverage function for 25 simulations.

4.3.4 Experimental Results

To demonstrate the effectiveness of presented theoretical developments, an experiment is conducted on a group of unmanned ground vehicles available at the Networked Autonomous Vehicles Lab (NAVL) of Concordia University, which are provided by Quanser [123]. In this experiment, it is considered that the search mission is still carried out by simulation due to the difficulty for flying fixed-wing UAVs in the indoor testing environment and lack of appropriate sensors. The service problem is performed using a network of virtual robots and three available physical Unmanned Ground Vehicles (UGVs). The considered UGVs are equipped with a QuaRC-powered single-board Gumstix embedded computer where QuaRC is the Quanser's Real-time Control software [124]. QuaRC allows rapidly developing and deploying controllers designed in the MATLAB/Simulink® environment for real-time control of the vehicles. Runtime sensors measurement, data logging and parameter tuning are supported between the host computer and the

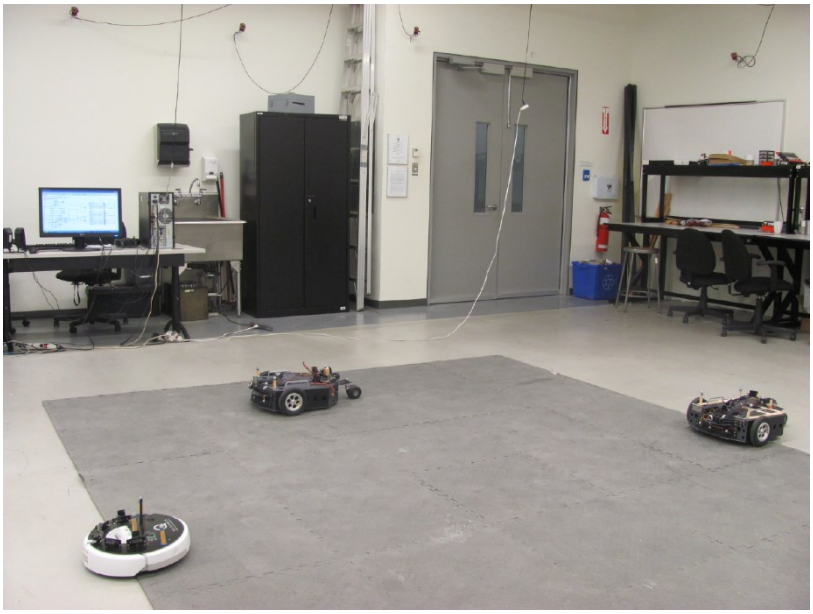


Figure 4-5.The experimental environment with three UGVs.

target vehicles. Since the experiment is taking place indoor in the absence of GPS signals, the system positions are measured by the network of OptiTrack camera systems from NaturalPoint Inc [125]. In Figure 4-5, the experimental environment including the unmanned vehicles, host computer, and network of OptiTrack cameras is illustrated.

The UGVs are differential drive Wheeled Mobile Robots (WMRs), as shown in Figure 4-5.The experimental environment with three UGVs.. They have a low power Gumstix Verdex XL6P 600 MHz on-board computer operated by Linux operating system. The kinematic model of the WMR is as follows

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \omega$$

where $\mathbf{p} = (x, y)$ represents the coordinates of the center of the axle of the actuated wheels on the plane (x, y) and θ is the angle that the longitudinal axis of the robot forms with the axis $\mathcal{X}$. Inputs v and ω are longitudinal velocity and angular velocity of the robot respectively. This

nonholonomic kinematic model can be transformed into a linear controllable system using *Dynamic* (i.e., time-variant) state feedback [126]. This results in a fully linearized model which can be described by a double integral model as follows [126]

$$\ddot{x} = u_x$$
$$\ddot{y} = u_y$$

where the resulting dynamic compensator is

$$v = \xi$$
$$\omega = \frac{u_x \cos(\theta) - u_y \sin(\theta)}{\xi}$$
$$\dot{\xi} = u_x \cos(\theta) + u_y \sin(\theta) \tag{4-16}$$

Therefore, as Theorem 1 suggests, using the following control law for each UGV guarantees that the whole system is asymptotically stable and the planar positions of all service vehicles converge to a centroidal Voronoi configuration

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = k_1 M_V \left(\mathbf{C}_V - \begin{bmatrix} x \\ y \end{bmatrix} \right) - k_2 \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{4-17}$$

Inputs v, and ω, therefore, can be calculated using (4-16). It is worth to mention that the model of virtual service vehicles is still a double integrator and their control law is given by (4-10). For all service vehicles, the gains of the controllers are $k_1' = k_1 M_{V_i}$=1 and k_2=5.

The environment in the experimental setup is similar to the simulation problem. The terrain is a 3m×3m square which is divided to 10000 cells to make a 100×100 square grid. There still exist three targets known to be in their 60cm×60cm uncertainty region but their exact positions are initially unknown. The *a priori* probability of existence of these targets is uniformly distributed in their uncertainty region. It is also considered that a virtual target exists in the environment and its uncertainty region is the entire terrain. The search mission is performed by a group of three virtual vehicles. At each decision time step, search vehicles must decide to go straight ahead, turn

15 degrees left, or turn 15 degrees right. The velocity of search vehicle is equal to 20 cm/s. The search algorithm and parameters are similar to the case in section 4.3.3. Simulation of the search mission for all three vehicles is performed in Matlab® R2012a environment on the host computer which has a dual core 3.2 GHz processor. The dynamic programming algorithm is implemented as a recursive function in Matlab®. Service vehicles include seven virtual vehicles and three physical UGVs. Simulation of virtual vehicles is also performed on the host computer in Matlab® R2012a environment. Multi-Parametric Toolbox (MPT) [121] is used to find the Voronoi diagrams. The dynamics are implemented in discrete time with a sampling time of 0.1 s. Controller of Real UGVs is implemented in Matlab® R2012a environment and uploaded to their on-board processors. The Value of $\mathbf{C}_V$ for the real UGVs is approximated by replacing the integral with a summation. Since the uncertainty value of all points inside a cell is equal, the approximation error is negligible. The position of vehicles is measured using the network of OptiTrack cameras. Host computer then sends the positions of all service vehicles (virtual and real) to the UGVs via the wireless communication channel. The Gaussian density function $\phi\left(q, q_c^i\right)$ is similar to the density function in simulation (4-15) and the standard deviation is equal to $\sigma = 20\ cm$.

At the beginning, service vehicles spread over the terrain based on the imprecise initial distribution density function which is derived from the *a priori* probability maps. After 30 s, the search mission is commenced. The updated probability map is transmitted to the service vehicles every 5 s by the host computer. The final configuration of planar position and the trajectories of all service vehicles for different time steps, 30 s, 50 s, 65 s, 80 s are shown in Figure 4-6. The distribution density function based on the most updated probability maps are also shown in the figures. The color intensity is proportional to the value of distribution density function at each

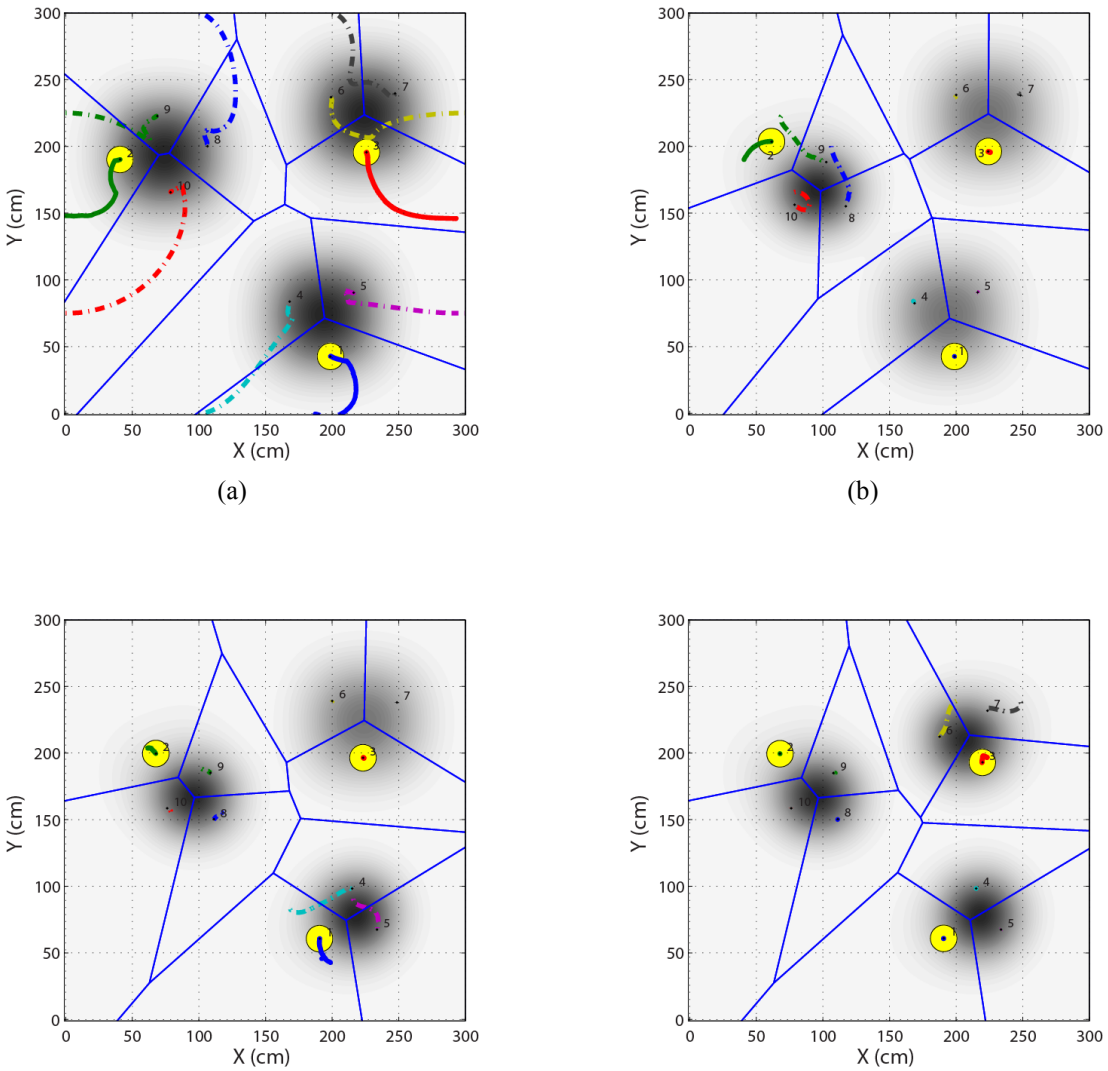


Figure 4-6. Experimental results: The configuration and the trajectories of all service vehicles and the corresponding distribution density function based on the probability maps. The color intensity is proportional to the value of density function. The UGVs are shown by ⊙ marker and their trajectories are shown by solid lines. (a) t=30 s, (b) t=50 s, (c) t=65 s, (d) t= 80 s

point. It can be seen from this figure that the configuration of service vehicles in the environment is optimal according to available density function.

The value of coverage function $\mathcal{H}$, using the exact density function, is shown in Figure 4-7. As expected, the value of coverage function decreases dramatically at time steps 45 s, 60 s, and 75 s when the probability map is considerably improved due to the detection of a new target [127].

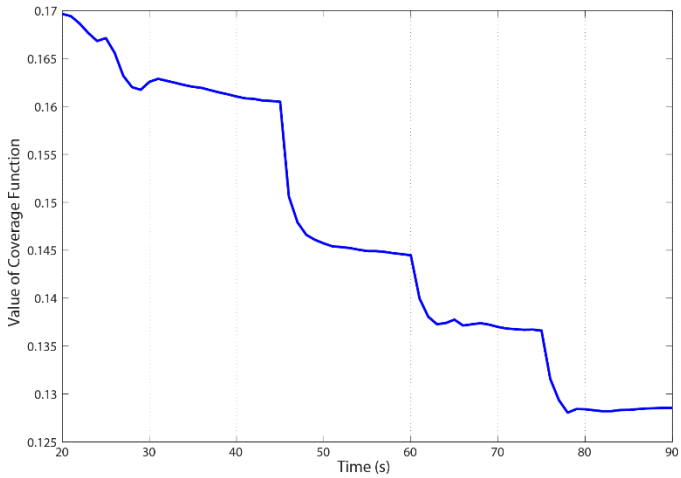


Figure 4-7. The value of coverage function.

4.4 Conclusion

In this chapter, the search and coverage problem in uncertain environments is presented using multiple vehicles. A group of service vehicles are deployed to serve the points or areas where they are most needed in the environment based on the Voronoi partitioning. Since the high service areas are not known beforehand, a group of search vehicles are used to explore the environment using the search algorithm that proposed in chapter 3. This technique leads to covering an uncertain environment more effectively and improving the coverage performance. The proposed approach has been successfully verified by both numerical simulations and experimental tests.

CHAPTER 5

Cooperative Search and Coverage using Dynamic Programming

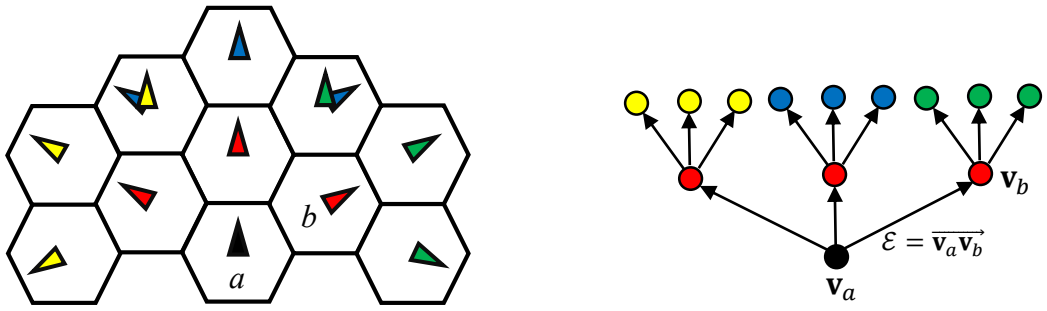
In this chapter, we investigate the search and coverage problem with a single type of agent. A team of agents are responsible to explore the environment to gather new information and at the same time concentrate around more important parts of the environment to provide service, if it is necessary. We first introduce the notion of *limited turn-rate Voronoi diagram*. The search and coverage problem is then formulated as a multi-objective optimization problem with different constraints including minimum fuel consumption, refuelling, obstacle avoidance, and collision avoidance.

5.1 Limited Turn-rate Voronoi Diagram

When the distance metric in making the Voronoi diagram is (weighted) Euclidean distance, the domain must be convex. The geodesic Voronoi partition is proposed in [128], which uses the geodesic distance to make Voronoi partitions in non-convex environments. Its discrete counterpart is introduced in [129] which uses the Dijkstra's shortest path method to construct the Voronoi diagram. However, that method is not useful when the agents have limited turn rate, which means they cannot change their heading immediately. To address this issue, we propose the limited turn-rate Voronoi diagram.

To construct the limited turn-rate Voronoi partition, we first define a weighted digraph that each node of the graph corresponds to a cell of the environment with a specific discretized heading. There is an edge $\mathcal{E} = \overrightarrow{\mathbf{v}_a\mathbf{v}_b}$ between the node $\mathbf{v}_a$ and its neighbor $\mathbf{v}_b$ if the agent is able to move from the position corresponds to $\mathbf{v}_a$ to the position that corresponds to $\mathbf{v}_b$. It is worth to mention that $\mathbf{v}_a$ and $\mathbf{v}_b$ may both correspond to one cell with different headings. A cost $c(\mathcal{E})$ is associated with every edge $\mathcal{E}$. Definition of the cost depends on the application and is mainly related to the time and fuel consumption of transition from the *predecessor* node to *successor* node.

Figure 5-1 shows how this digraph can be constructed. The left panel shows 2 steps of a search mission. It is assumed that at each time step, the agent can go straight to the next cell, turn 60 degrees to the right and go to the next cell, or turn 60 degrees to the left and go to the next cell. The starting position of the agent is a which includes both position and heading of the agent. Different positions (i.e. location and heading) of the agent at the next time step are shown with similar colors. The right panel shows the corresponding digraph. For example, since the agent can go from the position a to the position b, there is an edge between their corresponding nodes (i.e. $\mathbf{v}_a$ and $\mathbf{v}_b$).



(a) (b)

Figure 5-1. a) 2 steps of a search mission and b) its corresponding digraph.

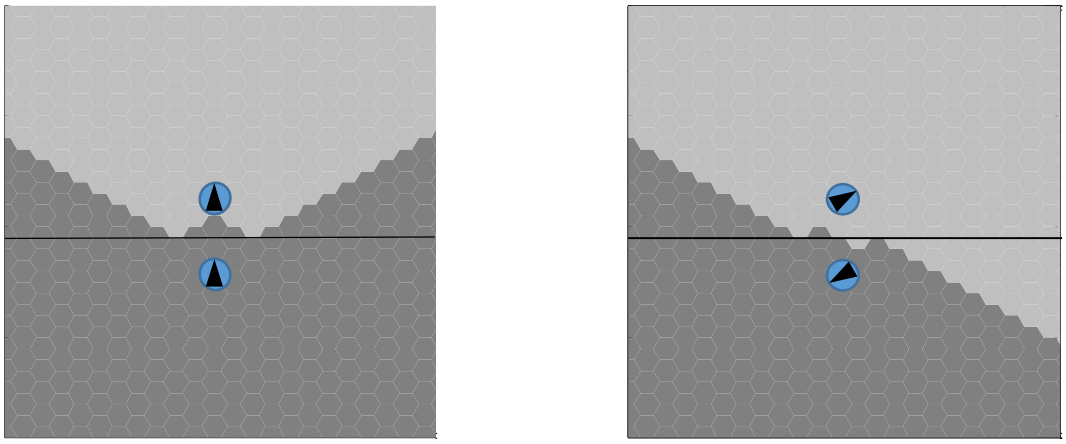
To find the Voronoi partitions, we use a modified version of Dijkstra's shortest path for digraphs [130] for all agents, which starts from the node that corresponds to the current position and heading of the agent and expands for all accessible nodes. The total cost of transition from one node to another is the summation of cost of all edges in the shortest path between two nodes. The cost of transition from the i^{th} agent position and heading to any other cell is the minimum cost between the corresponding node of the current position and heading of the agent and all corresponding nodes of that cell with different headings. In another word, if θ_i is the heading of the i^{th} agent, h_q is the h^{th} heading of the cell $\mathbf{q}$, and $C\left((\mathbf{p}_i,\ \theta_i),(\mathbf{q},h_q)\right)$ is the cost of the shortest path between the node corresponds to the current position and heading of i^{th} agent and the node corresponds to the heading h_q of the cell $\mathbf{q}$, then $d\big((\mathbf{p}_i,\ \theta_i),\mathbf{q}\big)=\min_{h_q} C((\mathbf{p}_i,\ \theta_i),(\mathbf{q},h_q))$, where $d((\mathbf{p}_i,\ \theta_i),\mathbf{q})$ is the cost of the shortest path between the agent $\mathbf{p_i}$ and the cell $\mathbf{q}$.

