COOPERATIVE CONTROL, LEARNING AND SENSING IN MOBILE SENSOR NETWORKS

By

HUNG MANH LA

Bachelor of Science
Thai Nguyen University of Technology
Thai Nguyen City, Thai Nguyen, Vietnam
2001

Master of Science
Thai Nguyen University of Technology
Thai Nguyen City, Thai Nguyen, Vietnam
2003

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2011

COOPERATIVE CONTROL, LEARNING AND SENSING IN MOBILE SENSOR NETWORKS

Dissertation Approved:

Dr. Weihua Sheng

Dissertation Advisor

Dr. Gary G. Yen

Dr. Satish Bukkapatnam

Dr. Qi Cheng

Dr. Mark Payton

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this chapter, we first present the motivation of our work, then present the challenges, the contributions and the current state of the art of the cooperative control, learning and sensing in MSNs.

1.1 Motivation

Mobile sensor networks (MSNs) [3], one type of sensor networks [4, 5, 6, 7], have been studied by many researchers in recent years. A typical mobile sensor is a mobile robot with various sensors such as camera, sonar or laser for sensing and navigation. Mobile sensor networks have several advantages over stationary sensor networks, such as the adaptation to environmental changes and reconfigurability for better sensing performance. Therefore mobile sensor networks can be applied in many applications including cooperative detection of toxic chemicals in contaminated environments [8, 9, 10]; environment exploring, monitoring and coverage [11, 12, 13]; performing search and rescue operation after disasters [14, 15]; target tracking [16, 17, 18] and protection of endangered species [19].

A main issue for multiple mobile sensors move together is that these sensors have to avoid collision among them, which requires the use of cooperative control methods [20, 21, 22, 23, 24]. One of these methods is flocking control [23]. We know that flocking or schooling is a phenomenon that a number of mobile agents move together and interact with each other while ensuring no collision, velocity matching, and flock centering [25]. In the nature, schools of fish (see Figure 1.1), birds, ants, and bees, etc. demonstrate the phenomenon of flocking [26]. The problem of flocking has been studied for many years.

(a) (b)

Figure 1.1: (a) Schooling of fish. (b) A predator and a school of fish (source: www.inmagine.com).

It has attracted many researchers in physics [27, 28], mathematics [29], biology [30], and especially in control science in recent years [31, 32, 33, 23, 34, 35, 36, 37, 38, 39, 40].

There are several interesting features established by the school of fish or flock of birds. These features can inspire us to design cooperative control, learning and sensing algorithms for MSNs.

- Fish school and bird flock can track a target (source of food) efficiently while avoiding obstacles. This inspires us to design a cooperative control algorithm that can allow mobile sensors to track a target better in cluttered environments.

- Each individual fish or bird communicates/interacts with its neighbors within its limited sensing range in order to move in the same direction as its neighbors, remain close to its neighbor, and avoid collision with its neighbors [25]. Based on only these local communications/interactions, the fish school or bird flock can still achieve a global goal. For example, in some cases only some individuals have the knowledge about the location of a food source and migration route, but the fish school or bird flock can still find the food source and track the migration route efficiently [41, 42]. Inspired by this natural ability we would like to design a cooperative control algorithm that can allow mobile sensors to track a target when only a very small subset of

them know the information of the target while maintaining the network connectivity.

- Fish school and bird flock also have ability to change their size of the formation in order to adapt to the environments. This motivates us to design an adaptive control algorithm for an MSN that can automatically adjust its size (shrink/recover) in order to adapt to the complex environments while maintaining the network connectivity and similar topology.

- Fish school and bird flock can track multiple food sources (targets) simultaneously. This ability encourages us to design a splitting/merging algorithm that can allow an MSN to track multiple moving targets simultaneously and efficiently in a dynamic fashion.

- Each individual fish or bird may not sense the position and velocity of its neighbors accurately, but it can still move with its neighbors and maintain the cohesion with them. This feature inspires us to design flocking control algorithms that can allow mobile sensors to work in noisy environments while maintaining the cohesion to the network.

- Fish schooling and bird flocking together can help the individual to avoid predators because many moving individuals create a sensory overload for the predator's visual channel [43, 44, 45] (see Figure 1.1b). This motivates us to design a cooperative learning algorithm that can allow an MSN to learn to avoid the enemy (predator) in a distributed fashion while maintaining the network connectivity and similar topology.

- Finally, each individual fish or bird only interacts locally, but as a whole the fish school or bird flock can agree on the same velocity (velocity matching ability) through distributed consensus. Understanding this feature can help us design cooperative and active sensing algorithms for an MSN which can allow each sensor to find an agreement among observations of itself and its neighbors by reaching consensus.

1.2 Challenges

Development of cooperative control, learning and sensing algorithms in a distributed fashion for MSNs is very challenging. These algorithms have to be performed at each sensor node using only local information, while as a whole they exhibit collective intelligence and achieve a global goal. In a resource-constrained multi-agent system, the communication range and sensing range of each agent are small compared to the size of the environments. Hence, agents cannot accomplish the mission without careful design of cooperative control, learning and sensing algorithms. Here are several challenges in designing cooperative control, learning and sensing algorithms for MSNs.

- **Cooperative Control in MSNs:**

  First, designing flocking control algorithms which maintain the target tracking performance in cluttered environments is a challenging task. In these environments, the agents usually get stuck behind the obstacles and sometimes can not follow the target [23], [17], [34], [35], [46], therefore causing poor tracking performance.

  Second, designing a distributed flocking control algorithm which can still perform well in terms of better tracking performance and connectivity maintenance when only few agents have information of the target is a difficult task. Flocking control algorithms [23] can allow agents to move together without collision and track a target. However, they are designed under the assumption that all agents have the information of the target. Su *et al*. [34, 35, 46] relaxed this assumption, however the network connectivity is not maintained.

  Third, designing an adaptive flocking control algorithm that can adapt to the complex environments, for example passing through a narrow space among obstacles, while maintaining connectivity, tracking performance and similar formation is a challenging task. Existing works [47, 48] do not consider controlling the size of the network, hence the connectivity and topology are not maintained.

Fourth, tracking multiple targets simultaneously in a dynamic fashion in a MSN is difficult, since this requires that some sensors should split from the existing formation(s) to track new targets while ensuring the least disturbance to other sensors. This raises the question of which sensors should split from the existing formation and how they should split. In addition, when some targets disappear the sensors which are tracking these targets should rejoin (merge with) the existing groups that are still tracking targets.

Fifth, designing distributed flocking control algorithms for MSNs which can still perform well when the measurements are corrupted by noise is very challenging. Existing works [37, 38, 23, 34, 35, 46] do not consider this issue in their flocking control algorithms.

- **Cooperative Learning in MSNs:**

  Designing an intelligent MSN which can provide ability to learn to avoid enemy (predator) while maintaining the network topology and connectivity is difficult, since this is a distributed decision making problem where each individual has a number of options (safe places) to choose from when the predators appear. Often in these decisions there is a benefit for consensus, where all individuals choose the same safe place. However, the existing consensus methods [40, 49, 50, 51, 52, 53, 54, 55, 56] require a connected network in which all robots can communicate with each other. This may not be valid in real environments because some robots may not connect to the network during the escape. In that case the consensus algorithms will fail. Therefore, there is a need to reach consensus even when the robots cannot connect to the network at sometimes.

- **Cooperative Sensing in MSNs:**

  Designing a distributed sensor fusion algorithm for MSNs with an emphasis on the task of environment estimation and mapping is an open problem, since it requires a

combination of cooperative sensing, cooperative motion control, and complete coverage path planning while using only local information. Existing works in the area of cooperative sensing using MSNs [57, 58, 59, 60, 61, 11, 62, 13] focus on target(s) tracking, environment exploring, sampling, modeling, and coverage. The problem of environmental estimation and mapping based on multi-agent cooperative and distributed sensing is still an open research.

1.3 Contributions

This work contributes to the research of MSNs by developing cooperative control, learning and sensing in a distributed fashion to realize coordinated motion control and intelligent situational awareness. Here are the main contributions of our work:

- **Cooperative Control:**

  We propose a novel approach to the problem of flocking control of a MSN to track and observe a moving target. Flocking algorithms that constrain the center of mass of positions and velocities of all mobile sensors in each group (Single-CoM) or the center of mass of position and velocity of each sensor and its neighbors (Multi-CoM) are developed. The main benefit of both algorithms is to make the center of mass (CoM) of each group track the target in the obstacle space. This makes the mobile sensors surround the target closely.

  We study the flocking control in the case of a small subset of informed agents. In nature, only few agents in a group have the information of the target, such as knowledge about the location of a food source, or the migration route. However, they can still flock together in a group based on local information. Inspired by this natural phenomenon, we propose a flocking control algorithm to coordinate the motion of multiple agents. Based on our algorithm, all agents can form a network, maintain connectivity and track the target even only very few of them know the information of

the target.

To deal with changing environments, for example in the case when the mobile sensor networks have to pass through a narrow space among obstacles, we propose an adaptive flocking control algorithm in which each agent (sensor) can cooperatively learn the network's parameters to decide the size of network in a decentralized fashion so that the connectivity, formation and tracking performance can be improved.

In the scenario of multiple dynamic target tracking, to solve the problem of sensor splitting/merging, a seed growing graph partition (SGGP) algorithm is proposed.

In noisy environments, a flocking control algorithm is proposed to coordinate the activities of multiple agents through noisy measurements. Based on our algorithm, all agents can form a network and maintain connectivity. This is of great advantage for agents to exchange information. We show that even with noisy measurements the flocks can achieve cohesion and follow the moving target. The stability and scalability of our algorithm are also investigated.

- **Cooperative Learning:**

  We propose a hybrid system that integrates reinforcement learning and flocking control in order to create adaptive and intelligent multi-robot systems. We study two problems in multi-robot concurrent learning of cooperative behaviors: (1) how to generate efficient combination of high level behaviors (discrete states and actions) and low level behaviors (continuous states and actions) for multi-robot cooperation; (2) how to conduct concurrent learning in a distributed fashion. To evaluate our theoretical framework, we apply it to enable multi-robot networks to learn avoiding predators while maintaining network topology and connectivity. We also investigate the stability and scalability of our algorithm.

- **Cooperative Sensing:**

  We propose a novel method for multiple mobile sensor nodes to build a map of a scalar field through noisy measurements. Our method consists of three parts. First, we develop a distributed sensor fusion algorithm integrating two different distributed consensus filters to achieve cooperative sensing among sensor nodes. Second, we use the distributed flocking control algorithm to drive the center of the mobile sensor formation to track the desired paths. Third, we build a path planning strategy to obtain a complete coverage of the field. Additionally, we extend our cooperative sensing method to active sensing in order to improve the sensing performance.

1.4 Literature Review

In this section, we review existing literature related to our work, which includes cooperative control, multi-agent learning, and cooperative sensing.

1.4.1 Cooperative Control

Cooperative control in multi-robot systems has been receiving growing attention in recent years [63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73], and its applications include multi-target tracking [63, 64], multi-vehicle formation control [65, 66, 67, 68], optimization based control [69, 70], cooperative control with limited communications [71, 72], graph-rigidity based control [74, 75, 76], and data gathering using mobile sensors [73]. In this subsection we review the existing works in the area of cooperative control which includes flocking control, adaptive flocking control, multiple targets tracking in both stationary sensor networks and mobile sensor networks.

Flocking Control

Flocking control has been studied by many researchers in recent years [77, 78, 64, 79, 80]. Wang and Gu [40] presented a survey of recent research achievements of robot flocking.