Therefore, when the shortest paths algorithm is performed for all agents, each cell of the environment has the associated costs $d((\mathbf{p}_i,\ \theta_i),\mathbf{q})\ ,\forall\ i=1,2,\dots,n_a$, where $d((\mathbf{p}_i,\ \theta_i),\mathbf{q})$ gives the cost of the shortest path between the current position and heading of the i^{th} agent, $(\mathbf{p}_i,\ \theta_i)$, and the cell $\mathbf{q}$. The number of agents in the environment is n_a. The Voronoi region of a given agent, V_i, is the region of the cells closer to that agent than to any other. Thus

$$V_i=\left\{\mathbf{q}\in\mathcal{Q}\middle|d((\mathbf{p}_i,\ \theta_i),\mathbf{q})\le d(\big(\mathbf{p}_j,\ \theta_j\big),\mathbf{q}),\forall j\neq i,j\in\{1,\dots,n_a\}\right\}$$

It should be noticed that in the limited turn-rate Voronoi diagram, *generators* are pairs of position and heading, i.e. $(\mathbf{p}_i,\ \theta_i)$.

When the environment is large relative to the range of the mobile agents, we may define *limited range and turn-rate Voronoi diagram*. The set $B((\mathbf{p}_i,\ \theta_i),r_i)$ is defined as the collection of nodes in the graph which are reachable from the node corresponds to the current position and



(a) (b)

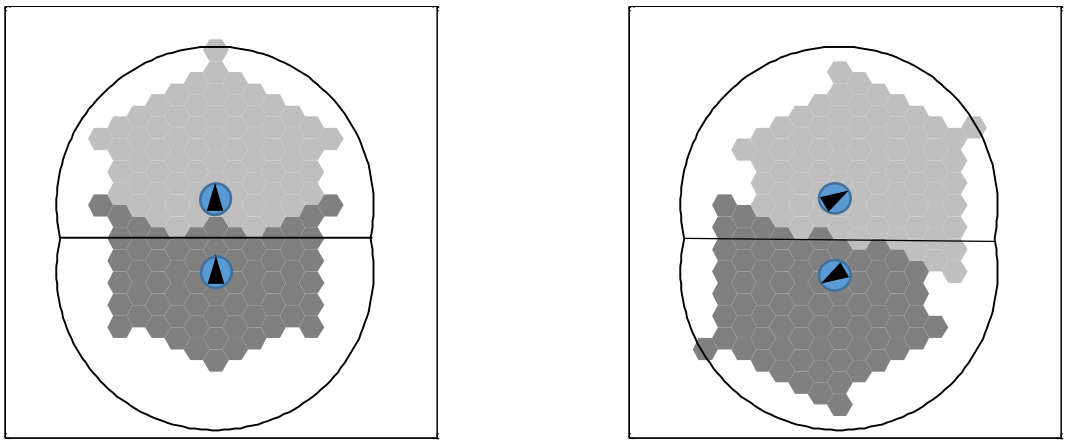
Figure 5-2. Limited turn-rate Voronoi regions are shown with different colors. Heading of agents is shown by ⊳. In (a) and (b) position of agents are the same but they have different headings. Voronoi region of both agents is also shown.

heading of i^{th} agent in a *walk* of less than r_i nodes. The limited range and turn-rate Voronoi region of the i^{th} agent then can be defined as

$$V_i = \{\mathbf{q} \in Q \cap B((\mathbf{p}_i, \theta_i), r_i) |\, d((\mathbf{p}_i, \theta_i), \mathbf{q}) \leq d((\mathbf{p}_j, \theta_j), \mathbf{q}), \forall j \neq i, j \in \{1, \dots, n_a\}\}$$

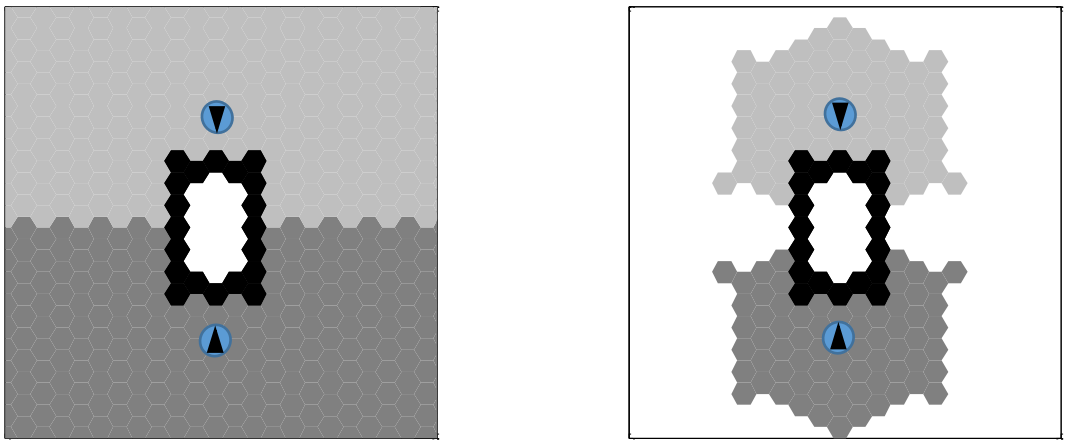
In Figure 5-2, the limited turn-rate Voronoi diagram of environment with two agents is shown. The environment is discretized with hexagonal cells. At each time, each agent is able to move straight ahead, turn 60 degrees left or turn 60 degrees right. It can be seen that in both figures, the Voronoi diagrams are the same while the limited turn-rate Voronoi diagrams are different due to different heading of the agents.

In Figure 5-3, the limited range and turn-rate Voronoi diagram of the environment as well as its limited range Voronoi diagram are shown. The structure of environment and the position and the heading of agents in this figure are the same as Figure 5-2. In Figure 5-4, the limited turn-rate Voronoi diagram of an environment that includes some obstacles is shown.



(a) (b)

Figure 5-3. Limited range and turn-rate Voronoi regions are shown with different colors. Heading of agents is shown by ⊳. In (a) and (b) position of agents are the same but they have different headings. Limited rang Voronoi region of both agents is also shown.



(a) (b)

Figure 5-4. Limited turn-rate Voronoi regions in presence of obstacles in the environment are shown with different colors. Heading of agents is shown by ⊳. In (b) range of Voronoi is limited.

5.2 Problem Statement

There are a team of agents which are equipped with appropriate sensors to detect some targets in an unknown environment. The agents cooperatively explore the environment to find the targets

while avoiding the obstacles. The environment is discretized with hexagonal cells. It is assumed that at each time step, an agent can move from its current cell to one of its neighbors, based on its current heading. All agents also have some proximity sensors that enable them to detect obstacles in some or all of their neighboring cells.

The agents should maintain a *probability map* of the targets. We define $P\left(E_{\mathbf{q}}^{i}\right)$ as the probability of existence of the target i in the cell $\mathbf{q} = (x, y)$. The probability map consists of the probability of existence of all targets in any given cell in the environment. At the beginning of the mission, the probability map is initialized based on the *a priori* information about the position of the targets. The sensors are assumed to be non-ideal which means they may miss an existing target (*false negative*) or report a target while it does not actually exist (*false positive*). During the mission, the probability map is updated based on the output of mobile sensors using the updating rules provided in (2-12).

The agents also maintain an *occupancy map* of the obstacles which represents the probability of presence of an obstacle in each cell of environment. This map is initialized based on *a priori* information about the location of obstacles. When there is no initial information. The probabilities of presence and non-presence of an obstacle in all cells are equal, therefore, $P\left(E_{\mathbf{q}}^{o}\right) = 0.5, \forall \mathbf{q} \in Q$, where $P\left(E_{\mathbf{q}}^{o}\right)$ is the probability that the cell $\mathbf{q}$ is occupied by an obstacle. During the mission, if an agent detects an obstacle in one cell, that cell is considered as an obstacle, i.e. $P\left(E_{\mathbf{q}}^{o}\right) = 1$ and the agents should never move to that cell.

5.3 Objectives of the Mission

In this section we investigate different objectives of the search and coverage mission.

5.3.1 Environment Exploration

The main objective of the mission is exploring the whole environment to decrease the uncertainty about the position of the targets in the environment. Each cell $\mathbf{q} = (x, y)$ is associated with an uncertainty value, $\zeta(\mathbf{q}, t) \in \mathbb{R}^+$. The collection of uncertainty values of all cells in the environment constructs the *uncertainty* map of the environment. Each cell in the environment is initialized with $\zeta(\mathbf{q}, 0)$ based on *a priori* knowledge available about that cell. If all cells are equally uncertain, then they all are initialized with $\zeta(\mathbf{q}, 0) = \zeta_0 \geq 0$. As the time goes on, $\zeta(\mathbf{q}, t)$ increases by a factor of η ($\eta \geq 1$) to show that the current information about the cell is no longer up-to-date.

Each sensor scan about the environment obtained during the search is a source of evidence about the state of that location. We consider a case of imperfect sensors in this study, that is, each sensor scan does not by itself provide 100% certainty about the state of the corresponding location. Here, we define an *uncertainty reduction rate*, denoted as $\mu_i \in (0,1)$, to model the uncertainties and inaccuracies about the i^{th} mobile sensor. Mathematically, μ_i quantifies the belief of a sensor scan from agent i committed to reducing the uncertainty in that cell. Since the agents are identical in this study, we use μ to denote the uncertainty reduction rate for all the agents. When an agent visits a cell, its uncertainty value decreases by the factor of μ indicating that the cell is recently visited and the knowledge of the team about that cell is up-to-date. Using Dempster's rule of combination [116], we define the uncertainty updating rule as follows

$$\zeta(\mathbf{q}, t+1) = \begin{cases} \mu\, \zeta(\mathbf{q}, t) & \exists i, \mathbf{p}_i(t) = \mathbf{q} \\ \eta\, \zeta(\mathbf{q}, t) & \text{otherwise} \end{cases} \tag{5-1}$$

where $\mathbf{p}_i$ is the position of the i^{th} agent. The value of μ is inversely proportional to the accuracy of sensors. When a cell is searched by an accurate sensor, the uncertainty about that cell decreases considerably which can be translated into a low value for the factor μ. It is easy to see that the first

scan of a cell results in the maximum reduction in uncertainty and further scans result in reduced benefit. Therefore, this update rule is a simple way to track the number of useful "looks" each cell has had and captures the nature of diminishing returns with each look. Value of η defines how dynamic the environment is. The high value of η means the environment is very dynamic and the status of a cell can be changed very quickly. When $\eta = 1$, it means the environment is static and the status of the cells never changes. Therefore, exploration objective of the mission is to minimize the total value of uncertainty in the whole environment in the minimum time.

5.3.2 Environment Coverage

Although the main objective of the mission is to explore the environment to gather more information about it, one could also be interested in exploiting that information to concentrate the agents around the targets. In fact, the objective of environment coverage is to distribute the agents across the environment while aggregating in more important areas. The function $\varphi(\mathbf{q})$ is a distribution density function that defines a weight for each cell. This function reflects a measure of relative importance of different regions in terrain. Therefore, the higher the value of $\varphi(\mathbf{q})$ the more attention the group has to pay to $\mathbf{q}$. The precise definition of the distribution density function $\varphi(\mathbf{q})$ depends on the desired application. Inspired from many real applications, in this study, it is assumed that the level of importance of each cell in the terrain is inversely proportional to the distance between the cell and the targets. Let $\varphi(\mathbf{q}) = \sum_{i=1}^{m} \phi(\mathbf{q}, \mathbf{T}_i)$, where $\mathbf{T}_i$ is the position of the i^{th} target and m is the number of targets. Function $\phi(\mathbf{q}, \mathbf{T}_i)$ is known *a priori* and is a decreasing function of the distance between cell $\mathbf{q}$ and the position of the i^{th} target, $\mathbf{T}_i$, and has a maximum at the point $\mathbf{T}_i$.

The location of the targets in the environment is not known precisely. The only information available about the position of the targets is the probability of existence of the targets in different

cells in the environment, i.e. $P\left(E_{\mathbf{q}}^{i}\right)$. Therefore, the distribution density function $\varphi(\mathbf{q})$ can be obtained as follows

$$\varphi(\mathbf{q}) = \sum_{i=1}^{m} \sum_{\forall \mathbf{T}_i \in \mathrm{Q}} P\left(E_{\mathbf{q}}^{i}\right) \phi(\mathbf{q}, \mathbf{T}_i) \tag{5-2}$$

Indeed, $\sum_{\forall \mathbf{T}_i \in \mathrm{Q}} P\left(E_{\mathbf{q}}^{i}\right) \phi(\mathbf{q}, \mathbf{T}_i)$ is the expected value of function $\phi(\mathbf{q}, \mathbf{T}_i)$ with respect to $\mathbf{T}_i$.

Let $V = \{V_1, V_2, \dots, V_n\}$ be the Voronoi partition of $\mathcal{Q}$, for which the position of agents are the generator points. The Voronoi region of a given agent, V_i, is the region of points that are closer to that agent than to any other, that is

$$V_i = \left\{ \mathbf{q} \in \mathcal{Q} |\, d(\mathbf{q} - \mathbf{p}_i) \leq d(\mathbf{q} - \mathbf{p}_j), \forall j \neq i, j \in \{1, \dots, n\} \right\}$$

where $d(.)$ is a distance function. We define the coverage function

$$\mathcal{H}(\mathcal{P}) = \sum_{i=1}^{n} \sum_{\forall \mathbf{q} \in V_i} d(\mathbf{q}, \mathbf{p_i})\, \varphi(\mathbf{q}) \tag{5-3}$$

as a measure of the performance of coverage, where $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p_n}\}$. In (5-3), it is assumed that the i^{th} agent is responsible for its Voronoi region V_i. This measures the ability of the coverage provided by the network of agents. Qualitatively, a low value of $\mathcal{H}$ corresponds to a good configuration for coverage of the environment. Therefore, the coverage objective of the mission is to minimize the value of coverage function.

5.3.3 Coordination

In our approach, the agents share their information about the environment, but they make individual decisions about what action to take next. It is desired to impede different agents from heading to the same destination at the same time so as to reduce the possible overlap. One way to address this issue is to treat the paths of other agents as soft obstacles as we did in chapter 3. In this section we take another approach. We use limited turn-rate Voronoi region to optimally assign different regions of environment to the agents. Then, each agent is only responsible for the cells in its Voronoi region. This approach guarantees that two agents never try to search the same cell

in the environment at the same time. In fact, each cell may only be searched by the closest agent to it. In addition, if the dimension of agents is less than the size of a cell, this approach guarantees collision avoidance.

5.3.4 Communication

At each time step, agents communicate their current position and sensor measurements. Therefore, all agents can maintain identical uncertainty maps, probability maps, and occupancy maps. If agents have limited communication ranges, they cannot communicate with the other agents who are not in their communication range. One can add a constraint to the optimization problem of the i^{th} agent to force it to stay in communication range of other agents; i.e. $\|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j, \forall j \neq i$, where R_j is the communication range of the j^{th} agent. When agents are able to work as repeaters and to relay the information they received from another agent to other agents in their communication range, the constraint can be relaxed as $\exists j \neq i \; s.t. \|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j$. In case the agents have limited communication range but they do not consider the communication range constraint in their optimization problem, they may not be able to maintain the same maps which could decrease the total performance of mission. When an agent loses its communication with other agents, it continues its mission without considering the actions and measurements of other agents. Thus, its actions may not optimize the total objective of multi agent mission and its map will not be the same as other agents. If the agent is able to gain its communication with other agents again, although it can communicate with other agents and use their information to update its maps in the future, these maps will not be the same as other agents due to information lost during the communication loss. However, if the communication channel of the agents is large enough, they might be able to transfer the information they have gathered during the

communication loss, once they gain the communication back and use an appropriate data fusion algorithm to make new identical maps which is beyond the scope of this work.

5.3.5 Obstacle Avoidance

Another important issue in the mission is to impede the agents from entering possible "forbidden zones" or colliding with obstacles. The most common way to address obstacles is "rivalling force" or "potential function" methods [131]. In fact, the obstacle avoidance issue is not usually considered a part of decision making process. Instead, the low-level controller takes care of this issue which may decrease the performance of mission. In this work, in the process of constructing the limited range Voronoi partitions, we exclude the obstacles from the environment. Therefore, the obstacles are not in the Voronoi region of the agents which guarantees the obstacle avoidance of all agents. As it has already been stated, all agents maintain an occupancy map which represents the current information of team about the position of obstacles. The obstacles can be detected when an agent is located in their adjacent cell with appropriate headings (in all simulation it is assumed that the agents can only detect an obstacle in the cell right in front of them). This occupancy map can ultimately reveal the structure of the environment. Although finding the structure of the environment may not be one of the primary objectives of the mission, it is a by-product of obstacle avoidance feature of our approach.

5.3.6 Fuel Management

One of the practical challenges in unmanned missions is minimizing the agents' fuel consumption and safe returning of the agents to the base for refuelling. Using the limited turn- rate Voronoi partition guarantees that a cell in the environment is not assigned to an agent unless it has minimum cost to visit the cell among all available agents. It divides the environment between neighbouring agents based on the cost of reaching to each cell. Therefore, each cell in the

environment belongs to the Voronoi region of the agent who can reach that cell with the least cost. Since the cost of each edge in the graph is assumed to be an increasing function of the amount of fuel that agents consume to traverse that edge, this approach can minimize the total fuel consumption.

The refuelling issue is studied in the literature [132]. The main condition is that at each time step, the agent at least must have enough fuel to return to its base. In this work, $f_i(t)$ is defined as the remaining fuel of the i^{th} agent which is a decreasing function of time and depends on its previous path. To guarantee the safe returning of an agent to the base, it is necessary that cost of reaching the base(s) always be less than the remaining fuel, i.e. $C_i(\mathbf{B}_i) = d((\mathbf{p}_i, \theta_i), \mathbf{B}_i) \leq f_i(t), \forall t \geq 0$, where $\mathbf{B}_i$ is the base of the i^{th} agent. Finding $C_i(\mathbf{B}_i)$ for all agents at each time step is computationally expensive and it is not feasible in the large environments. To solve this problem, we use internal geodesic distance to approximate the cost of returning to the base. The internal geodesic distance is the length of shortest path between two points which is entirely contained in the environment. This path may consist of sequence of segment $\{\overrightarrow{\mathbf{q}\mathbf{q}_1}, \overrightarrow{\mathbf{q}_1\mathbf{q}_2}, \dots, \overrightarrow{\mathbf{q}_{r-1}\mathbf{q}_r}, \overrightarrow{\mathbf{q}_r\mathbf{B}_i}\}$ where $\mathbf{q}_i$ is a reflex vertex which has an internal angle greater than 180°. The algorithm to find the internal geodesic distance is presented in [*133*]. To further simplify the process of finding the cost of return-to-the base, the agent can find a path to the base by following the straight line which connects it to the base. Whenever this path crosses an obstacle it should continue along the obstacle such that it eventually gets closer to the base. As soon as it passes the obstacle, it will follow the straight line again. The second method is less optimal than the internal geodesic distance method but it needs much less computation. We call this method approximate internal geodesic distance. A typical example of the path using both methods is shown in Figure 5-5.

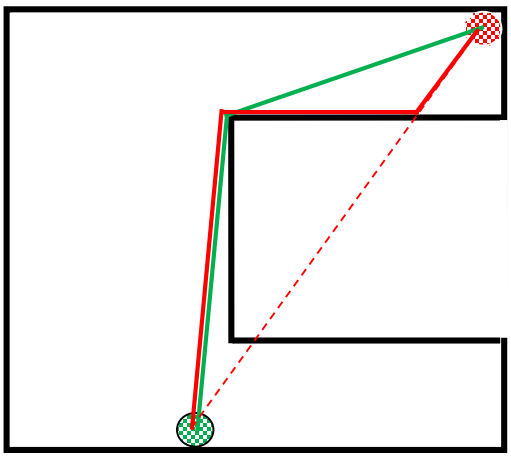


Figure 5-5. Internal geodesic distance: green and approximate internal geodesic distance: red

The shortest path is produced based on the current information about the structure of the environment. This information may be inaccurate especially at the beginning of the mission. Therefore, the calculated cost should be scaled up by an uncertainty factor $\mu_f \geq 1$ that reflects the lack of information about the true structure of the environment. This factor is proportional to the complexity of the environment and can be a decreasing function of time to incorporate more accurate information about the structure of the environment as time goes on. Finding a reasonable uncertainty factor μ_f is a trade-off between efficiency and health of agents. A conservative choice for μ_f (a large value) guarantees the safe returning of the agents to the base but it might decrease the efficiency by unnecessarily sending the agents for refueling. Choosing a small value for μ_f (close to 1) might lead to the event that some agents run out of fuel before reaching to the base.

5.3.7 Multi-objective Mission

The multi-objective single step cost function of the i^{th} agent at time step k can be defined as

$$g(\mathbf{p}_i, \theta_i, k) = -w_e g_e(\mathbf{p}_i, \theta_i, k) + w_c g_c(\mathbf{p}_i, \theta_i, k) \tag{5-4}$$

where $g_e = \lambda^k \zeta(\mathbf{p}_i, k)$ is the exploration value, $g_c = \lambda^k \sum_{\forall \mathbf{q} \in V_i} d(\mathbf{q}, \mathbf{p_i})\, \varphi(\mathbf{q})$ is the coverage value, λ ($0 < \lambda \leq 1$) is a time discount factor, and $w_e > 0$ and $w_c > 0$ are weights of each value

function. At each time step, each agent can find its optimal action by solving the following optimization problem

$$\min_{u_i} J_k(\mathbf{p}_i, \theta_i, u_i) \tag{5-5}$$

Such that

$$\begin{cases} \mathbf{p}_i \in V_i \\ \|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j, \forall j \neq i \\ C_i(\mathbf{B}_i) \leq f_{\ i}(k) \end{cases} \tag{5-6}$$

where u_i: $(\mathbf{q}_i, \theta_i, k) \rightarrow (\mathbf{q}_i, \theta_i, k+1)$ is the control input, and

$$J_k(\mathbf{p}_i, \theta_i, u_i) = \ g(\mathbf{p}_i, \theta_i, k) + J_{k+1}(\mathbf{p}_i, \theta_i, u_i) \tag{5-7}$$

is the "cost-to-go" from time step k to the end of the mission. The second condition can be replaced by $\|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j, \exists j \neq i$, if the agents can act as repeaters and relay the received information to the other agents in their neighborhood.

Assume that the mission duration is N time steps, i.e. the agent will make N decisions over the course of the mission. Thus, the terminal cost of the mission is $J_N(\mathbf{p}_i, \theta_i, u_i)$ and is achieved at the end of the mission. The complete cost-to-go for any time step k is then found by iterating enough times until the terminal cost is reached. However, as the dimension of the problems, i.e. possible states to examine over the planning horizon of the entire mission, grows, the computation time grows exponentially for this approach. In order to make the problem feasible, a rolling horizon limited look-ahead policy can be utilized. It makes the problem tractable, but at a cost of optimality. This rolling horizon approximation defines a horizon of time steps T, and then replaces the final cost J_N with the value of J_{k+T}. This produces a much smaller problem space, so it has the benefit of always producing a tractable result. However, this solution is optimal with respect to the sub-problem and may not be optimal in respect to the main problem. There is a trade-off between optimality and computational complexity here.

When the optimization problem has no feasible solution, it means condition $C_i(\mathbf{B}_i) \leq f_{\ i}(k)$ does not hold anymore. That is because it is always possible for all agents to only change their heading and stay at the same cell that guarantees the conditions $\mathbf{p}_i \in V_i$ and $\|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j, \forall j \neq i$. Therefore, when there is no feasible solution for the optimization problem, the agent is no longer able to continue its normal mission and it should switch to the refueling mode. In that mode, it must skip all other tasks and heads immediately to the refueling base. At each time step, the agent who needs to refuel can find its optimal action by solving the following optimization problem

$$\min_{u_i} C_i(\mathbf{B}_i) \tag{5-8}$$

Such that

$$\begin{cases} \mathbf{p}_i \in V_i \\ \|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j, \forall j \neq i \end{cases} \tag{5-9}$$

where $C_i(\mathbf{B}_i)$ is the cost of reaching the base of the i^{th} agent. When $C_i(\mathbf{B}_i)$ is calculated by using (approximate) internal geodesic distance, the optimal solution to this optimization problem is simply the one that follows the "shortest path to the base" defined by that distance. The condition $\mathbf{p}_i \in V_i$ is needed to guarantee the collision avoidance. However, the definition of "cost of an edge" is different for the agents in the refueling mode. To construct the Voronoi partition in this case, the agent who needs to refuel, assigns minimum cost (typically zero) to all edges on its way to the refueling base, i.e. on the shortest path to the base, and assigns maximum cost (typically very high value) to all other edges. The rest of the process is the same as always, and each cell belongs to the Voronoi of the agent with minimum cost.

Communication constraint $\|\mathbf{p}_i - \mathbf{p}_j\| \leq R_j, \forall j \neq i$ may prevent the agent from returning to the refueling base on time which leads to the loss of the agent. Therefore, in practice when the above optimization problem has no feasible solution which means it is not possible for the agent

to maintain its minimum distance from other agents while having enough fuel to return to the base, the agents should switch to the following optimization problem

$$\min_{u_i} C_i(\mathbf{B}_i) \tag{5-10}$$

Such that

$$\mathbf{p}_i \in V_i \tag{5-11}$$

Of course, in this case, the agent will lose its communication with others but it may safely return to the base.

5.4 Uncertainty-weighted Voronoi

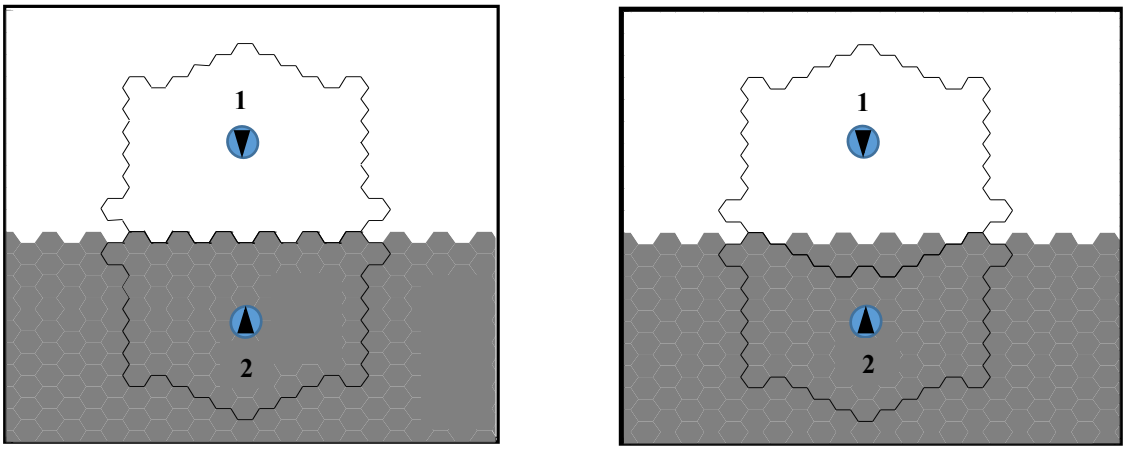
We divided the optimization problem into two stages; finding the optimal partitioning of the environment and then finding the optimal action of each agent in its region. The limited turn-rate Voronoi diagram divides the environment between two adjacent agents based on the cost of reaching to each cell. Therefore, each cell in the environment belongs to the Voronoi region of the agent who can reach that cell with the least cost which guarantees the coordination between agents and the fuel efficiency of the mission. However, this partitioning may not be optimal from exploration point of view.

In Figure 5-6, the lower half of the environment is uncertain while the upper half has no uncertainty. The limited turn-rate Voronoi region of the *1st* agent is completely located in the upper half and the Voronoi region of the *2nd* agent is located in the lower half of the environment. All cells inside the Voronoi region of the *1st* agent are closer to it than to the *2nd* agent and, therefore, it is more efficient for the *1st* agent to search those cells rather than the *2nd* agent. However, those cells have no uncertainty and there is no point for any agent to search them. On the other hand, all

uncertain cells belong to the *2nd* agent, therefore, in this extreme situation the performance of the mission is like the performance of a mission with a single agent.

We are interested in partitioning the environment based on the uncertainty value. To take into account the value of uncertainty function in constructing the Voronoi partitions, we modify the definition of cost function. The cost $c(\mathcal{E})$ which is associated with the edge $\mathcal{E} = \overline{\mathbf{v}_a\mathbf{v}_b}$ now consists of two terms; a fixed term $c_o(\mathcal{E})$ which depends on the application and is mainly related to the time and fuel consumption of the transition from the *predecessor* node to the *successor* node, and a variable term $\frac{\zeta(\mathbf{v}_a,t)+\zeta(\mathbf{v}_b,t)}{2}$, where $\zeta(\mathbf{v},t)$ is the uncertainty associated with the cell that corresponds to the node $\mathbf{v}$ at the time t. It is worth to mention that all graph nodes that correspond to the one cell of the environment have the same value of uncertainty. Therefore, the cost of an edge $\mathcal{E} = \overline{\mathbf{v}_a\mathbf{v}_b}$ can be defined as

$$c(\mathcal{E}) = w_e \frac{\zeta(\mathbf{v}_a,t)+\zeta(\mathbf{v}_b,t)}{2} + w_f c_o(\mathcal{E}) \tag{5-12}$$

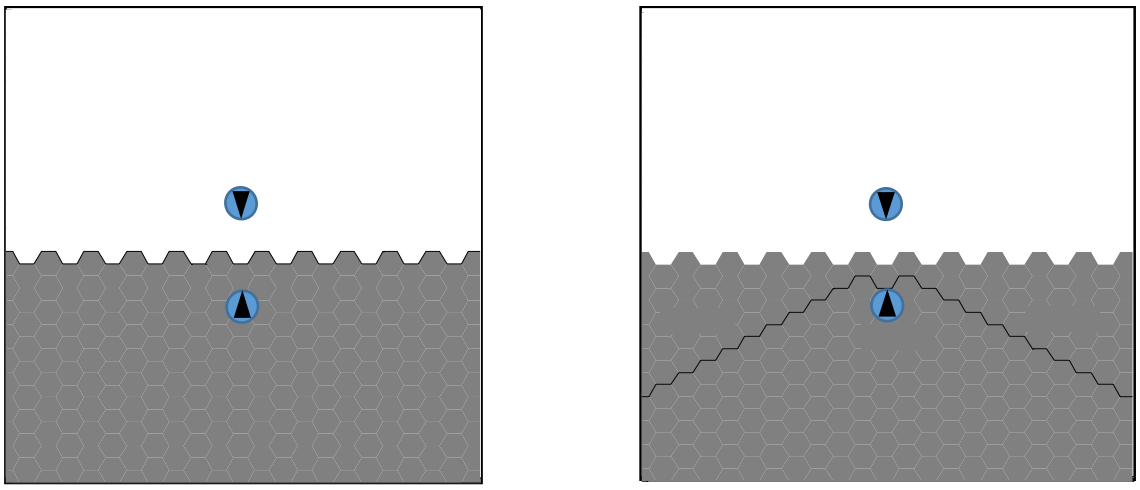


(a) (b)

Figure 5-6. a) Limited range and turn-rate Voronoi diagram. b) Uncertainty-weighted limited range and turn-rate Voronoi diagram. The color intensity is proportional to the value of uncertainty.

where $w_f > 0$ is the weight of full efficiency. When $w_e = 0$, the Voronoi diagram divides the cells between the agents to minimize the fuel consumption (or time-to-reach). When $w_f = 0$, the Voronoi diagram uniformly divides the uncertainty between the agents. Using this cost, the procedure of finding the Voronoi partition is the same as previous case.

Figure 5-6-b shows the uncertainty-weighted limited range and turn-rate Voronoi diagram. It can be seen that the Voronoi region of the I^{st} agent also includes some cells with high uncertainty. In Figure 5-7, the uncertainty-weighted limited turn-rate Voronoi with $w_e = 0$ and $w_f = 0$ are shown respectively. In can be seen that when $w_e = 0$, the environment is equally divided between two agents. When $w_f = 0$, the total value of uncertainty in both Voronoi regions is almost the same.



(a) (b)

Figure 5-7. Uncertainty-weighted limited turn-rate Voronoi diagram. The color intensity is proportional to the value of uncertainty. a) $w_e = 0$ b) $w_f = 0$

5.5 Simulation Results

The proposed distributed algorithm has been demonstrated via numerical. All simulation have been done in Matlab® R2012a environment on a PC with 2.4 GHz CPU. The dynamic programming algorithm is implemented as a recursive function in Matlab®.

The environment is discretized into hexagonal cells. There are three agents which are responsible to search and cover the environment. At each time step, each agent can decide to go straight ahead to the adjacent cell, turn left, or turn right. In fact, each agent in a cell has six possible headings which can lead to one of its six neighbours. Cost of transition from one cell to its neighbour is assumed to be equal to 1, and cost of changing heading (turning right or left) is assumed to be 0.5. These values are used by the agents to construct the Voronoi partitions and to calculate the remaining fuel.

There are four indistinguishable targets in the environment that their number and their positions are not known by the agents. It is assumed that no prior information about the position of the targets is available. Therefore, initial probability map is built such that the probability of presence of the target in all cells is set to be 0.5 which reflects the complete uncertainty about the environment. All agents also store an uncertainty map. The initial uncertainty of all cells considered to be equal to 1. The value of μ and η are assumed to be 0.5 and 1.01 respectively. Since there is no information about the possible position of obstacles at the beginning, occupancy map is initialized with zero which means there is no obstacle in the environment. When an agent visits a cell it can measure if there is a target in the cell. Probabilities of *true positive* and *false positive* measurement of sensors are set to be 0.9 and 0.1 respectively. Agents can also detect an obstacle in the cell right in front of them and use that information to update their occupancy map.

For the coverage purpose, we used the following Gaussian density function

$$\phi(q,p) = \frac{1}{\sigma\sqrt{2\pi}}\left(e^{-\frac{(q-p)^2}{2\sigma^2}}\right)$$

where $\sigma = 4$. The distance function $dist(q, p_i)$ in calculation of coverage function $\mathcal{H}$ is assumed to be Euclidian distance.

There is a refuelling base located at the starting point of the targets that they should return to it to refuel when it is necessary. Approximate geodesic method is used to find the distance between agents and the refuelling base. The scale factor μ_f is set to be 1.2 to take into account the lack of information about the true structure of the environment which means the agents always have at least 20 percent back-up fuel. Since it is considered that each agent needs five time steps to refuel, $C_i(Base)$ is increased by 5 units. The initial amount of fuel of agents is equal to 150 units.

In order to execute the simulations in a reasonable amount of time, we set the look-ahead horizon of the Dynamic Programming algorithm to 4-time steps and use walks of less than 4 nodes to construct the limited range Voronoi diagram.

The simulation has been performed for environments with three different structures which are depicted in Figure 5-8. Obstacles are shown by black cells. The blue cells are the position of refuelling bases. Three agents start their mission from their refuelling bases. Simulation has been done 50 times for each structure while the targets are randomly placed in the accessible area of the environment.

First, the communication range of agents is assumed to be big enough to ensure the uninterrupted communication between agents at all times. Each agents chooses its next action by solving optimization problem (5-5) with constraints (5-6) where J_k is defined in (5-7). The exploration gain w_e =0.9, the coverage gain w_c =0.1, and the time discount factor λ=1. When this

optimization problem has no feasible solution, the agents switches to the refuelling mode by solving optimization problem (5-8) with constraints (5-9). The average number of truly detected targets and total uncertainty of the environment for these 150 simulations are shown in Figure 5-9 and Figure 5-10, respectively. To evaluate the efficiency of the refuelling method, the same 150 simulations have been repeated without considering the refuelling constraint. In fact, the initial fuel of agents is set to be very high (1000 units) which ensures that agents never need to refuel during the simulation interval. The average number of truly detected targets and total uncertainty of the environment in this case are also shown in Figure 5-9 and Figure 5-10, respectively. It can be seen that the performance does not considerably reduce when the agents have limited fuel and they need to refuel in comparison with the case that agents have enough fuel to explore the environment without refuelling. Figure 5-11 shows some snapshots of a typical mission at different times. When the probability of existence of a target in a cell is above 0.9, the target is considered as found and it is shown by a green cell.