Their paper gave an overview of the related basic knowledge of graph theory, potential function, network communication and system stability analysis. In [23], a theoretical framework for design and analysis of distributed flocking algorithms was proposed. These algorithms solved the flocking control in the absence and presence of obstacles. The static and dynamic virtual leaders were used as a navigational feedback for all mobile agents. An extension of the flocking control algorithms in [23], flocking of agents with a virtual leader in the case of a minority of informed agents and in the case of varying velocity of the virtual leader, was presented in [34, 35, 46]. Shi and Wang [36] investigated the dynamic properties of mobile agents for the case where the state of the virtual leader is time varying and the topology of the neighboring relations between agents is dynamic was proposed. Anderson *et al*. [31] demonstrated a new technique for generating the constrained group animations of flocks in which users can impose constraints on agents' positions at any time in the animation, or control the entire group to meet the shape constraints. Tanner *et al*. [37, 38] studied the stability properties of a system of multiple mobile agents with double integrator dynamics in the case of fixed and dynamic topologies. In addition, the experimental implementation of flocking algorithms proposed in [37] and [38] on wheeled mobile robots was presented in [39]. Gervasi and Prencipe [33] studied the distributed coordination and control of a set of asynchronous, anonymous, memoryless mobile vehicles in the case of no communication among the vehicles. In particular, their paper analyzed the problem of flocking in a certain pattern and following a designated leader vehicle, while maintaining the pattern. Olfati-Saber [17] developed a distributed flocking algorithm for mobile sensor networks to track a moving target. In his paper, an extension of a distributed Kalman filtering algorithm was used for the sensors to estimate the target's position. In [32], a scalable multi-vehicle platform was developed to demonstrate a cooperative control algorithm in mobile sensor networks. Their flocking algorithm was implemented with five TXT-1 monster truck robots.

Adaptive Flocking Control

Adaptive flocking control, an extension of flocking control, has also gained attention from researchers in recent years. Folino and Spezzano [81] presented a parallel clustering algorithm based on the use of swarm intelligence techniques. Their algorithm is a combination of a smart exploratory strategy based on a flock of birds and a density-based cluster algorithm to discover clusters of arbitrary shape and size in spatial data. Yang *et al*. [47] proposed an adaptive flocking control algorithm to avoid collision among robots themselves and between robots and obstacles. However, their algorithm did not consider the problem of formation, connectivity and tracking performance. Lee and Chong [48] proposed a decentralized approach for adaptive flocking of swarms of mobile robots to navigate autonomously in complex environments populated with obstacles. The problem of splitting/merging mobile robots in the network according to the environment conditions is addressed in their paper. In their work, the problem of controlling the size of the network was not considered.

Multiple Targets Tracking

Multiple targets tracking in mobile sensor networks has received adequate attention in the last decade. Jung *et al*. [82] introduced a region-based approach to address the problem of multiple targets tracking using a network of communicating robots and stationary sensors. A coarse deployment controller distributes robots across regions using a topological map, and a target-following controller maximizes the number of tracked targets within a region. Chung *et al*. [57] proposed a gradient search based decentralized algorithm for the problem of active sensing using multiple cooperative sensor nodes for distributed sensing to estimate the state of dynamic targets. Tang and Ozguner [83] investigated the motion planning for a limited number of mobile sensor agents in an environment with multiple dynamic targets. The motion planning problem is formulated as an optimization problem to minimize the average time duration between two consecutive observations of each target.

Jung and Sukhatme [63] proposed an algorithm based on treating the densities of robots and targets as properties of the environment in which they are embedded to improve the target tracking performance. Kamath *et al*. [3] studied the problem of motion planning and sensor assignment in a mobile sensor network for tracking multiple targets. The triangulation based tracking where two sensors merge their measurements to estimate the position of a target is considered. Kolling and Carpin [84] presented a distributed control algorithm for multiple targets surveillance by multiple robots. Their algorithm utilizes information from sensors and communication to predict the minimum time before a robot loses a target.

Sensor network partitioning, a fundamental technique for sensor networks dealing with multiple targets tracking, has been studied by many researchers. The methods for network partitioning can be divided into centralized and decentralized. For centralized graph partition, there are several algorithms such as the decomposition scheme to partition a given graph into compactly connected two terminal subgraphs [85], a graph clustering method based on the minimum cuts within a graph [86] , a new graphical adaptation of the k-medoids algorithm [87] and the Girvan-Newman method [88]. For distributed graph partition, Derbel and Mosbah [89] proposed a linear time distributed algorithm for decomposing a graph into a disjointed set of clusters. In [90, 91], Goebels *et al*. presented a neighborhood-based strategy, a border switch strategy, and an exchange target strategy for the partitioning of large sets of agents into multiple groups. Derbel *et al*. [92] proposed efficient deterministic and randomized distributed algorithms for decomposing a graph into a disjointed set of connected clusters with small radius and few inter-cluster edges. Bettstetter [93] gave equations for the cluster density and cluster order of hemogeneously distributed nodes running the distributed mobile adaptive clustering algorithm. Virrankoski and Savvides [94] proposed a topology adaptive spatial clustering (TASC) for sensor networks. Durresi and Paruchuri [95] presented an adaptive clustering protocol for sensor network. This approach is based on the covering problem that aims at covering an area with minimum possible circular disk assuming ideal conditions. Meka and Singh [96] presented a

distributed algorithm called ELink based on a quad-tree decomposition and a level by level expansion using sentinel sets. Belghith *et al*. [97] proposed a novel distributed clustering algorithm for ah-hoc networks. Their algorithm is based on a synchronized and layered process.

Summary of Cooperative Control

In general, for cooperative control based on flocking control, most works focus on the configuration and topology of flocks. For single target tracking based on the flocking control, the literatures solve the problem of estimating the target's state by using the distributed Kalman filter, or solve the problem of target tracking while a minority of agents in the network have the knowledge of the target. Their algorithms work well in free space, but in the obstacle space they have some limitations such as bad tracking performance, low speed and connectivity loss. To our best knowledge, for adaptive flocking control most of existing works focused on the coordination, formation and splitting/merging problems in both fixed and switching topologies. For multiple targets tracking all of reviewed literatures solve the tracking problem in both stationary and mobile sensor networks without paying attention to the network formation such as α-lattice. The problem of graph partitioning focuses on both centralized and decentralized methods, and most of them decompose the network based on the density of node's distribution. This means that the size of subgraphs after decomposing are not predetermined. There are several open problems in cooperative control based on flocking control such as:

- How to utilize the a minority of informed agents to lead the whole network to track the target while maintaining the connectivity.
- How to control the size of the network in a decentralized and adaptive fashion in complex or changing environment while maintaining connectivity, tracking performance and similar formation.