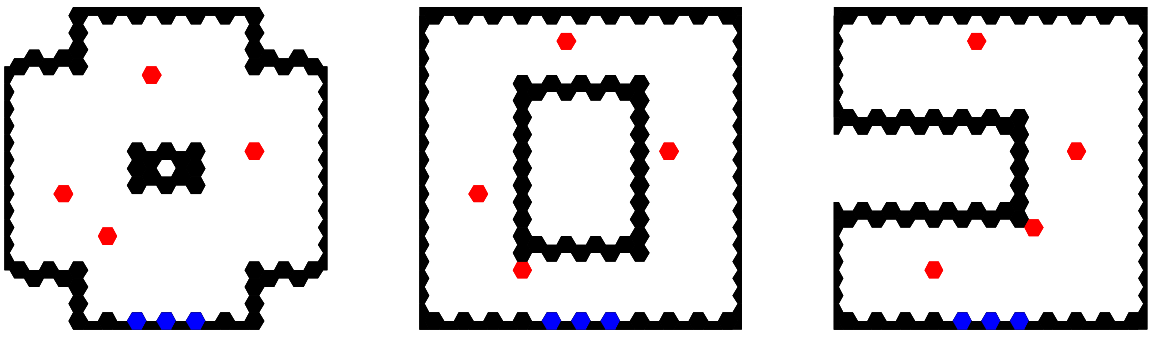


Figure 5-8. Three different structures that used in simulation studies. Red cells are targets, black cells are obstacles, and the blue cells are the starting points.

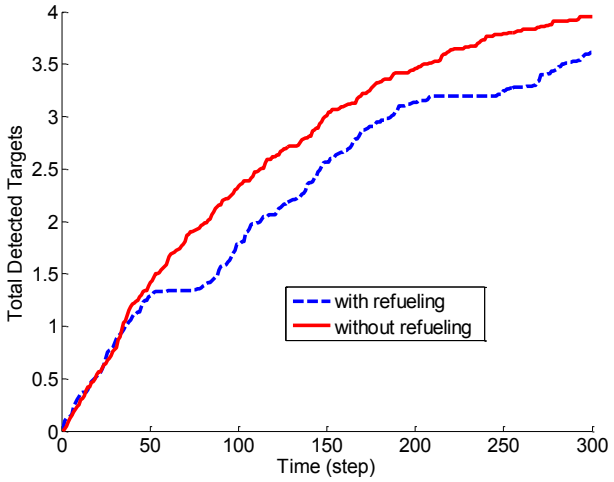


Figure 5-9. The average number of truly detected targets for 150 simulations.
Blue dashed line: with refueling; Red solid line: without refueling

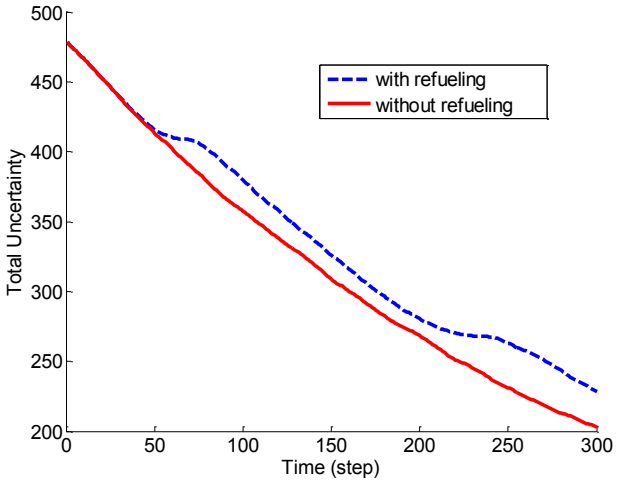


Figure 5-10. The average value of uncertainty for 150 simulations.
Blue dashed line: with refueling; Red solid line: without refueling

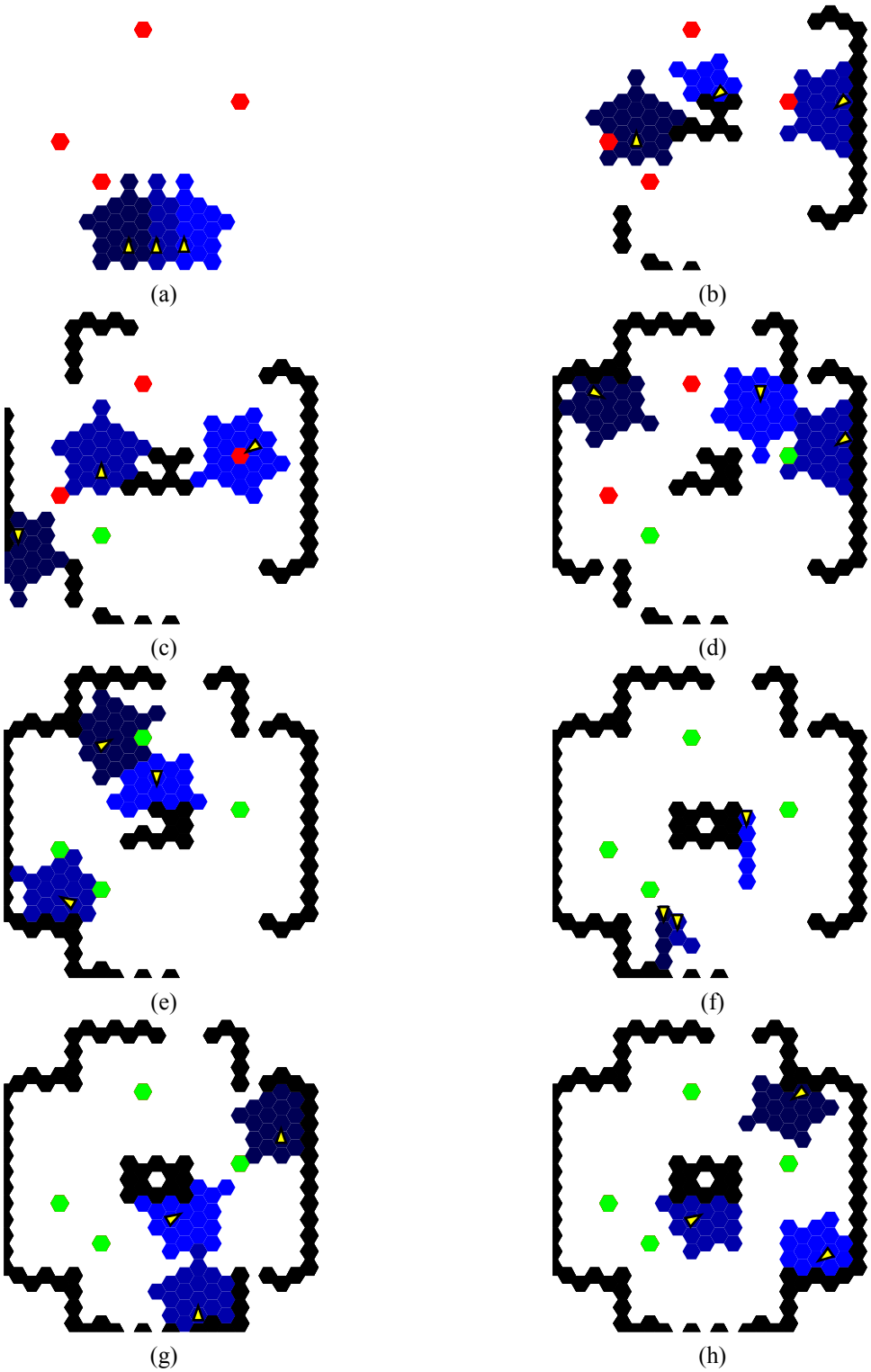


Figure 5-11. The snapshots of a typical mission at different times;
Limited range and turn-rate Voronoi of agents are shown by different shades of blue.
Red cells: undetected targets, Green cells: detected targets, Black cells: obstacles, Marker ▻ : agents
a) t= 0 b) t=30 c) t=60 d) t=90 e) t=120 f) t=150 g) t=180 h) t=210

In Figure 5-12, the number of truly detected targets in three cases has been compared; the case where there is no limit in communication, the case where the commination range of all agents is 20 and they use communication constraint in their decision process, and finally the case that the communication range is 20 but the agents do not consider the communication constraint in their optimization problem. 50 simulations have been performed for each structure and the position of the targets was randomly chosen. In order to make comparison more clear, the initial fuel of agent is set to 1000 to prevent the agents from refuelling. It can be seen that having limitation in communication decreases the performance of the mission. When the communication is limited but the agents do not try to remain in the communication range of each other, the performance is very low. In fact, the performance in this case is comparable to the performance of single agent missions.

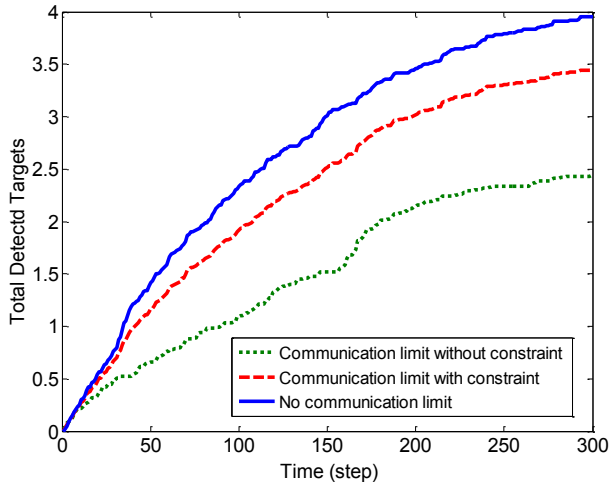


Figure 5-12. The average number of truly detected targets for 150 simulations; Blue Solid line: No communication limit;
Red dashed line: Communication limit with constraint;
Green dotted line: Communication limit without constraint

5.6 Conclusion

In this chapter, the problem of search and coverage in uncertain environments is investigated. First, the limited turn rate Voronoi diagram is introduced. The search and coverage problem is formulated as a multi-objective optimization problem with different constraints including minimum fuel consumption, refuelling, obstacle avoidance, and collision avoidance. Despite the method used in the previous chapter, there is only one type of agents which perform both search and coverage tasks. The problem of "multi agent search" and the problem of "multi agent coverage" are special cases of this problem.

CHAPTER 6

Voronoi-Based Cooperative Search

In this chapter, we introduce a Voronoi-based search strategy for a team of mobile agents with limited range sensors. One of the main differences between the approach in this chapter and what we discussed in chapters 3 and 5 is that the new approach combines the mid-level and low-level controller. It considers a double integral model for the agents and provides a low-level control law for each agent that constantly decreases the amount of uncertainty in the environment.

6.1 Sensors and Uncertainty

It is assumed that the model of sensors is Bernoulli-type. The probabilities of *true positive* and *false positive* measurement of sensors are assumed to be $\gamma = P(D|E)$ and $\varepsilon = P(D|\bar{E})$ respectively, where γ is the probability of detecting a target and ε is the probability of reporting a target existence while it does not exist. These two parameters are specifications of sensors and assumed to be known *a priori*. The value of γ and ε is assumed to depend on the distance between the sensor and the observed point. We assume a second-order polynomial function of $r = \|\mathbf{q} - \mathbf{p}\|$ model for γ, where $\mathbf{p}$ is the location of sensor and $\mathbf{q}$ is the point being observed as follows

$$\gamma(\mathbf{q}) = \begin{cases} \frac{\gamma_0 - 0.5}{{r_\gamma}^2}({r_\gamma}^2 - r^2) + 0.5 & r \le r_\gamma \\ 0.5 & r > r_\gamma \end{cases} \tag{6-1}$$

where γ_0 $(0.5 \le \gamma_0 \le 1)$ is the peak value of γ at the observation point. Parameter r_γ is the range

of sensor. This model indicates that the sensor's detection probability is maximum at its position and monotonically decreases with the distance. The probability is equal to 0.5 outside of sensing range implying that it is equally likely for sensor to truly detect a target or miss it outside the sensing range. A similar model is also assumed for *true negative* measurement, $1-\varepsilon = P(\bar{D}|\bar{E})$. Therefore, the model of *false positive* measurement ε is as follows

$$\varepsilon(\mathbf{q}) = \begin{cases} \frac{\varepsilon_0 - 0.5}{{r_\varepsilon}^2}({r_\varepsilon}^2 - r^2) + 0.5 & r \le r_\varepsilon \\ 0.5 & r > r_\varepsilon \end{cases} \tag{6-2}$$

where $\varepsilon_0 (0 \le \varepsilon_0 \le 0.5)$ is the bottom value of ε at the observation point and r_ε is the range of sensor. The probability update rule for a sensor with this model is discussed in section 1.1.2.1.

For any point in the environment, the probability of true measurement of the point by the sensor can be calculated using the total probability law as follows

$$\begin{aligned} P(T) &= P(D|E)P(E) + P(\bar{D}|\bar{E})P(\bar{E}) \\ &= \gamma P(E) + (1-\varepsilon)\big(1 - P(E)\big) \end{aligned} \tag{6-3}$$

If we assume that the capability of sensor in making *true positive* and *true negative* measurements is the same, i.e. $\gamma = 1-\varepsilon$ and $r_\gamma = r_\varepsilon$, then $P(T) = \gamma$ is the probability of true measurement of that point. After each measurement by the sensor, the certainty about the status of each point is increased by the factor of the probability of true measurement of sensor at that point, $P(T)$, and decreased by the factor of probability of false measurement of sensor at that point, $P(\bar{T})$. Therefore, the total certainty about the status of each point is increased by $\mu = P(T) - P(\bar{T})$. One can simply show that

$$\mu(\mathbf{q}, \mathbf{p}) = \begin{cases} \mu_0({r_\gamma}^2 - \|\mathbf{q} - \mathbf{p}\|^2) & \|\mathbf{q} - \mathbf{p}\| \le r_\gamma \\ 0 & \|\mathbf{q} - \mathbf{p}\| > r_\gamma \end{cases} \tag{6-4}$$

where $\mu_0 = 2\frac{\gamma_0 - 0.5}{{r_\gamma}^2}$. The lack of information about the environment can be modeled as an uncertainty density function $\phi: Q \longmapsto [0,1]$. If the uncertainty density of a given point $\mathbf{q}$ at the time

step n is denoted by $\phi_n(\mathbf{q})$ and the position of the i^{th} agent is denoted by $\mathbf{p}_i$, the uncertainty density of the point at the next time step changes as follows

$$\phi_{n+1}(\mathbf{q}) = \phi_n(\mathbf{q}) \min_i(1 - \mu(\mathbf{q}, \mathbf{p_i})) \tag{6-5}$$

where it is assumed that at any given point, only the measurement from the agent which can reduce the uncertainty by largest value is incorporated. The sensor fusion for multiple mobile sensors is beyond the scope of this work. Since the function $\mu(\mathbf{q}, \mathbf{p_i})$ is a decreasing function of $\|\mathbf{q} - \mathbf{p_i}\|$ and each agent is closer to the points in its Voronoi cell than any other agent, the minimum value of $1 - \mu(\mathbf{q}, \mathbf{p_i})$ occurs when $\mathbf{q}$ belongs to $\mathbf{V_i}$, i.e. $\mathbf{q} \in \mathbf{V_i}$. Therefore, the uncertainty density of the point $\mathbf{q}$ can be updated as follows

$$\phi_{\mathrm{n+1}}(\mathbf{q}) = \phi_n(\mathbf{q})\,(1 - \mu(\mathbf{q}, \mathbf{p_i})), \quad \mathbf{q} \in \mathbf{V_i} \tag{6-6}$$

The uncertainty reduction from time step n to time step $n + 1$ is as follows

$$\Delta\,\phi_n(\mathbf{q}) = \phi_{n+1}(\mathbf{q}) - \phi_n(\mathbf{q}) = \phi_n(\mathbf{q})\,\mu(\mathbf{q}, \mathbf{p_i}),\ \mathbf{q} \in \mathbf{V_i} \tag{6-7}$$

The objective of mission is to maximize the total uncertainty reduction in each time step. Thus, the following performance function must be maximized

$$H = \int_Q \Delta\,\phi(\mathbf{q}).\,d\mathbf{q} = \sum_i \int_{\mathbf{V_i}} \phi(\mathbf{q})\,\mu(\mathbf{q}, \mathbf{p_i}).\,d\mathbf{q} \tag{6-8}$$

We define $\mathbf{V}' = \{\mathbf{V_1}', \mathbf{V_2}', \dots, \mathbf{V_n}'\}$ as the limited-range Voronoi diagram with the range of r_γ as follows

$$\mathbf{V_i}' = \{\mathbf{q} \in Q \cap B(\mathbf{p_i}, r_\gamma) |\ \|\mathbf{q} - \mathbf{p_i}\| \le \|\mathbf{q} - \mathbf{p_j}\|, \forall j \neq i\}, i \in \{1, \dots, n\}$$

Where $B(\mathbf{p_i}, r_\gamma)$ is a circle with the center of $\mathbf{p_i}$ and the radius of r_γ. The limited-range Voronoi diagram is not necessarily a partition and some points in the environment may not belong to any Voronoi region.

Since $\mu(\mathbf{q}, \mathbf{p_i})$=0 if $\|\mathbf{q} - \mathbf{p_i}\| > r_\gamma$, the performance function can be shown as

$$H = \sum_i \int_{\mathbf{V_i}'} \phi(\mathbf{q})\,\mu_0({r_\gamma}^2 - \|\mathbf{q} - \mathbf{p_i}\|^2).\,d\mathbf{q} \tag{6-9}$$

Therefore, the total performance is the sum of performance of all mobile agents, i.e. $\mathcal{H} = \sum_{i=1}^{n} \mathcal{H}_i$, where the performance of each agent is equal to

$$\mathcal{H}_i(\mathbf{p_i}) = \bar{\mathcal{H}}_i - \bar{\bar{\mathcal{H}}}_i \tag{6-10}$$

where

$$\bar{\mathcal{H}}_i = \int_{\mathbf{V_i}'} \phi(\mathbf{q})\, \mu_0 r_\gamma{}^2 d\mathbf{q} \tag{6-11}$$

and

$$\bar{\bar{\mathcal{H}}}_i = \int_{\mathbf{V_i}'} \phi(\mathbf{q})\, \mu_0(\|\mathbf{q} - \mathbf{p_i}\|^2).\, d\mathbf{q} \tag{6-12}$$

Since the value of $\bar{\mathcal{H}}_i$ is not a function of the position of mobile agent, the maximum of $\mathcal{H}_i$ occurs when the value of $\bar{\bar{\mathcal{H}}}_i$ is minimum. The partial derivative of $\bar{\bar{\mathcal{H}}}_i$ with respect to the position of mobile agent is equal to

$$\frac{\partial \bar{\bar{\mathcal{H}}}_i}{\partial \mathbf{p_i}} = -2\mu_0 \int_{\mathbf{V_i}'} \phi(\mathbf{q})\, (\mathbf{q} - \mathbf{p_i}).\, d\mathbf{q} \tag{6-13}$$

It can be shown that

$$\frac{\partial \bar{\bar{\mathcal{H}}}_i}{\partial \mathbf{p_i}} = -2\mu_0 M_{\mathbf{V_i}'}(\, \mathbf{C}_{\mathbf{V_i}'} - \mathbf{p_i}) \tag{6-14}$$

Where $M_{\mathbf{V_i}'}$ and $\mathbf{C}_{\mathbf{V_i}'}$ are mass and centroid of the limited-range Voronoi diagram of the i^{th} agent respectively

$$M_{\mathbf{V_i}'} = \int_{\mathbf{V_i}'} \phi(\mathbf{q})\, d\mathbf{q} \tag{6-15}$$

and

$$\mathbf{C}_{\mathbf{V_i}'} = \frac{\int_{\mathbf{V_i}'} \phi(\mathbf{q})\, \mathbf{q} d\mathbf{q}}{M_{\mathbf{V_i}'}} \tag{6-16}$$

The extremum points of $\mathcal{H}$ are those in which every agent is at the centroid of its limited-range Voronoi region, i.e. $\mathbf{p_i} = \mathbf{C}_{\mathbf{V_i}'}, \forall i$.

6.2 Control Strategy

Each mobile agent is modeled as a *double-integrator* point mass moving on a two-dimensional (2-D) plane as follows

$$\ddot{\mathbf{p}}_i = \mathbf{u}_i \tag{6-17}$$

Equation of motion of a broad class of vehicles can be expressed by a double-integrator dynamic model. In addition, dynamics of many vehicles can be feedback linearized to double integrators.

We propose the following position control law for the i^{th} agent

$$\mathbf{u}_i = -k_1^i M_{\mathbf{V}_i'}\left(\mathbf{p}_i - \mathbf{C}_{\mathbf{V}_i'}\right) - k_2^i \dot{\mathbf{p}}_i \tag{6-18}$$

where k_1^i and k_2^i are the positive gains.

Theorem 6-1: Consider a group of *n* agents whose dynamic models are described in (6-17). Under control law (6-18), it is guaranteed that the whole system is asymptotically stable and the planar positions of agents converge to the centroid of their limited-range Voronoi region.

Proof: Consider the Lyapunov function candidate as

$$\vartheta = \frac{1}{\mu_0}\sum_{i=1}^{n} k_1^i \bar{\bar{\mathcal{H}}}_i + \sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i \tag{6-19}$$

where k_1^i is the positive controller gain. Since k_1^i is a positive value, $0 < \mu_0$, and $\bar{\bar{\mathcal{H}}}_i$ is also a strictly positive function, then the candidate Lyapunov function ϑ is lower-bounded by zero. Taking the time derivative of ϑ along the trajectories of systems gives

$$\dot{\vartheta} = \frac{1}{\mu_0}\sum_{i=1}^{n} k_1^i \dot{\bar{\bar{\mathcal{H}}}}_i + 2\sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \ddot{\mathbf{p}}_i \tag{6-20}$$

By substituting (6-14) into the above equation, one obtains

$$\dot{\vartheta} = \sum_{i=1}^{n} 2\left[\dot{\mathbf{p}}_i{}^T\left(k_1^i M_{\mathbf{V}_i'}\left(\mathbf{C}_{\mathbf{V}_i'} - \mathbf{p}_i\right) + \ddot{\mathbf{p}}_i\right)\right] \tag{6-21}$$

Finally, by substituting the model of each mobile agent (6-17) into (6-21) , and using control input (6-18), the time derivative of Lyapunov function can be optioned as follows

$$\dot{\vartheta} = 2\sum_{i=1}^{n}\left(-k_2^i\,\dot{\mathbf{p}}_i^T\dot{\mathbf{p}}_i\right) \tag{6-22}$$

which is clearly non-positive. Let S be the set of all points in Q where $\dot{\vartheta} = 0$. Due to the convexity of the region Q, one can conclude that each of the limited-range Voronoi centroids $\mathbf{C}_{\mathbf{V}_i'}$ lies in the interior of the i^{th} limited-range Voronoi region and so in the interior of the region Q. So the mobile agents move toward the interior of the region Q and never leave it. Therefore, Q is a positive invariant set for the trajectories of the closed-loop system. Since this set is closed and bounded, one can make use of LaSalle's invariance principle to infer that the planar positions of all agents converge to the largest invariant subset of the S. Suppose a trajectory belongs to the set S. By considering the agents' model (6-17) and the control law (6-18), we have

$$\dot{\mathbf{p}}_i = \mathbf{0} \quad \Rightarrow \quad \ddot{\mathbf{p}}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{u}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{p}_i = \mathbf{C}_{\mathbf{V}_i'}, \forall i$$

Then we can conclude that $\mathbf{p}_i = \mathbf{C}_{\mathbf{V}_i'}, \forall i$, is the largest invariant set. Therefore, under control law (6-18), the closed-loop system is asymptotically stable and the planar positions of agents converge to the centroid of their limited-range Voronoi region.

□

We define ϕ^* as the maximum accepted uncertainty about each point in the environment. It is assumed that when the uncertainty is less than ϕ^* it is possible to decide about the status of the point with high level of accuracy. Therefore, when the uncertainty of all points becomes less than this threshold, the mission is complete. The amount of error about the status of each point can be defined as

$$e(\mathbf{q}) = \max(0, \phi(\boldsymbol{q}) - \phi^*) \tag{6-23}$$

Since each agent is responsible for searching the areas in its Voronoi region, the total error associated with a mobile agents is

$$e_i = \int_{\mathbf{V}_i} e(\mathbf{q})d\mathbf{q} \tag{6-24}$$

This total error comprises two parts; error from within limited-range Voronoi and error from outside of limited-range Voronoi as follows

$$e_i = e_i' + e_i'' \tag{6-25}$$

where

$$e_i' = \int_{\mathbf{V}_i'} e(\mathbf{q}) d\mathbf{q} \tag{6-26}$$

and

$$e_i'' = \int_{\mathbf{V}_i''=\mathbf{V}_i-\mathbf{V}_i'} e(\mathbf{q}) d\mathbf{q} \tag{6-27}$$

Under the control law (6-18), a mobile agent is in continuous motion until it converges to the centroid of its limited-range Voronoi region, i.e. $\dot{\mathbf{p}}_i = \mathbf{0}$ and $\mathbf{C}_{\mathbf{V}_i'} = \mathbf{p}_i$. Next lemma shows that the error inside the limited-range Voronoi region (e_i') becomes zero in a limited time.

Lemma 6-1: The uncertainty of all points inside the limited-range Voronoi region of a sensor goes below any desired threshold level in finite time.

***Proof*:** Uncertainty of a point inside the limited-range Voronoi at n^{th} time step is

$$\begin{aligned} \phi_n(\mathbf{q}) &= \phi_{n-1}(\mathbf{q}) \left(1 - \mu(\mathbf{q}, \mathbf{p}_i)\right) \\ &= \phi_0(\mathbf{q}) \left(1 - \mu(\mathbf{q}, \mathbf{p}_i)\right)^n \end{aligned}$$

where $\phi_0(\mathbf{q})$ is the initial uncertainty. Since $(1 - \mu(\mathbf{q}, \mathbf{p}_i)) < 1$, there is $0 \leq n^*$ that for any $n^* \leq n$,

$$\left(1 - \mu(\mathbf{q}, \mathbf{p}_i)\right)^n \leq \frac{\phi^*}{\phi_0(\mathbf{q})}$$

□

When a mobile agent converges to the centroid of its limited-range Voronoi, i.e. $\dot{\mathbf{p}}_i = \mathbf{0}$ and $\mathbf{C}_{\mathbf{V}_i'} = \mathbf{p}_i$, above lemma implies that e_i' eventually goes to zero. However, e_i is not necessary

zero, since e_i'' can be non-zero. Whenever the mobile agent is at the centroid of its limited-range Voronoi and the error from within limited-range Voronoi is zero, but the total error is non-zero, i.e. $e_i' = 0$ and $e_i \neq 0$, another control law must be utilized to perturb the system from its local minimum. Once the system is away from the local minimum, the controller is switched back to the nominal control.

A simple control law to perturb the system from the local minimum is a law that forces the mobile agent to move toward the point with the highest value of uncertainty in the Voronoi region. We define

$$\mathbf{q_i}^* = \arg\max_{\mathbf{q}\in\mathbf{V_i}''} \phi(\mathbf{q}) \tag{6-28}$$

It should be noted that when $e_i' = 0$ and $e_i \neq 0$, the point with the highest value of uncertainty in the Voronoi region, $\mathbf{q_i}^*$, always belongs to $\mathbf{V_i}''$. We assume that $\mathbf{q_i}^*$ is unique or there is an algorithm to choose a unique $\mathbf{q_i}^*$ when (6-28) has multiple solutions.

Then the control law that drives the agent to the $\mathbf{q_i}^*$ is as follows

$$\mathbf{u}_i' = -k_3^i(\mathbf{p}_i - \mathbf{q_i}^*) - k_4^i\dot{\mathbf{p}}_i \tag{6-29}$$

where k_3^i and k_4^i are positive controller gains.

***Theorem 6-2*:** Consider an agent whose dynamic model is described in (6-17). The control law (6-29) will drive the agent towards its associated $\mathbf{q_i}^*$.

***Proof*:** By using the following Lyapunov function

$$\vartheta = k_3^i(\mathbf{q_i}^* - \mathbf{p}_i)^T(\mathbf{q_i}^* - \mathbf{p}_i) + \dot{\mathbf{p}}_i^T\dot{\mathbf{p}}_i \tag{6-30}$$

and using LaSalle's invariance principle, one can conclude that the closed loop system is asymptotically stable and the position of the agent converge to its associated $\mathbf{q_i}^*$.

□

We define the switching condition *C* as follows

- *Condition C*: $\mathbf{u}_i = \mathbf{0},\ e_i' = 0$ and $e_i \neq 0$

Therefore, when condition *C* holds, the control law switches to the control law (6-29) that drives the mobile agent to the point with the highest uncertainty in its Voronoi-region. When an agent converges to the centroid of its limited-range Voronoi it does not switch its control law immediately. In fact, it uses the nominal control law, which is zero at that moment, to stay at the centroid of the limited-range Voronoi until the total uncertainty inside the limited-range Voronoi region goes below the threshold, i.e. $e_i' = 0$. It is worth to mention that the condition *C* does not occur very often because after each time step both uncertainty distribution in the environment and the position of mobile agents change which changes the limited-range Voronois and their centroids. The following theorem summarizes what we discussed so far.

***Theorem 6-3*:** Consider a group of *n* agents whose dynamic models are described in (6-17). Under the uncertainty model (6-4), the control law

$$\mathbf{u}_i^* = \begin{cases} \mathbf{u}_i & \text{if } C \text{ does not hold} \\ \mathbf{u}_i' & \text{if } C \text{ holds} \end{cases} \tag{6-31}$$

can drive the total error to zero, where $\mathbf{u}_i$ and $\mathbf{u}_i'$ can be calculated by using (6-18) and (6-29) respectively.

6.3 Collision Avoidance

As we mentioned earlier, it is assumed that at any given point, only the measurement from the agent which can reduce the uncertainty by largest value is incorporated. Thus, in order to use the maximum capability of all mobile agents, it is beneficial if the sensing area of mobile agents do not interfere with each other's. Therefore, the mobile agents are expected to maintain the

distance greater than $2r_\gamma$ from each other, if it is possible. In addition, in order to avoid collision between agents, there is a minimum distance between mobile agents that must always be respected. If r_o is defined as the radius of the disc centered at the agent's location and circumscribes the mobile agent, the minimum acceptable distance between two agents is $2r_o$. We consider the following function

$$U_{ij} = \min(0, \frac{\|\mathbf{p}_i - \mathbf{p}_j\| - 2r_\gamma}{\|\mathbf{p}_i - \mathbf{p}_j\| - 2r_o}) \tag{6-32}$$

It is assumed that in general $r_\gamma > r_o > 0$. The value of this function is decreasing and negative when $2r_o < \left\|\mathbf{p}_i - \mathbf{p}_j\right\| < 2r_\gamma$ and is zero otherwise.

The following control law

$$\mathbf{u}_i = k_5^i \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i} - k_6^i \dot{\mathbf{p}}_i \tag{6-33}$$

ensures that as long as all agents start from initial conditions that $2r_o < \left\|\mathbf{p}_i(0) - \mathbf{p}_j(\mathbf{0})\right\|$, the mobile agents never collide and their final position is such that the sensing area of mobile sensors do no interfere. Parameters k_5^i and k_6^i are positive controller gains.

Assumption 1: The initial position of mobile agents satisfies the collision avoidance condition, i.e. $2r_o < \left\|\mathbf{p}_i(0) - \mathbf{p}_j(\mathbf{0})\right\|$.

***Theorem 6-4*:** Consider a group of *n* agents whose dynamic models are described in (6-17). If the assumption 1 is true, the control law (6-33) guarantees collision avoidance and ensures that the planner position of system converge to a configuration that the sensing area of mobile agents do not interfere.

***Proof*:** By using the following Lyapunov function

$$\vartheta = -\sum_{i=1}^{n} \sum_{j=1,\neq i}^{n} k_5^i U_{ij} + \sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i$$

Taking the time derivative of ϑ along the trajectories of systems gives

$$\dot{\vartheta} = -\sum_{i=1}^{n}\sum_{j=1,\neq i}^{n} k_5^i \dot{\mathbf{p}}_i^T \frac{\partial U_{ij}}{\partial \mathbf{p}_i} + \sum_{i=1}^{n}\sum_{j=1,\neq i}^{n} k_5^i \dot{\mathbf{p}}_j^T \frac{\partial U_{ij}}{\partial \mathbf{p}_j} + 2\sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \ddot{\mathbf{p}}_i$$

$$= -\sum_{i=1}^{n}\sum_{j=1,\neq i}^{n} k_5^i \dot{\mathbf{p}}_i^T \frac{\partial U_{ij}}{\partial \mathbf{p}_i} + \sum_{i=1}^{n}\sum_{j=1,\neq i}^{n} k_5^i \dot{\mathbf{p}}_j^T \frac{\partial U_{ji}}{\partial \mathbf{p}_j} + 2\sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \ddot{\mathbf{p}}_i$$

$$= -\sum_{i=1}^{n}\sum_{j=1,\neq i}^{n} k_5^i \dot{\mathbf{p}}_i^T \frac{\partial U_{ij}}{\partial \mathbf{p}_i} + \sum_{i=1}^{n}\sum_{j=1,\neq i}^{n} k_5^i \dot{\mathbf{p}}_i^T \frac{\partial U_{ij}}{\partial \mathbf{p}_i} + 2\sum_{i=1}^{n} \dot{\mathbf{p}}_i^T \ddot{\mathbf{p}}_i$$

where we used $U_{ij} = U_{ji}$ and $\frac{\partial U_{ji}}{\partial \mathbf{p_j}} = \frac{\partial U_{ij}}{\partial \mathbf{p_i}}$ in deriving the above equation. Finally, by substituting the model of each mobile agent (6-17) into above equation, and using control input (6-33), the time derivative of Lyapunov function can be obtained as follows:

$$\dot{\vartheta} = -2\sum_{i=1}^{n}\left(k_6^i\, \dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i\right)$$

which is clearly non-positive with equality holding if and only if $\dot{\mathbf{p}}_i = \mathbf{0}$. By considering the agent's model (6-17) and the control law (6-33), we have

$$\dot{\mathbf{p}}_i = \mathbf{0} \quad \Rightarrow \quad \ddot{\mathbf{p}}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{u}_i = \mathbf{0} \quad \Rightarrow \quad \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i} = 0\,, \forall i$$

The latter condition is satisfied if and only if $\frac{\partial U_{ij}}{\partial \mathbf{p_i}} = 0, \forall i$, which means the relative distance of any two agents cannot be between r_o and r_γ. On the other hand, since the initial relative distance was greater than $2r_o$, if two agents are going to collide, i.e. $\|\mathbf{p_i} - \mathbf{p_j}\| \to 2{r_o}^+$ it implies $\vartheta \to \infty$ which is not possible because that initial value of ϑ is finite and non-increasing. Therefore, this guarantees the collision avoidance.

□

It is worth to mention that if the sensing radius of the agents is large in relation to the dimension of environment, the above control law may force some of mobile agents outside of the domain. In practice this is not an issue, since usually a small group of agents are responsible to search a relatively large environment.

By using the control law (6-18), the agents incorporate a greedy strategy to locally optimize the mission objective. They assume that the current partitioning of the domain is optimal and try to choose the best actions that maximize the objective function. On the other hand, using the control law (6-33) locally optimizes the partitioning of the domain and guarantees collision avoidance. The control law can be a combination of both (6-18) and (6-33) as follows

$$\bar{\mathbf{u}}_i = -k_c^i M_{\mathbf{V}_i'}\left(\mathbf{p}_i - \mathbf{C}_{\mathbf{V}_i'}\right) - k_p^i \dot{\mathbf{p}}_i + k_u^i \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i} \tag{6-34}$$

where k_c^i, k_p^i, and k_u^i are positive gains. It is straightforward to show that by using this control law the mobile agents converge to the position that $\mathbf{p}_i = \mathbf{C}_{\mathbf{V}_i'} + \frac{k_u^i}{k_c^i M_{\mathbf{V}_i'}} \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$. The gains k_c^i and k_u^i can be chosen to balance between different objectives.

Based on Lemma 6-1, after the mobile agent converges to any point inside its limited-range Voronoi, the error from within limited-range Voronoi goes to zero in limited time. However, as we saw earlier, it is possible that the error from within limited-range Voronoi is zero, but the total error is non-zero, i.e. $e_i{}' = 0$ and $e_i \neq 0$. In that case another control law must be utilized to perturb the system from its local minimum. Once the system is away from the local minimum, the controller is switched back to the nominal control.