- How to partition the MSN to track multiple moving target while minimizing the total energy consumption and time consumption.

- How to design a flocking control algorithm to maintain the cohesion among agents while the measurements are corrupted by noise.

1.4.2 Cooperative Learning in MSNs

Through cooperative learning agents in a MSN attempt via their interactions to jointly solve tasks or to maximize utility [98]. Cooperative learning has been studied by many researchers in recent years. The overview of cooperative learning including reinforcement learning, evolutionary computation, game theory, complex systems, agent modeling, and robotics can be found in [98, 99]. Reinforcement learning, one of the most powerful machine learning techniques, has been developed for multi-robot systems that allow robots to learn cooperation [100, 101, 99, 102]. Reinforcement learning techniques for solving cooperative problems in teams of homogeneous robots such as the problem of maintaining a formation of mobile robots are studied in [103]. Cooperative reinforcement learning associating VQQL (Vector Quantization to Q Learning) is developed and applied for multi-robot observation of multiple moving targets [104, 105, 101]. In their work, they solved two problems. The first one focuses on defining the reinforcement signal for each robot in a cooperative team to achieve a global goal. The second one is working with large domains, where the amount of data can be large and different in each moment of a learning step. As a result, their work achieved successful cooperative behaviours, but the learned behaviors are still discrete, and the learning space is still large. Other work on cooperative multi-robot reinforcement learning [102] tried to reduce the learning space by using a hybrid state space that integrates a neural perception module and a progressive rescheduling algorithm. Their algorithm can on-line execute and learn through experience to adapt to environmental uncertainties and enhance performance. However, their work still relies on discrete and finite state/action spaces.

To our best knowledge most of existing works in the area of cooperative reinforcement learning assumes discrete and finite state/action spaces; therefore, it is difficult to directly apply reinforcement learning to most real world applications that inherently involve with continuous and infinite space. Furthermore, even if the states can be discretized, the learned behaviors are still discrete. In addition, the switching of discrete behaviors usually causes the control of the robots to become non-smooth, which is undesirable in most applications. The open question is *can we combine reinforcement learning and flocking control to create a general framework for intelligent robot systems that can allow (1) to generate efficient combination of high level behaviors (discrete states and actions) and low level behaviors (continuous states and actions) for multi-robot cooperation; (2) and to conduct concurrent learning in a distributed fashion?*

1.4.3 Cooperative Sensing in MSNs

Cooperative sensing in MSNs has recently attracted researchers in control engineering [11, 12, 13], and it can be utilized in target tracking, and environmental mapping, monitoring, exploration and coverage.

Cooperative sensing networks have been developed [106, 60, 13] for environmental sampling and exploring. In [106], underwater vehicles are deployed to measure temperature, currents, and other distributed oceanographic signals. The vehicles communicate via an acoustic local area network and coordinate their motion in response to local sensing information and to evolving environments. This mobile sensor network has the ability to sample the environment adaptively in space and time. By identifying evolving temperature and current gradients with higher accuracy and resolution than current static sensors, this technology could lead to the development and validation of improved oceanographic models. In [60], a class of underwater vehicles are used to obtain a sampling coverage over a large area. A cooperative control method is proposed to control vehicles to generate patterns on closed smooth curves. To further improve the cooperative sensing performance,

both the cooperative motion control and the cooperative sensing are integrated based on cooperative Kalman filter [13] to control the shape of the sensor node formation in order to minimize error in the estimates.

Other significant works in cooperative sensing developing for environmental estimation, coverage and modeling can be found in [59, 11, 12, 62]. Cooperative sensing based on the gradient descent algorithms to obtain the optimal coverage is developed in [59]. For dynamic environment coverage, a control strategy based on the discrete Kalman filter is developed [11]. The approach relies on using the Kalman filter to estimate the field and on the filter's prediction step to plan the vehicles' next move to maximize the quality of the field estimate. In [62], an optimal filtering approach toward fusing local sensor data into a global model of the environment is developed. Their approach is based on the use of average consensus filters to distributedly fuse the sensory data through the communication network. Along with the consensus filters, the control laws are developed for mobile sensors to move to maximize their sensory information relative to current uncertainties in the model.

Additionally, cooperative sensing for estimating the state of dynamic targets can be found in [57, 58]. The localization and tracking tasks of dynamic targets are addressed in [58]. In their work, the mobility of sensing agents is utilized to improve this quality of sensing. However, their gradient controller for cooperative sensing is designed in centralized way. The extension to make the control algorithm in [58] distributedly is proposed in [61], and both formation control and cooperative sensing are integrated to improve the sensing performance.

Overall, all of the existing works in the area of cooperative sensing using MSNs focus on target(s) tracking, environment exploring, sampling, modeling, and coverage. *The problem of environmental estimation and mapping based on multi-agent cooperative and distributed sensing is still open research.*

1.5 Organization of This Dissertation

The rest of this dissertation is organized as follows. In Chapter 2 we first present the potential field based moving target tracking algorithm for a single mobile sensor and then extend it to multiple mobile sensors coordination based on flocking control. Chapter 3 describes the flocking control algorithm with a minority of informed agents; the adaptive flocking control algorithm for single target tracking and observing; and the algorithm for dynamic multiple targets tracking and observing, respectively. Chapter 4 presents the flocking control algorithms in noisy environments. Chapter 5 presents a hybrid system of flocking control and reinforcement learning for cooperative predator avoidance. Chapter 6 presents the cooperative sensing algorithm based on distributed consensus filters and flocking control, then extends to cooperative and active sensing algorithm. Conclusions and future work are given in Chapter 7. The flowchart of the organization of the dissertation is illustrated in Figure 1.2.