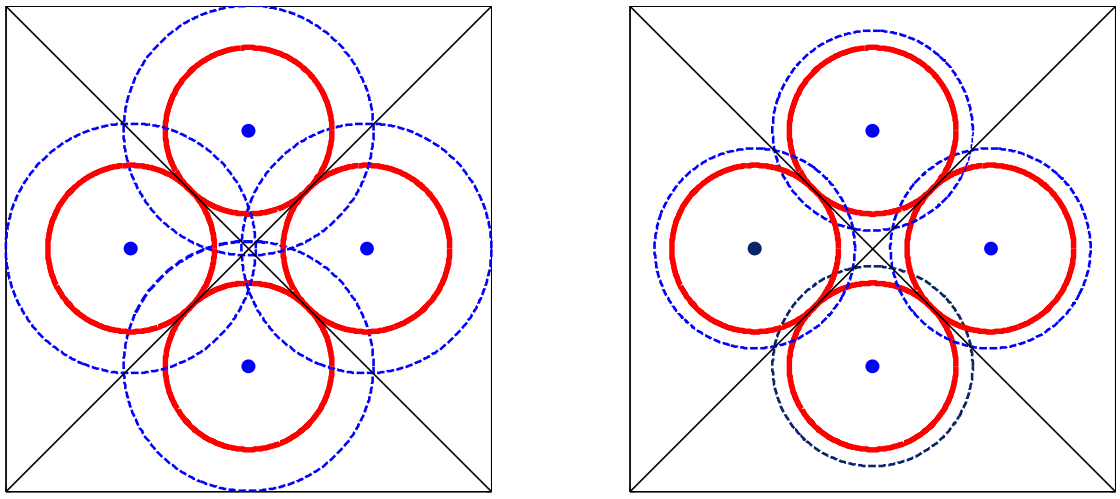
A weighted combination of the control laws (6-29) and (6-33) can be utilized in this case to force the mobile agent to move toward the point with highest uncertainty in the Voronoi region while avoiding the collision with other agents

$$\bar{\mathbf{u}}_i{}' = -k_{c'}^i(\mathbf{p}_i - \mathbf{q}_i^*) - k_{p'}^i \dot{\mathbf{p}}_i + k_{u'}^i \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i} \tag{6-35}$$

where $k_{c'}^i$, $k_{p'}^i$, and $k_{u'}^i$ are positive gains. Similarly, this control law moves the mobile agents toward the position that $\mathbf{p}_i = \mathbf{q}_i^* + \frac{k_{u'}^i}{k_{c'}^i} \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$. Although the above control law guarantees

the collision avoidance, it cannot guarantee that the total error inside the Voronoi region goes to zero. In fact, in a symmetric situation like Figure 6-1 where multiple agents try to reach to almost the same point, the repulsive force of the others prevent them from doing that. In that case, a part of the Voronoi region may left uncovered.

One way to mitigate this problem is to replace r_γ with a smaller value $r_{\gamma'}$ ($r_o < r_{\gamma'} < r_\gamma$) in constructing U_{ij} for the (6-35) control law. This way, the agents can get closer to each other which decreases the chance of having unexplored area. However, this means that the sensing area of mobile agents may interfere in any time which is not efficient. More importantly, from practical point of view this may lead to a situation that two agents get dangerously close to each other such that the control input required to prevent collision is too high and the agent are not able to produce such a large input.



(a) (b)

Figure 6-1. A symmetric situation with 4 agents. Red solid circles are safety regions of agents and blue dashed circles are sensory domains of agents. The point with the highest value of uncertainty in the Voronoi region of all agents, i.e. $\mathbf{q}_i^*$, is located near the node at the intersection of all Voronois. In (**a**) the sensory domain of the agents is large enough to cover the whole area around the intersection, while in (**b**) some part of the domain remains uncovered.

The other way to solve this problem is to using a conflict resolution algorithm that prioritizes the agents and resolves these sort of conflicts. This may need more communication between agents.

We define the following switching conditions

- *Condition* $\bar{C}$: $\bar{\mathbf{u}}_i = \mathbf{0},\ e_i' = 0$ and $e_i \neq 0$
- *Condition* $\bar{C}'$: $\bar{\mathbf{u}}_i' = \mathbf{0},\ e_i' = 0$ and $e_i \neq 0$

A simple example of such conflict resolution algorithm is the one that when condition $\bar{C}'$ holds, the agent with higher priority chooses its $\mathbf{q_i}^*$ using (6-28), while the other agents keep choosing less optimal value for their corresponding $\mathbf{q_i}^*$ in descending order of optimality until the condition $\bar{C}'$ is violated.

One may use another conflict resolution algorithm to deal with such situation without need of extra communication between agents. In this algorithm all agents have the same priority. When the condition $\bar{C}'$ holds for any agent, that agent randomly chooses a point inside $\mathbf{V_i}''$ as $\mathbf{q_i}^*$. This randomness can eventually break the symmetry and violate the condition $\bar{C}'$ for all agents. This algorithm is clearly less optimal than the former algorithm, but it needs less computation and less communication between agents.

Therefore, the control law

$$\bar{\mathbf{u}}_i^{\,*} = \begin{cases} \bar{\mathbf{u}}_i & \text{if } \bar{\mathbf{C}} \text{ does not hold} \\ \bar{\mathbf{u}}_i' & \text{if } \bar{\mathbf{C}} \text{ holds} \end{cases} \tag{6-36}$$

guarantees the collision avoidance, but it cannot necessarily guarantee that the total error of the entire domain goes to zero in highly symmetric domains. However, using a conflict resolution algorithm when the condition $\bar{\mathbf{C}}'$ holds, can break the symmetry and reduce the uncertainty to any desired threshold. It should be noted that when $\bar{\mathbf{C}}'$ holds, the control input is still equal to $\bar{\mathbf{u}}_i'$, but

the value of $\mathbf{q_i}^*$ is chosen based on the conflict resolution algorithm. As we will see in the simulation and result section, in practice, the total performance of the proposed control strategy is very high and close to the performance of the cases without collision avoidance requirement (mass point agents).

The control strategy is summarized below:

- If $\overline{\mathbf{C}}$ does not hold: $\overline{\mathbf{u}}_i{}^*=\overline{\mathbf{u}}_i=-k_c^i M_{\mathbf{V}_i'}\left(\mathbf{p}_i - \mathbf{C}_{\mathbf{V}_i'}\right) - k_p^i \dot{\mathbf{p}}_i + k_u^i \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$
- If $\overline{\mathbf{C}}$ holds but $\overline{\mathbf{C}}'$does not hold: $\overline{\mathbf{u}}_i{}^*=\overline{\mathbf{u}}_i{}' = -k_{c'}^i(\mathbf{p}_i - \mathbf{q_i}^*) - k_{p'}^i \dot{\mathbf{p}}_i + k_{u'}^i \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$

  where $\mathbf{q_i}^*$ is calculated using $\mathbf{q_i}^* = \arg\max_{\mathbf{q}\in\mathbf{V}_i''} \phi(\mathbf{q})$ (6-28)

- If $\overline{\mathbf{C}}$ and $\overline{\mathbf{C}}'$ hold: $\overline{\mathbf{u}}_i{}^*=\overline{\mathbf{u}}_i{}' = -k_{c'}^i(\mathbf{p}_i - \mathbf{q_i}^*) - k_{p'}^i \dot{\mathbf{p}}_i + k_{u'}^i \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$

  where $\mathbf{q_i}^*$ is chosen using conflict resolution algorithm (randomly)

6.4 Simulation Results

In this section, we provide different simulation results to show the performance of the proposed search method. All simulation have been done in Matlab® R2012a environment on a PC with 2.4 GHz CPU. The dynamics are implemented in discrete time with a sampling time of 0.1 s. A discrete Voronoi partitioning is used where a cell belongs to a Voronoi region if its center is closer to the generating point of that region than to any other generating point. The Value of $\mathbf{C}_V$ is approximated by replacing the integral with a summation. Since the uncertainty value of all points inside a cell is equal, the approximation error is negligible.

The environment is a 10 m × 10 m square that discretized into 10000 cells. The radius of safety region and sensory domain of the agents is r_o =50 cm and r_γ =100 cm, respectively. The

peak value of γ at the observation point is γ_0 =0.9. In all simulations, the sampling rate of the sensors is 10 Hz. In other words, each sensor scans the environment every 100 ms. The maximum accepted uncertainty about each point in the environment is $\phi^* = 0.1$. The value of control gains are $k_c^i = k_{c'}^i$ =0.5, $k_p^i = k_{p'}^i$=2.5, and $k_u^i = k_{u'}^i$=0.5 for all agents.

In the first scenario, it is assumed that there is no *a priori* information about the environment. Therefore, at the beginning, the uncertainty density $\phi(\mathbf{q})$ is uniformly distributed all over the environment. To make comparison between different simulations possible, the total value of uncertainty in the domain is set to 10000 at the beginning of all simulations. Thus, for this scenario the uncertainty density is initialized with $\phi(\mathbf{q}) = 1$ for all cells $\mathbf{q}$. There are three agents in the environment that start their mission from a randomly chosen position inside the environment. However, the initial position of the agents satisfies the collision avoidance criteria at the beginning of the search mission.

The path and position of agents at different times are shown in Figure 6-2. The color intensity is proportional to the value of uncertainty. It can be seen that the agents can eventually explore the entire domain and reduce the value of uncertainty below any desired threshold. To evaluate the average performance of the proposed search method, we performed a Monte Carlo simulation for 50 times and the results are reported. The simulations are also repeated for the case with five agents. Figure 6-3 shows the average value of error for the mission with three and five agents.

As one may expect, when the number of agent increases, the performance of mission increases too, which means the total error decreases more rapidly. The average value of control input and velocity of all agents are shown in Figure 6-4 to 6-7. From the practical point of view, these values are reasonable for a wide range of commercial wheeled mobile robots and quad rotor helicopters.

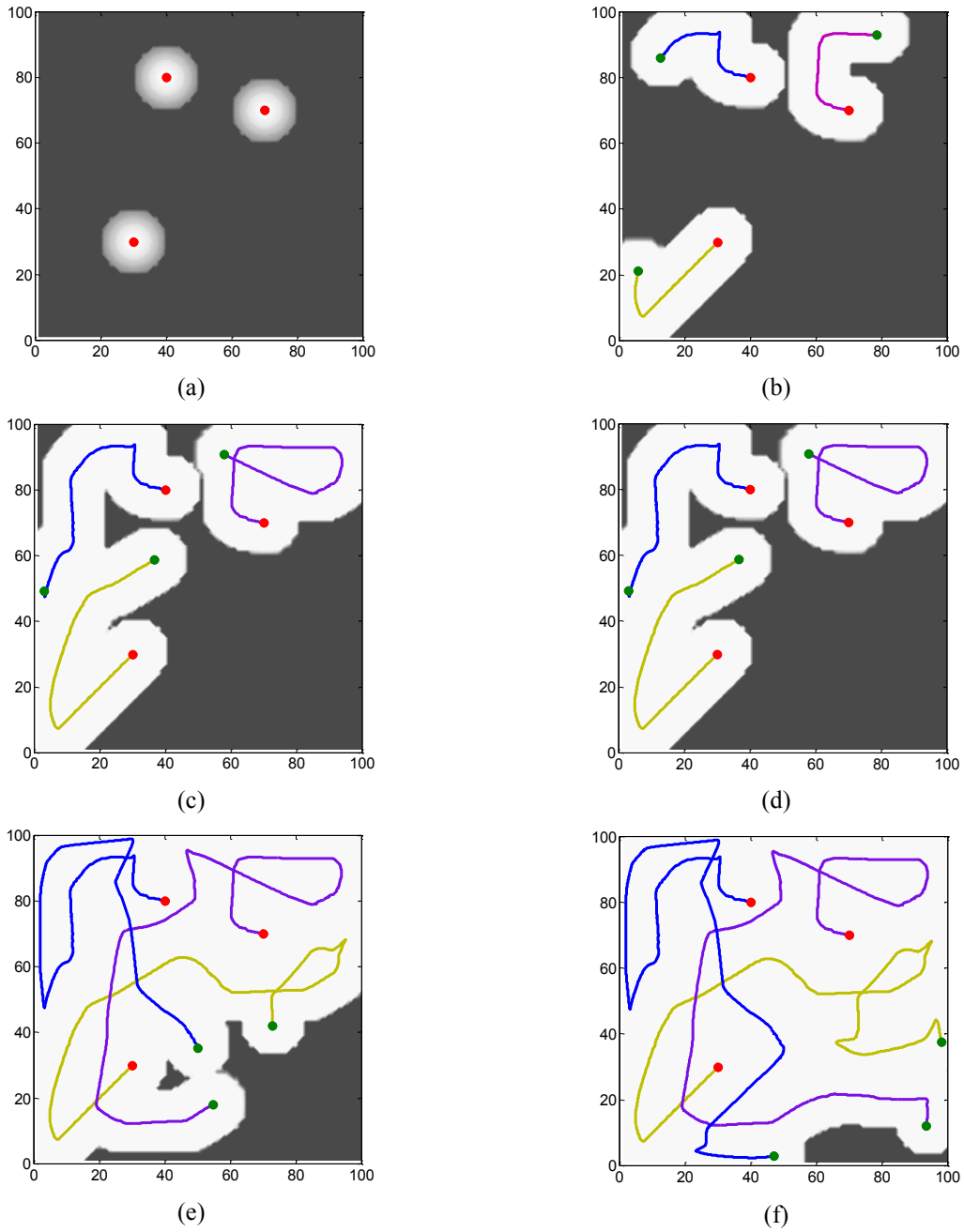


Figure 6-2. The path and the position of agents at different times in an environment with uniformly distributed uncertainty. The color intensity is proportional to the value of uncertainty. Red markers denote the initial position of the agents and green markers denote the current positions.

a) t=0 s, b) t=30 s, c) t=60 s, d) t=90 s, e) t=120 s, f) t=150 s

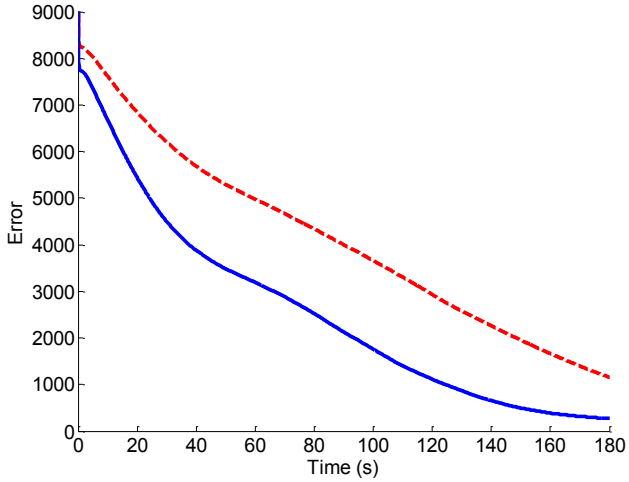


Figure 6-3. The average error for 50 random simulations.
Red dashed: mission with 3 agents.
Blue solid: mission with 5 agents.

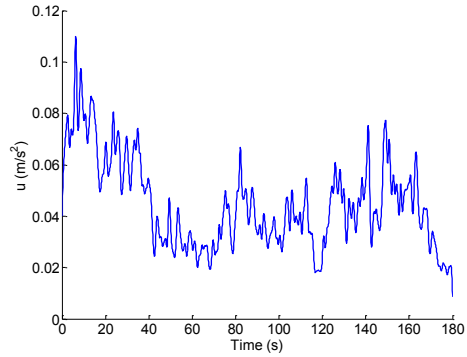


Figure 6-4. The average value of control input for 50 random simulations with 3 agents.

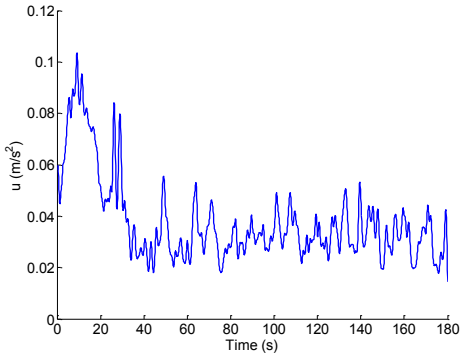


Figure 6-5. The average value of control input for 50 random simulations with 5 agents.

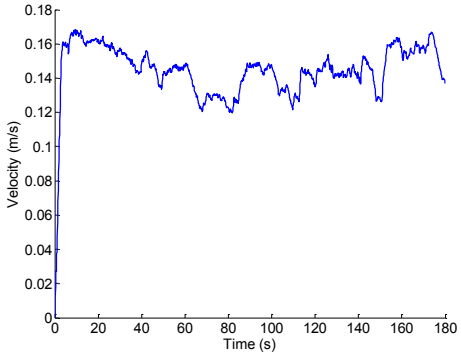


Figure 6-6. The average value of velocity for 50 random simulations with 3 agents.

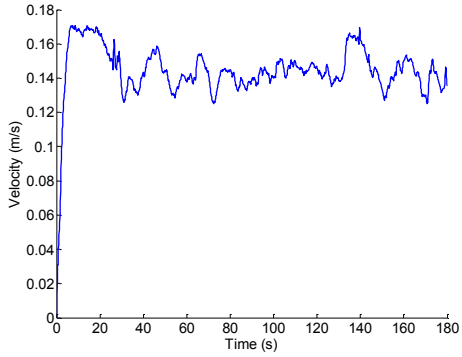


Figure 6-7. The average value of velocity for 50 random simulations with 5 agents.

In the first scenario, since the initial uncertainty is uniformly distributed over the entire domain, the advantage of the proposed control algorithm cannot be seen clearly. In the second scenario, the initial uncertainty distribution is not uniform. It is assumed that there are three random uncertainty centers in the environment q_c^j and the uncertainty is normally distributed around them.The total uncertainty of each cell is the summation of the uncertainty from all three different sources

$$\phi(\mathbf{q}) = \sum_{j=1}^{3} \frac{1}{\sigma\sqrt{2\pi}} \left(e^{-\frac{\left(q-q_c^j\right)^2}{2\sigma^2}} \right)$$

where $\sigma = 1.5\ m$.

There are three agents that start their mission from the bottom-left corner of the environment while respecting the safety region of each other. The path and position of agents at different times are shown in Figure 6-8. It can be seen that using the control law (6-36) causes the agent to concentrate their efforts around the area with higher value of uncertainty. We performed Monte Carlo simulation for 50 times and the results are reported. The simulations are also repeated for the case with five agents. Figure 6-9 shows the average value of error for the mission with three and five agents. As we expect, when the number of agent increases, the performance of mission increases too. The average value of control input and velocity of all agents are shown in Figure 6-10 to 6-13. It can be seen that these values are reasonable from the practical point of view for a wide range of applications. In fact, in all simulation in this chapter, we set the maximum value of control input at 0.5 m/s^2, i.e. $\| \overline{\mathbf{u}}_i^* \| \leq 0.5$. However, in all simulations, the control input produced by (6-36) rarely went over that limit.

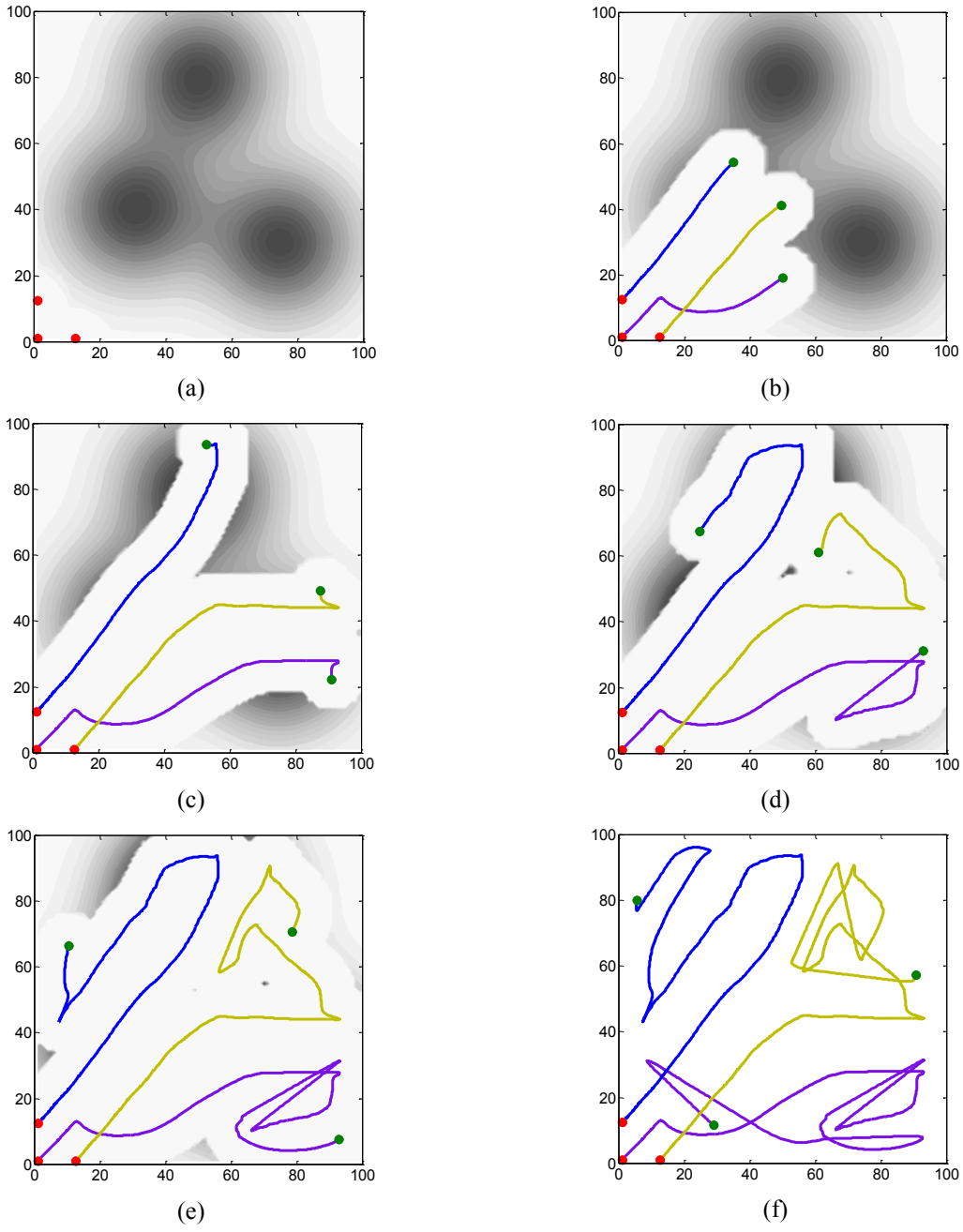


Figure 6-8. The path and the position of agents at different times in an environment with non-uniformly distributed uncertainty. The color intensity is proportional to the value of uncertainty. Red markers denote the initial position of the agents and green markers denote the current positions.

a) t=0 s, b) t=20 s, c) t=40 s, d) t=60 s, e) t=80 s, f) t=100 s

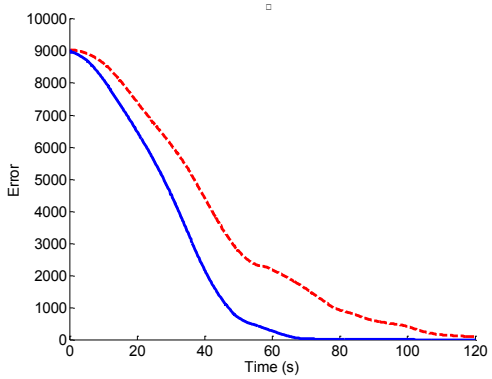


Figure 6-9. The average error for 50 random simulations.
Red dashed: mission with 3 agents.
Blue solid: mission with 5 agents.

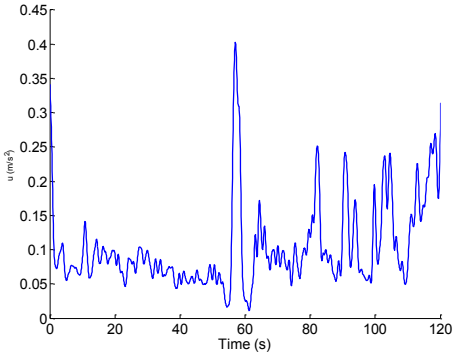


Figure 6-10. The average value of control input for 50 random simulations with 3 agents.

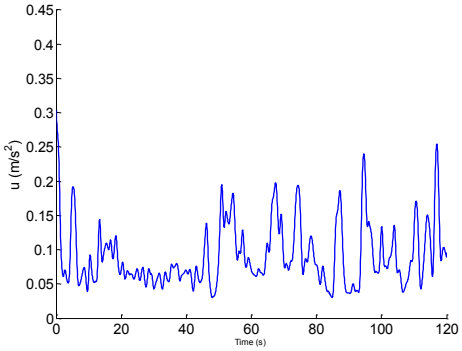


Figure 6-11. The average value of control input for 50 random simulations with 5 agents.

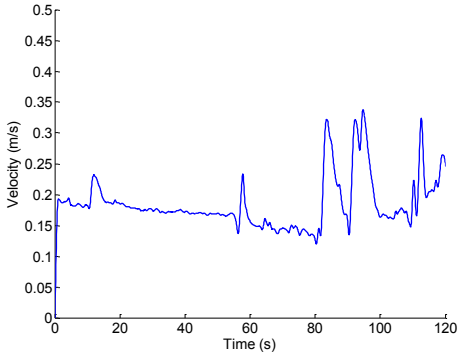


Figure 6-12. The average value of velocity for 50 random simulations with 3 agents.

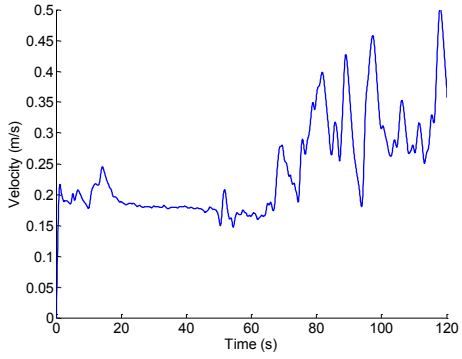


Figure 6-13. The average value of velocity for 50 random simulations with 5 agents.

When there is a limit on the control input, it may not be able to ensure collision avoidance. In order to decrease the chance of collision in such situations, we need to increase the value of k_u^i (or $k_{u'}^i$) with respect to k_c^i (or $k_{c'}^i$). This can reduce the performance of mission by pushing the equilibrium point further away from the optimal point $\mathbf{C}_{\mathbf{V}_i'}$(or $\mathbf{q}_i^*$) to the new point $\mathbf{C}_{\mathbf{V}_i'} + \frac{k_u^i}{k_c^i M_{\mathbf{V}_i'}} \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$ (or $\mathbf{q}_i^* + \frac{k_{u'}^i}{k_{c'}^i} \sum_{j=1,\neq i}^{n} \frac{\partial U_{ij}}{\partial \mathbf{p}_i}$). In addition, we may also need to increase the value of k_p^i (or $k_{p'}^i$) with respect to k_c^i (or $k_{c'}^i$) to limit the velocity of agents which can slow down the convergence of system.

It should be noted that although having more agents in the mission naturally increases the performance of mission, when the control input is limited, having too many agents in a small environment also increases the chance of collision. To prevent collision, we need to tune the gains which has negative effects on the performance. Therefore, when there are a few agents in a relatively large domain, adding a new agent considerably increases the overall performance, but when the number of agent with respect to dimension of the environment is high, adding a new agent is less desirable since it may have very little effect on the overall performance of the mission.

6.5 Conclusion

The problem of multi agent search in uncertain environments is investigated. The lack of information about the environment is modeled with an uncertainty density. A team of mobile sensors with limited sensory domain searches the environment until the value of uncertainty of all points in the environment goes below a pre-set value. It is shown that the optimal decision of each mobile sensor at each time step is to move towards the centroid of its limited range Voronoi region. A distributed control strategy is proposed that guarantees the asymptotic convergence of each mobile sensor to the centroid of its limited range Voronoi region. A collision avoidance component

is then added to the controller which guarantees the collision avoidance, but it cannot necessarily guarantee that the total error of the entire domain goes to zero in highly symmetric domains. However, using a conflict resolution algorithm can break the symmetry and reduce the uncertainty to any desired threshold. Simulations results illustrates the effectiveness of the proposed approach

CHAPTER 7

Conclusions and Future Work

This chapter summarizes the dissertation contributions and presents future directions for the research.

7.1 Conclusions

The main objective of this dissertation is to investigate distributed architectures for cooperative multi agent search and coverage problem. The main thesis contributions are as follows:

- Different models for uncertain environments are presented in chapter 2. The probability map updating rule is developed for different types of sensors (single-cell footprint or multiple-cells footprint) and different types of targets (distinguishable or indistinguishable). Environments with unknown number of targets and environments with known number of targets are both investigated. The idea of relative probability is extended for the latter case to decrease the computational burden.
- In chapter 3, a decentralized approach is used for the search mission where each mobile agent chooses its optimal action individually. To make cooperation between agents possible, two approximation methods are proposed to modify the objective function of agents and take into the account the action of other agents. This approach is then extended for the case with known communication delay between mobile agents.

- Search and coverage problem is introduced and formulated in chapter 4. Inspired by real applications, a new distribution density model is introduced which is a function of position of some unknown targets in the environment. The problem is formulated such that the information about the positions of the targets is updated by some search vehicles agents. The cooperative search method developed in chapter 2 and a Centroidal Voronoi Configuration method for coverage are used to solve the problem.
- In chapter 5, the limited turn rate Voronoi diagram is introduced and the search and coverage problem is formulated as a multi-objective optimization problem with different constraints including minimum fuel consumption, refuelling, obstacle avoidance, and collision avoidance. Despite the method used in the previous chapter, there is only one type of agents which perform both search and coverage tasks.
- In chapter 6, a Voronoi-based search strategy for a team of mobile agents with limited range sensors is presented which combines mid-level and low-level controllers. The collision avoidance between agents is guaranteed. The control law is designed to balance between the myopic objective of maximizing uncertainty reduction in the next step and the long term objective of distributing the agent in the environment with minimum overlap in their sensory domain. The dynamic model of agents is also a double integral which can express the equation of motion of a broad class of vehicles.

7.2 Future Work

The following problems are suggested for future research:

- A more general model for the uncertain environments can be investigated where the probability of existence of a target in the environment and the distribution of that probability are known *a priori.*

- A 3D model for the sensors can be developed which captures the effect of altitude of sensor on its performance.
- When the search agents are far from the region of interest, using the rolling horizon limited look-ahead strategy is not efficient. In that situation, two different approaches can be considered: using an approximate Dynamic Programming method to evaluate the value function at the end of look-ahead horizon; or using a switching mechanism that changes the decision making method to a reactive one to guides the agents to vicinity of the targets.
- We assumed that all agents always maintain the same cognitive map of the environment. Different factors such as communication delay, limited communication range, or agent failure may cause the agents to have different cognitive maps. Effect of having different cognitive maps and methods of combining these different maps to make a global map should be investigated.
- High level decision making issues like task assignment can also be considered.
    - In the problem of search and coverage in chapter 4, all agents can be identical but with different equipment. The task assignment unit must decide about the distribution of tasks (and therefore equipment) between available agents.
    - In chapter 3, 4, and 6, the task assignment unit can change the number of active agents, based on the new information from the agents or perhaps other sources. It can also decide about when to remove or replace a faulty agent based on the different criteria such as the importance of the task, severity of damage, chance of crash, and the price of agent.

Bibliography

[1] Y. Wang and I. Hussein, "Bayesian-based decision making for object search and characterization," in *American Control Conference*, St. Louis, MO, 2009, pp.1964 -1969.

[2] M. L. Baum and K. M. Passino, "A search-theoretic approach to cooperative control for uninhabited air vehicles," In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, CA, 2002, pp.1-6.

[3] Bellingham, J. S., M. Tillerson, M. Alighanbari and J. P. How, "Cooperative Path Planning for Multiple UAVs in Dynamic and Uncertain Environments," in *IEEE Conference on Decision and Control*, Las Vegas, NV, 2002, pp. 2816–2822.

[4] Jin, Y., A. Minai, and M. Polycarpou, "Cooperative Real-Time Search and Task Allocation in UAV Teams," in *IEEE Conference on Decision and Control*, Maui, HI, 2003, pp. 7-12.

[5] D. J. Pack, P. DeLima, G. J. Toussaint, and G. York, "Cooperative control of UAVs for localization of intermittently emitting mobile targets," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 4, pp. 959–970, Aug. 2009.

[6] N. Nigam, S. Bieniawski, I. Kroo, J. Vian, "Control of Multiple UAVs for Persistent Surveillance: Algorithm and Flight Test Results," *IEEE Transactions on Control Systems Technology*, vol.20, no.5, pp.1236-1251, Sep. 2012.

[7] B. O. Koopman, "The Theory of Search I: Kinematic Bases," *Operations Research*, vol. 4, no. 3, pp. 324-346, June 1956.

[8] B. O. Koopman, "The Theory of Search II: Target Detection," *Operations Research*, vol. 4, no.5, pp. 503-531, 1956.

[9] B. O. Koopman, "The Theory of Search III: The Optimum Distribution of Searching Effort," *Operations Research*, vol. 5, no. 5, pp. 613-626, Oct. 1957.

[10] L. D. Stone, *Theory of Optimal Search*, New York, NY: Academic Press, 1975

[11] B. O. Koopman, *Search and Screening: General principles with Historical Application*, New York, NY: Pergamon, 1980.

[12] J. N. Eagle and J.R. Yee, "An optimal branch-and-bound procedure for the constrained path moving target search problem", *Operations Research*, vol. 38, no. 1, pp. 110-114, Feb. 1990

[13] R. Hohzaki and K. Iida, "An optimal search plan for a moving target when a search path is given", *Mathematica Japonica*, vol. 41, no.1, pp.175-184, Jan. 1995.

[14] R. Hohzaki and K. Iida, "Path constrained search problem with reward criterion", *Journal of the Operations Research Society of Japan*, vol. 38, no. 2, pp.254-264, June 1995.

[15] L. Wu, M. A. Garcia, D. Puig, and A. Sole, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, vol. 1, no. 1, pp. 37–44, Sep. 2007.

[16] A. Solanas and M. A. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendal, Japan, 2004, pp. 717–721.

[17] M. Flint, M. Polycarpou, and E. Fernandez, "Stochastic Models of a Cooperative Autonomous UAV Search Problem," *Military Operations Research Journal*, vol. 8, no. 4, pp. 13–32, Sep. 2003.

[18] Y. Yang, M. M. Polycarpou, A. A. Minai, "Multi-UAV cooperative search using an opportunistic learning method," *ASME Journal of Dynamic Systems, Measurement and Control* 129, pp. 716-728, Sep. 2007.

[19] P. B. Sujit and D. Ghose, "Negotiation schemes for multi-agent cooperative search" *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 223 no. 6, pp. 791-813, June 2009.

[20] P. Sujit, D. Kingston and R. Beard, "Distributed coordination of multi-agent systems for coverage problem in presence of obstacles", in *IEEE Conference on Decision and Control*, Montreal, Canada, 2007, pp. 5252–5257.

[21] C. Luo, A. Espinosa, D. Pranantha, A. Gloria, "Multi-robot search and rescue team," in *IEEE International Symposium on Safety, Security and Rescue Robotics*, Kyoto, Japan, 2011, pp. 296–301.

[22] C. Phan and H. H. T. Liu, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *International Conference on System Simulation and Scientific Computing*, Beijing, China, 2008, pp. 494-498.

[23] M. Kumar, K. Cohen and B. H. Chaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 1, pp. 1–16, Jan. 2011.

[24] P. R. Chandler, "Cooperative Control of a Team of UAVs for Tactical Missions," in *AIAA Intelligent Systems Technical Conference*, Chicago, IL, 2004, pp. 1-9.

[25] S. G. Loizou and K. Kyriakopoulos. "Closed loop navigation for multiple holonomic vehicles". In *IEEE International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002, pp. 2861 – 2866.

[26] M. Howard, B. Hoff, and C. Lee, "Hierarchical Command and Control for Multi-Agent Teamwork," in *International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology,* Manchester, UK, 2000, pp. 1-13.

[27] S. M. Li, J. D. Boskovic, S. Seereeram, R. Prasanth, R. Amin, R. K. Mehra, and R.W. Mclain. T.W. Beard. "Autonomous hierarchical control of multiple unmanned combat air vehicles (UCAVs)," in *American Control Conference*, Anchorage, AK., 2002, pp. 274-279.

[28] P.R. Chandler and M Pachter, "Hierarchical Control for Autonomous Teams", in *AIAA Guidance, Navigation, and Control Conference*, Montreal, Canada, 2001, pp. 632-642.