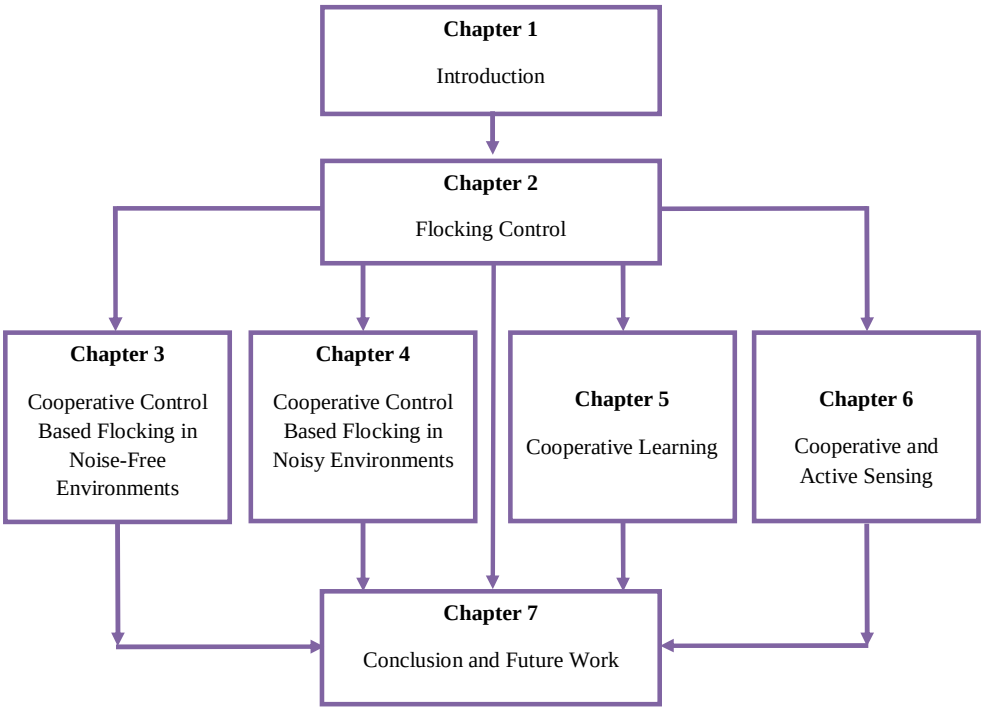


Figure 1.2: The organization of the dissertation.

CHAPTER 2

FLOCKING CONTROL FOR DYNAMIC TARGET TRACKING

In this chapter, we first present a potential field approach for a single mobile sensor node to track a moving target. This establishes the background of the potential field method that is extended to multiple mobile sensor nodes. Then, we present the flocking control background which establishes three basic flocking rules: no collision among agents, velocity matching among agents, and flocking centering. We extend the existing flocking control to more constraints such as *Single-CoM* (Center of Mass) or *Multi-CoM* to allow MSNs to track a target better in cluttered environments. In addition, stability analysis and simulation results with a comparison among the flocking control without CoM (*No-CoM*), *Single-CoM* and *Multi-CoM*, respectively, are given.

This chapter is organized as follows. Section 2.1 presents a potential field approach for one mobile sensor node to track a moving target. Section 2.2 presents flocking control for MSNs to track a moving target. Finally, Section 2.3 concludes this chapter.

2.1 Single Mobile Sensor Node and Dynamic Target Tracking

2.1.1 Problem formulation

We consider a mobile sensor tracking a target which moves in a two dimensional environment. The dynamic equation of the mobile sensor is described as follows:

$$\begin{cases} \dot{q}_r = p_r \\ \dot{p}_r = u_r. \end{cases} \tag{2.1}$$

Figure 2.1 shows a mobile sensor tracking a moving target with notations defined as

follows.

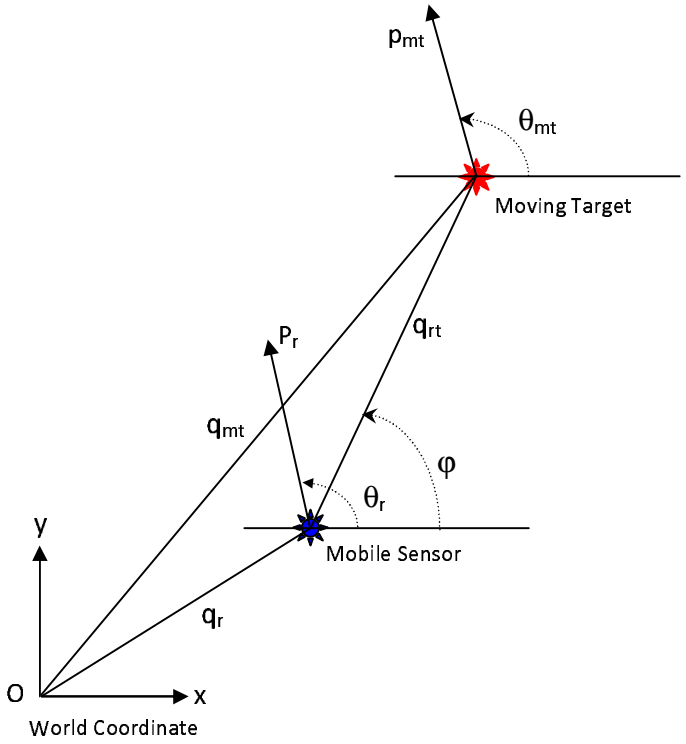


Figure 2.1: A mobile sensor tracks a moving target.

$q_r \in R^2, p_r \in R^2, \theta_r \in R^1$ are position, velocity, and heading of the mobile sensor at time t, respectively. $q_{mt} \in R^2, p_{mt} \in R^2, \theta_{mt} \in R^1$ are position, velocity, and heading of the moving target at time t, respectively. q_{rt}, φ are the relative position from the mobile sensor to the moving target and the angle of q_{rt}, respectively.

Assumption 1. We have the following assumption: The mobile sensor is equipped with sensors such as cameras, sonars or laser sensors and the associated algorithms to estimate the trajectory (position and velocity) of the moving target.

Let $q_{rt} = [x_{rt},\ y_{rt}]^T$ be the relative position between a mobile sensor and a moving target, then the relative velocity between them can be expressed as the derivative of relative position q_{rt}. Hence the relative velocity vector is $p_{rt} = \dot{q_{rt}} = [\dot{x}_{rt},\ \dot{y}_{rt}]^T$, where $\dot{x}_{rt}$ and $\dot{y}_{rt}$

are computed as follows:

$$\begin{cases} \dot{x}_{rt} = \|p_{mt}\| cos(\theta_{mt}) - \|p_r\| cos(\theta_r) \\ \dot{y}_{rt} = \|p_{mt}\| sin(\theta_{mt}) - \|p_r\| sin(\theta_r), \end{cases} \tag{2.2}$$

where $\|.\|$ is the Euclidean distance.

The tracking task is to make $\|q_{rt}\|$ approach to zero as soon as possible. This means that $q_r = q_{mt}$ and $p_r = p_{mt}$.

2.1.2 Potential field approach

To solve the problem of moving target tracking, we use the potential field approach which consists of an attractive potential function defined as follows [107, 108, 109]:

$$V_a = 0.5\lambda_1 q_{rt}^T q_{rt}, \tag{2.3}$$

here λ_1 is a positive scale factor for the attractive potential field function.

In target tracking, we want the mobile sensor to follow a target. Hence, we only need the attractive potential field for the total potential field of q_{rt}.

$$V = V_a = 0.5\lambda_1 q_{rt}^T q_{rt}. \tag{2.4}$$

The velocity p_r of the mobile sensor is computed as

$$p_r = \dot{q}_r = \nabla_{q_{rt}} V = \lambda_1 q_{rt}. \tag{2.5}$$

Equation (2.5) is with respect to the stationary target ($p_{mt} = 0$) (conventional potential field method). While for a moving target ($p_{mt} \neq 0$) we compute the velocity p_r of the mobile sensor as follows [107]:

$$p_r = p_{mt} + \lambda_1 q_{rt}. \tag{2.6}$$

Equation (2.6) is equivalent to the following equations [107]:

$$\|p_r\| sin(\theta_r - \varphi) = \|p_{mt}\| sin(\theta_{mt} - \varphi), \tag{2.7}$$

$$\|p_r\| = (\|p_{mt}\|^2 + 2\lambda_1\|q_{rt}\|\|p_{mt}\|cos(\theta_{mt} - \varphi) + \lambda_1^2\|q_{rt}\|^2)^{1/2}. \quad (2.8)$$

Assumption 2. The velocity of the moving target is limited by its maximum velocity p_{mt}^{max}.

From this assumption we have:

$$\begin{aligned} \|p_r\| &= min(\|p_{mt}^{max}\|, (\|p_{mt}\|^2 + 2\lambda_1\|q_{rt}\|\|p_{mt}\| \\ &\quad \times cos(\theta_{mt} - \varphi) + \lambda_1^2\|q_{rt}\|^2)^{1/2}). \end{aligned} \quad (2.9)$$

By dividing both sides of Equation (2.7) with $\|p_r\|$ and taking *arcsin* we obtain the heading or direction of the mobile sensor as

$$\theta_r = \varphi + arcsin(\frac{\|p_{mt}\|sin(\theta_{mt} - \varphi)}{\|p_r\|}). \quad (2.10)$$

The velocity of the mobile sensor in a two dimensional space is obtained as Equation (2.11).

$$p_r = [\|p_r\|cos(\theta_r),\ \|p_r\|sin(\theta_r)]^T. \quad (2.11)$$

Theorem 1. Equation (2.6) allows the mobile sensor (q_r, p_r) to track a moving target (q_{mt}, p_{mt}).

Proof:

We choose a Lyapunov function as follows:

$$L = V_a = 0.5\lambda_1 q_{rt}^T q_{rt} = 0.5\lambda_1\|q_{rt}\|^2. \quad (2.12)$$

This function is positive definite, and the derivative of L is given by

$$\dot{L} = \frac{\partial L}{\partial q_{rt}}\dot{q}_{rt} = \frac{\partial L}{\partial q_{rt}}p_{rt}, \quad (2.13)$$

where the relative velocity between the mobile sensor and the moving target is designed as $p_{rt} = -\nabla_{q_{rt}}V_a = -\nabla_{q_{rt}}L$. Hence, Equation (2.13) is rewritten as follows:

$$\dot{L} = -\frac{\partial L}{\partial q_{rt}}\nabla_{q_{rt}}L = -\lambda_1^2\|q_{rt}\|^2 = -2\lambda_1\frac{1}{2}\lambda_1\|q_{rt}\|^2 = -2\lambda_1 V_a < 0. \quad (2.14)$$

Since the Lyapunov function L is considered the same as the attractive potential field function V_a, Equation (2.14) is rewritten as follows:

$$\dot{V}_a = -2\lambda_1 V_a. \tag{2.15}$$

Solving this equation we get the solution as follows:

$$V_a = V_a(0)e^{-2\lambda_1 t} \tag{2.16}$$

here $V_a(0)$ is the value of V_a at $t = 0$. This solution shows that V_a and $\|q_{rt}\|$ converge to zero with the converging rate λ_1, or the position and velocity of the mobile sensor asymptotically converges to those of the moving target after a certain time ($t > 0$).

2.1.3 Simulation results

In this section we test our theoretical results with a circular trajectory of the moving target. Parameters used in this simulation are specified as follows:

Parameters of circle trajectory: $q_{mt} = [210 - 70cos(t),\ 80 + 70sin(t)]^T$.

Parameters of moving target: $p_{mt} = [3,\ 3]^T$, $p_{mt}^{max} = [55,\ 55]^T$, and $\theta_{mt} = \frac{\pi}{2} - t$.

Initial parameters of the mobile sensor: $q_r(0) = [0,\ 0]^T$, $p_r(0) = [0,\ 0]^T$, and $\theta_r(0) = \frac{\pi}{2}$, and other parameters: $\lambda_1 = 9$, $0 \leq t \leq 5$.

Figure 2.2 represents the result of one mobile sensor tracking the target moving in a circular trajectory. At the beginning, the position of the sensor is far from the target, but after certain time the sensor can catch up the moving target and then continue to track the target. This confirms the theory stated in Theorem 1. Figure 2.3 shows the tracking performance (position error between the mobile sensor and the moving target). As can be seen in this figure, after 33 iterations the mobile sensor tracks a moving target very well with very small error.