[29] G. Vachtsevanos, L. Tang, and J. Reimann, "An Intelligent Approach to Coordinated Control of Multiple Unmanned Aerial Vehicles" in *American Helicopter Society Annual Forum*, Baltimore, MD, 2004, pp. 1-6.

[30] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic Pursuit-Evasion Games: Theory, Implementation and Experimental Evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662 – 669, Dec. 2002.

[31] P. Chandler, "Decentralized control for an autonomous team," in *AIAA Unmanned Unlimited systems, Technologies, and Operations*, San Diego, CA., 2003, pp.1-6.

[32] Y. Yang, A. Minai and M. Polycarpou, 2002, "Analysis of opportunistic method for cooperative search by mobile agents", in *IEEE Conference on Decision and Control*, Cincinnati, OH, 2002, pp. 576-577.

[33] P. B. Sujit and D. Ghose. "Multiple UAV Search using Agent Based Negotiation Scheme", in *American Control Conference*, Portland, OR, 2005, pp. 2995–3000.

[34] M. Flint, E. Fernandez-Gaucherand, and M. Polycarpou, "A probabilistic framework for passive cooperation among UAV's performing a search," in *International Symposium on Mathematical Theory of Networks and Systems*, Leuven, Belgium, 2004, pp.1-8.

[35] Yijia Zhao, Stephen D. Patek, and Peter A. Beling, "Decentralized Bayesian Search Using Approximate Dynamic Programming Methods" IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics, vol. 38, no. 4, pp. 970 – 975, Aug. 2008

[36] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Decentralized Cooperative Search by Networked UANs in an Uncertain Environment," in *American Control Conference*, Boston, MT, 2004, pp.5558-5563.

[37] K. Morris , B. Mullins , D. Pack and R. Baldwin, "Impact of limited communications on a cooperative search algorithm for multiple UAVs," in *IEEE International Conference Networking, Sensing and Control*, Ft. Lauderdale, FL, 2006, pp.572 -577.

[38] J.-P. LeCadre and G. Souris, "Searching tracks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 4, pp. 1149-1166, Oct. 2000.

[39] M. Jun, and R. D'Andrea, "Probability Map Building of Uncertain Dynamic Environments with Indistinguishable Obstacles," in *IEEE American Control Conference*, Denver, CO, 2003, pp. 3417 – 3422.

[40] P. Millet, D. Casbeer, T. Mercker, J. Bishop, "Multi-Agent Decentralized Search of a Probability Map with Communication Constraints," in *AIAA Guidance, Navigation, and Control Conference*, Toronto, Canada, 2010, pp. 1-12.

[41] L. F. Bertuccelli and J. P. How, "Robust UAV Search for Environments with Imprecise Probability Maps" in *IEEE Conference on Decision and Control, and the European Control Conference,* Seville, Spain, 2005, pp. 5680-5685.

[42] D. Pagac, E. M. Nebot and H. Durrant-Whyte, "An Evidential Approach to Map-Building for Autonomous Vehicles," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 623–629, Aug.1998.

[43] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated Decentralized Search for a Lost Target In a Bayesian World," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2003, pp. 48-53.

[44] M. M. Polycarpou, Y. Yang, and K. Passino, "A cooperative search framework for distributed agents," in *IEEE International Symposium in Intelligent Control*, Mexico City, Mexico, 2001, pp. 1-6.

[45] Y. Yang, A. Minai, and M. Polycarpou, "Evidential Map-Building Approaches for Multi-UAV Cooperative Search," in *American Control Conference*, Portland, OR, 2005, pp. 116-121.

[46] B. Lavis, T. Furukawa, and H. F. Durrant-Whyte, "Dynamic Space Reconfiguration for Bayesian Search-and-Tracking with Moving Targets," *Autonomous Robots*, vol. 24, pp. 387–399, May 2008.

[47] J. Kadane, "Optimal whereabouts search," *Operations Research*, vol. 19, no. 4, pp. 894–904, Nov. 1971.

[48] I. Wegener, "The discrete sequential search problem with non-random cost and overlook probabilities," *Mathematical Operation Research*, vol. 5, no. 3, pp 373–380, Aug. 1980

[49] T. H. Chung and J. W. Burdick, "Analysis of search decision making using probabilistic search strategies," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp.132 -144, Feb. 2012

[50] L. D. Stone, C. A. Barlow, and T. L. Corwin, "Bayesian Multiple Target Tracking", Boston, MA: Artech House, 1999

[51] Y. Wang and I. Hussein, "Bayesian-based decision-making for object search and classification," *IEEE Transaction on Control System Technology*, vol. 19, no. 6, pp. 1639-1647, Sep. 2011.

[52] A. Tsourdos, B. White, M. Shanmugavel, "Cooperative Path Planning of Unmanned Aerial Vehicles," New York, NY: John Wiley & Sons, 2010.

[53] Z. Tang, U. Ozguner, "Motion Planning for Multi target Surveillance with Mobile Sensor Agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898- 908, Sep. 2005.

[54] A. Ghaffarkhah and Y. Mostofi, "Path planning for networked robotic surveillance," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3560 - 3575, July 2012.

[55] L. Lin and M. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *IEEE/RSJ international conference on intelligent robots and systems*, St. Louis, MO, 2009, pp. 709–714.

[56] J. Baxter, E. Burke, J. Garibaldi, M. and Norman, "Multi-robot search and rescue: A potential field based approach," *Autonomous Robots and agents*, vol. 76, no. 1, pp. 9-16, 2007

[57] D. W. Casbeer, S.-M. Li, R. W. Beard, R. K. Mehra, and T. W. McLain, "Forest fire monitoring with multiple small UAVs," in *American Control Conference*, Portland, OR, 2005, pp. 3530–3535.

[58] P. Chandler, S. Rasmussan, and M. Pachter, "UAV cooperative path planning," in *AIAA Guidance, Navigation, and Control Conference,* Denver, CO, 2000, pp.1-9.

[59] M. Jun, and R. D'Andra, "Path planning for unmanned aerial vehicles in uncertain and adversarial environments," in *Cooperative Control: Models, Applications and Algorithms,* Kluwer academic, 2003, pp. 95-110.

[60] L. E. Kavarki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmap for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566-580, Aug 1996.

[61] P. Cheng, Z. Shen, and S. M. LaValle, "RRT-based trajectory design for autonomous automobile and aircrafts," *Archives of Control Science*, vol. 11, no. 3-4, pp. 167-194, 2001.

[62] J. N. Amin, J. D. Boskovic, and R. K. Mehra, "A fast and efficient approach to path planning for unmanned vehicles," in *AIAA Guidance, Navigation, and Control Conference*, Denver, CO, 2006, pp. 1-8.

[63] Y. Eun and H. Bang, "Cooperative control of multiple unmanned aerial vehicles using potential field theory," *Journal of Aircraft*, vol. 43, no. 6, pp.1805-1814, Dec. 2006

[64] F. A. Cosio and M. A. P. Castaneda, "Autonomous robot navigation using adaptive potential fields," *Mathematical and Computer Modelling*, vol. 40, no. 9-10, pp. 2479-2484, Nov. 2004.

[65] S. Spires and S. Goldsmith, "Exhaustive Geographic Search with Mobile Robots along Space-Filling Curves," in *International Workshop on Collective Robotics*, Paris, France, 1998, pp. 1–12.

[66] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Distributed Covering by Ant Robots using Evaporating Traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, Oct.1999.

[67] S. Yang, and C. Luo, "A Neural Network Approach to Complete Coverage Path Planning," *IEEE Transaction on Systems, Man, Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 718–725, Jan. 2004.

[68] H. Choset, "Coverage for Robotics—A Survey of Recent Results," Annals of Mathematics and Artificial Intelligence, vol. 31, no .1–4, pp. 113–126, Oct. 2001.

[69] T. A. Ademoye, A. Davari, and W. Cao, "Three dimensional obstacle avoidance maneuver planning using mixed integer linear programming," in *Robotics and Automations Conference*, Honolulu, HI, 2006, pp. 1-8.

[70] T. Schouwenaars, E. Feron, and J. How, "Multiple-vehicle path planning for non-line of sight communication," in *American Control Conference*, Minneapolis, MN, 2006, pp. 1-6.

[71] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference*, 2002, Anchorage, AK, pp. 1936-1941.

[72] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, "Evolutionary route planner for unmanned aerial vehicles," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 609-620, Aug 2005.

[73] S. Mittal and K. Deb, "Three dimensional offline path planning for UAVs using multi-objective evolutionary algorithms," in *IEEE Congress on Evolutionary Computation*, Singapore, 2007, pp. 3195 – 3202.

[74] X. Wu, Z. Feng, J. Zhu, and R. Allan, "GA-based path planning for multiple UAVs," *International Journal of Control*, vol. 80, no. 7, pp.1180-1185, 2007.

[75] P. B. Sujit and D. Ghose, "Optimal uncertainty reduction search using the k-shortest path algorithm," in *American Control Conference*, Denver, CO, 2003, pp. 3269–3274,

[76] P. B. Sujit, and D. Ghose, "Multiple Agent Search of an Unknown Environment Using Game Theoretical Models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 1–4, pp. 491–508, 2004

[77] P. B. Sujit and D. Ghose, "Multiple Agent Search of an Unknown Environment Using Game Theoretical Models," in *American Control Conference*, Boston, MI, 2004, pp. 5564–5569.

[78] C. A. Rabbath, N. Léchevin, "Safety and Reliability in Cooperating Unmanned Aerial Systems," World Scientific, 2010

[79] R. Bellman, "Dynamic Programming," Princeton, NJ: Princeton University Press, 1957.

[80] M. L. Puterman, "Markov Decision Processes," New York, NY: John Wiley and Sons, 1994.

[81] R. Beard and T. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," in *IEEE Conference on Decision and Control*, Maui, HI, 2003, pp. 25–30.

[82] M. Flint, E. Fernandez-Gaucherand, and M. Polycarpou, "Cooperative control for UAVs searching risky environments for targets," in *IEEE Conference on Decision and Control*, Maui, HI, 2003, pp.3567 -3572.

[83] D. Rogers, R. Plante, R. Wong, and J. Evans, "Aggregation and disaggregation techniques and methodology in optimization," *Operations Research*, vol. 39, no. 4, pp. 553-582, July 1991.

[84] D. Bertsekas, and D. Castanon, "Adaptive aggregation methods for infinite horizon dynamic programming," *IEEE Transactions on Automatic Control,* vol. 34, no. 6, pp. 589-598, June 1989.

[85] R. Luus, "Iterative Dynamic Programming," New York, NY:Chapman and Hall/CRC, 2000.

[86] W. B. Powell, "Approximate Dynamic Programming: Solving the curses of dimensionality," New York, NY: John Wiley and Sons, 2007.

[87] D. P. Choi, and B. Van Roy, "A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning," *Discrete Event Dynamic Systems*, vol. 16, no.2, pp. 207-239, Apr. 2006.

[88] D. Bertsekas, and J. Tsitsiklis, "Neuro-Dynamic Programming," Belmont, MA: Athena Scientific, 1996.

[89] I. I. Hussein and D. Stipanovic, "Effective Coverage Control for Mobile Sensor Networks with Guaranteed Collision Avoidance," *IEEE Transactions on Control Systems Technology, Special Issue on Multi-Vehicle Systems Cooperative Control with Applications*, vol. 15, no. 4, pp. 642–657, July 2007.

[90] Y. Wang and I. Hussein, "Awareness coverage control over large-scale domains with intermittent communications," *IEEE Transactions on Automation. Control*, vol. 55, no. 8, pp. 1850–1859, Aug. 2010.

[91] K.R. Guruprasad and D. Ghose, "Automated multi-agent search using centroidal Voronoi configuration," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 420-4234, Apr. 2011.

[92] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, July 1999.

[93] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions in Information Theory*, vol. 28. No. 2, pp. 129–137, Mar. 1982.

[94] J. Cortes, S. Martinez, T. Karatas and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[95] 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. 4, pp. 691–719, Oct. 2005.

[96] G. Wang, G. Cao and T. La Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, June 2006.

[97] J. Cortes, "Coverage optimization and spatial load balancing by robotic sensor networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 749–754, Mar. 2010.

[98] J. S. Marier, C. A. Rabbath, N. Lechevin, "Health-Aware Coverage Control With Application to a Team of Small UAVs," *IEEE Transaction on Control Systems Technology*, vol. 21, no. 5, pp. 1719-1730, Sep. 2013.

[99] J. Hu and Z. Xu, "Distributed cooperative control for deployment and task allocation of unmanned aerial vehicle networks," *IET Control Theory and Applications*, vol. 7, no. 11, pp. 1574–1582, Nov. 2013.

[100] C. H. Caicedo-Nunez and M. Zefran, "A coverage algorithm for a class of non-convex regions," in *IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4244–4249.

[101] J. W. Durham, R. Carli, P. Frasca and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, Feb. 2012.

[102] J. S. Marier, C. A. Rabbath, N. Lechevin, "Visibility-limited Coverage Control Using Nonsmooth Optimization," in American Control Conference, Montreal, Canada, 2012, pp.6029-6034.

[103] Y. Stergiopoulos and A. Tzes, "Convex Voronoi inspired space partitioning for heterogeneous networks: a coverage-oriented approach," *IET Control Theory and Applications*, vol. 4, no. 12, pp. 2802–2812, Dec. 2010.

[104] T. P. Lambrou and C. G. Panayiotou, "Collaborative area monitoring using wireless sensor networks with stationary and mobile nodes," *EURASIP Journal on Advances in Signal Processing*, Vol. 2009, pp. 1-16, 2009.

[105] M. Schwager, D. Rus and J. J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, Mar. 2009.

[106] A. Dirafzoon, S. Salehizadeh, S. Emrani, M. Menhaj and A. Afshar, "Coverage control for mobile sensing robots in unknown environments using neural network," in *IEEE International Symposium on Intelligent Control*, Yokohama, Japan, 2010, pp. 1482–1487.

[107] S. Martinez, “Distributed interpolation schemes for field estimation by mobile sensor networks,” *IEEE Transactions on Control Systems Technology*, vol. 18, no.2, pp. 491–500, Feb. 2010.

[108] S. Bhattacharya, N. Michael and V. Kumar, “Distributed coverage and exploration in unknown non-convex environments,” *Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics*, vol. 83, pp. 61–75, 2013.

[109] D. Enns, D. Bugajski, and S. Pratt, “Guidance and Control for Cooperative Search,” in *American Control Conference*, Anchorage, AK, 2002, pp. 1923–1929.

[110] W. Burgard, M. Moors, C. Stachniss, F.E. Schneider, “Coordinated multi-robot exploration,” *IEEE Transactions on Robotics,* vol. 21, no. 3, pp. 376 – 386, June 2005

[111] F. Hu, Q. Hao, “Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning” Taylor & Francis, 2012.

[112] R. Niu, and P.K. Varshney, “Target Location Estimation in Sensor Networks With Quantized Data” IEEE Transactions on Signal Processing, vol. 54, no. 12, pp. 4519-4528, Dec. 2006

[113] N. Xiong; P. Svensson, "Multi-sensor management for information fusion: issues and approaches". Information Fusion. Vol. 3, no. 2, pp. 163–186, June 2002.

[114] B. Sinopoli, M. Micheli, G. Donatot. T. J. Koo, “Vision Based Navigation for an Unmanned Aerial Vehicle,” in IEEE International Conference on Robotics and Automation, Seoul, Korea, 2001, pp. 1757-1764.

[115] D. P. Bertsekas, “Dynamic Programming and Optimal Control,” vol 1. Athena Scientific, Belmont, Massachusetts, 2nd edition, 2000

[116] A. P. Tirumalai, B. G. Schunck, and R. C. Jain, “Evidential reasoning for building environment maps,” *IEEE Transactions on System, Man, and Cybernetics*, vol. 25, no. 1, pp. 10-20, Jan. 1995.

[117] Claude E. Shannon, “A Mathematical Theory of Communication”, *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, Oct. 1948.

[118] F. Aurenhammer, R. Klein, D. T. Lee “Voronoi diagrams and Delaunay triangulations,” Hackensack, NJ : World Scientific, 2013

[119] Z. Drezner., “Facility Location: A Survey of Applications and Methods,” Springer Verlag, New York, NY, 1995.

[120] F. Bullo, J. Cort´es and S. Martinez, “Distributed Control of Robotic Networks” Princeton, NJ: Princeton University Press, 2009, Available at http://www.coordinationbook.info.

[121] http://people.ee.ethz.ch/~mpt/2/

[122] M. Mirzaei, F. Sharifi, B. W. Gordon, C. A. Rabbath, and Y. M. Zhang, “Cooperative multi-vehicle search and coverage problem in uncertain environments,” in *IEEE Conference on Decision and Control and European Control Conference*, Orlando, USA, 2011, pp. 4140–4145.

[123] http://www.quanser.com/products/qbot2

[124] http://www.quanser.com/english/downloads/solutions/QuaRC_PIS_080310.pdf.

[125] https://www.optitrack.com/products/flex-3/

[126] G. Oriolo, A. De Luca and M. Vendittelli, “WMR control via dynamic feedback linearization: Design, implementation, and experimental validation,” *IEEE Transaction Control System Technology*, vol. 10, no. 6, pp. 835–852, 2002.

[127] F. Sharifi, M. Mirzaei, Y. Zhang, B. W. Gordon, “Cooperative Multi-Vehicle Search and Coverage Problem in an Uncertain Environment,” Unmanned Systems, Vol. 3, No. 1, 2015, pp. 35–47.

[128] L. Pimenta, V. Kumar, R. Mesquita and G. Pereira, “Sensing and Coverage for a Network of Heterogeneous Robots”, in *IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 3947 - 3952.

[129]. S. Bhattacharya, N. Michael, and V. Kumar, “Distributed coverage and exploration in unknown non-convex environments,” in *International Symposium on Distributed Autonomous Robotics Systems*, Lausanne, Switzerland, 2010, pp. 1–14.

[130] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Jan. 1959.

[131] M. M. Polycarpou , Y. Yang and K. M. Passino, “Cooperative control of distributed multi-agent systems,” *IEEE Control System Magazine,* 2001

[132] B. Bethke, J. How, and J. Vian, "Group health management of UAV teams with applications to persistent surveillance," in American Control Conference, 2008, Jun. 2008, pp. 3145-3150.

[133] B. Aronov, “On the geodesic Voronoi diagram of point sites in a simple polygon,” in *the Symposium on Computational Geometry,* New York, NY, 1987, pp. 39–49.