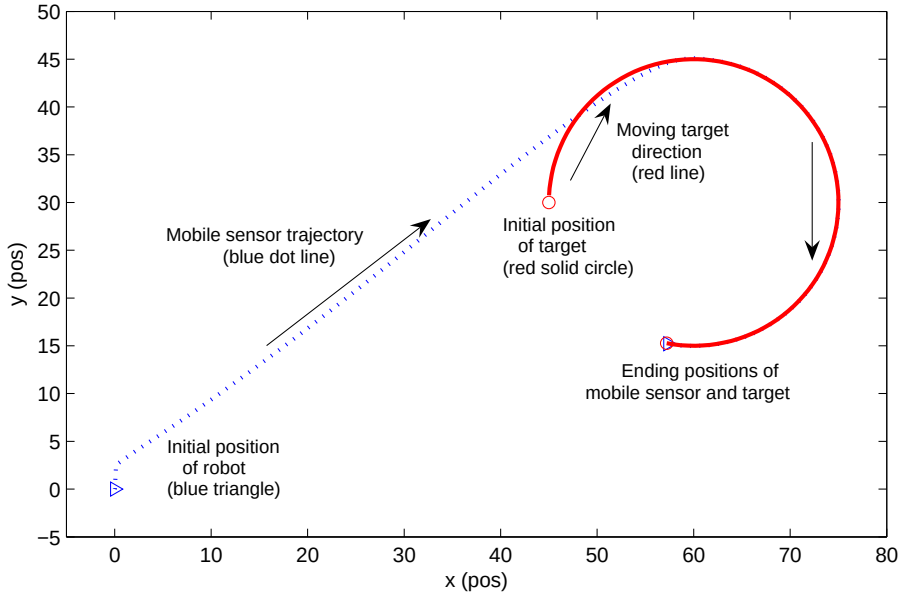


Figure 2.2: The mobile sensor tracks the target moving in a circular trajectory.

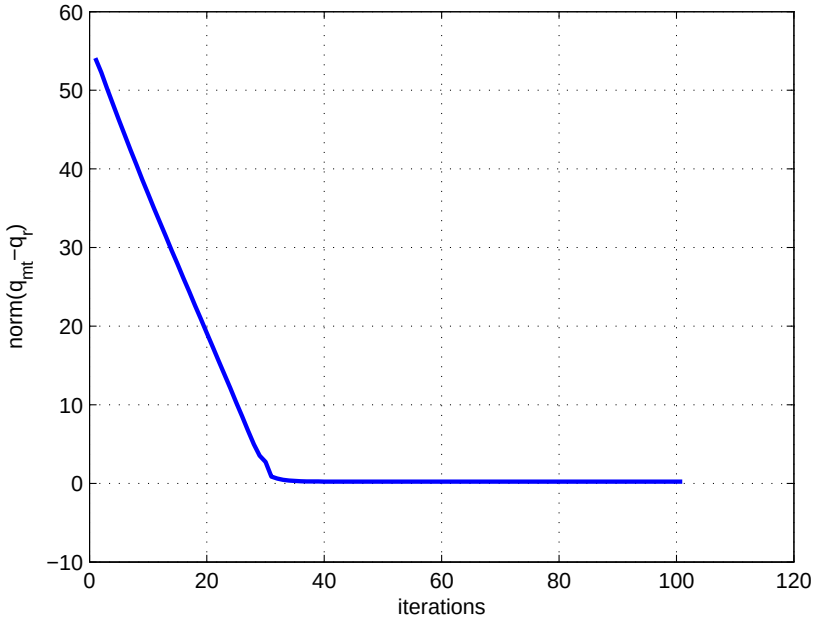


Figure 2.3: Tracking error between the mobile sensor and the moving target.

2.2 Flocking Control for Single Target Tracking and Observing

In this section we will extend the potential field approach to multiple mobile sensor nodes (agents) based on flocking control. The artificial potential field is created to generate a pairwise attractive/repulsive force to control agents to form a lattice formation while tracking the target. However, with this type of traditional flocking control [23], there are still some problems in cluttered environments where the agents usually get stuck behind the obstacles and sometimes can not follow the target [23]. To handle this problem we present new approaches to flocking control of multi-agent systems to track a moving target while avoiding obstacles. The main motivation of these approaches is to make the CoM (Center of Mass) of the network track the moving target better in cluttered environments where the traditional flocking control algorithms [23], [17], [34], [35], [46] have poor tracking performance. In our methods all mobile agents can surround the target closely in the obstacle space. This will allow the network to observe and recognize the target more accurately. Specifically, in our *Single-CoM* algorithm, the center of mass of positions and velocities of all mobile agents in the network is controlled to track a moving target. This algorithm works well in small networks, but it has limited scalability in large networks. In contrast with the *Single-CoM* algorithm, we proposed another flocking control algorithm called *Multi-CoM* where the center of mass of positions and velocities of each agent's local neighborhood, respectively is controlled to track a moving target. This algorithm allows agents to perform better in large networks in a distributed fashion.

2.2.1 Flocking control background

To describe a dynamic topology of flocks or swarms we consider a dynamic graph $G(\vartheta, E)$ consisting of a vertex set $\vartheta = \{1, 2..., n\}$ and an edge set $E \subseteq \{(i, j) : i, j \in \vartheta, i \neq j\}$. In this topology each vertex denotes one member of the flock, and each edge denotes the communication link between two members.

Let $q_i, p_i \in R^m$ ($m = 2,3$) be the position and velocity of node i, respectively. We know that during the motion of sensors, the relative distance between them may change, hence the neighbors of each sensor also change. Therefore, we can define a set of neighborhood of sensor i at time t as follows:

$$N_i(t) = \left\{ j \in \vartheta : \|q_j - q_i\| \leq r,\ \vartheta = \{1,2,...,n\}, i \neq j \right\} \tag{2.17}$$

Here, r is an interaction range (radius of the neighborhood circle in the case of two dimensions, $m = 2$, or radius of the neighborhood sphere in the case of three dimensions, $m = 3$), and $\|.\|$ is the Euclidean distance.

In this chapter we consider n sensors moving in an m dimensional Euclidean space. We address the motion control problem for a group of sensors with the objective of dynamic target(s) tracking. We assume that each sensor has a large enough communication range to allow it to communicate with others and a large enough sensing range to allow it to sense the target. We also assume that each sensor is equipped with sonar or laser sensor that allows it to estimate the position and velocity of the target.

The dynamic equation of each sensor is described as follows:

$$\begin{cases} \dot{q}_i = p_i \\ \dot{p}_i = u_i, \quad i = 1,2,...,n \end{cases} \tag{2.18}$$

The geometry of a flock is modeled by an α-lattice [23] that has the following condition:

$$\|q_i - q_j\| = d, j \in N_i \tag{2.19}$$

here d is a positive constant indicating the distance between sensor i and its neighbor j.

The configuration which approximately satisfies the condition (2.19) is called a quasi α-lattice, i.e. $(\|q_i - q_j\| - d)^2 < \delta^2$, with $\delta << d$.

To construct a collective potential (discuss later) that is differentiable at singular configuration ($q_i = q_j$), the set of algebraic constrains is rewritten in term of σ - norm (defined in (2.24)) as follows:

$$\|q_j - q_i\|_\sigma = d^\alpha, j \in N_i \tag{2.20}$$

In [23], Olfati-Saber proposed his control law for flocking of multiple mobile agents with obstacle avoidance. This algorithm consists of three components as follows:

$$u_i = f_i^{\alpha} + f_i^{\beta} + f_i^{\gamma} \tag{2.21}$$

The first component of Equation (2.21) f_i^{α} which consists of a gradient-based component and a consensus component (more details about these components see [73], [50], [51]) is used to regulate the gradient of potentials (impulsive or attractive forces) and the velocity among sensors.

$$f_i^{\alpha} = c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \tag{2.22}$$

where each term in (2.22) is computed as follows [23]:

The set of α neighbors at time t is

$$N_i^{\alpha}(t) = \left\{ j \in \vartheta : \|q_j - q_i\| \leq r,\ \vartheta = \{1, 2, ..., n\},\ i \neq j \right\} \tag{2.23}$$

The $\sigma - norm$, $\|.\|_{\sigma}$, of a vector is a map $R^m \Longrightarrow R_+$ defined as

$$\|z\|_{\sigma} = \frac{1}{\varepsilon}[\sqrt{1 + \varepsilon \|z\|^2} - 1] \tag{2.24}$$

here ε is the positive constant.

The action function $\phi_{\alpha}(z)$ vanishing for all $z \geq r_{\alpha}$ with $r_{\alpha} = \|r\|_{\sigma}$ is used to construct a smooth pairwise attractive/repulsive potential function, $\psi_{\alpha}(z) = \int_{d_{\alpha}}^{z} \phi_{\alpha}(s) ds$. This action function $\phi_{\alpha}(z)$ is defined as follows:

$$\phi_{\alpha}(z) = \rho_h(z / r_{\alpha}) \phi(z - d_{\alpha}) \tag{2.25}$$

where $\phi(z)$ is the uneven sigmoidal function

$$\phi(z) = 0.5[(a+b)\sigma_1(z+c) + (a-b)] \tag{2.26}$$

here $\sigma_1(z) = z / \sqrt{1 + z^2}$, and parameters $0 < a \leq b$, $c = |a-b| / \sqrt{4ab}$ to guarantee $\phi(0) = 0$, and constraints $d_{\alpha} = \|d\|_{\sigma}$ with $d = r/k$ for k being the scaling factor (in the simulations in this dissertation $k = 1.2$).

The bump function $\rho_h(z)$ with $h \in (0,1)$

$$\rho_h(z) = \begin{cases} 1, & z \in [0,h) \\ 0.5[1+cos(\pi(\frac{z-h}{1-h}))], & z \in [h,1] \\ 0, & otherwise. \end{cases} \tag{2.27}$$

The vector along the line connecting q_i and q_j is defined as

$$n_{ij} = (q_j - q_i)/\sqrt{1+\varepsilon\|q_j - q_i\|^2} \tag{2.28}$$

The adjacency matrix $a_{ij}(q)$ is defined as

$$a_{ij}(q) = \begin{cases} \rho_h(\|q_j - q_i\|_\sigma / r_\alpha), & if \quad j \neq i \\ 0, & if \quad j = i \end{cases} \tag{2.29}$$

The second component of Equation (2.21) f_i^β is used to control the mobile sensors to avoid obstacles,

$$f_i^\beta = c_1^\beta \sum_{k \in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma)\hat{n}_{i,k} + c_2^\beta \sum_{k \in N_i^\beta} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \tag{2.30}$$

where the set of β neighbors (virtual neighbors) of sensor i at time t with k obstacles is

$$N_i^\beta(t) = \left\{ k \in \vartheta_\beta : \|\hat{q}_{i,k} - q_i\| \leq r', \vartheta_\beta = \{1,2,...,k\} \right\} \tag{2.31}$$

here r' is selected to be less than r, in our simulations $r' = 0.6r$. ϑ_β is a set of obstacles. $\hat{q}_{i,k}, \hat{p}_{i,k}$ are the position and velocity of sensor i projecting on the obstacle k, respectively.

Similar to vector n_{ij} defined in Equation (2.28), vector $\hat{n}_{i,k}$ is defined as

$$\hat{n}_{i,k} = (\hat{q}_{i,k} - q_i)/\sqrt{1+\varepsilon\|\hat{q}_{i,k} - q_i\|^2}. \tag{2.32}$$

The adjacency matrix $b_{i,k}(q)$ is defined as

$$b_{i,k}(q) = \rho_h(|\hat{q}_{i,k} - q_i\|_\sigma / d_\beta) \tag{2.33}$$

where $d_\beta = \|r'\|_\sigma$.

The repulsive action function of β neighbors is defined as

$$\phi_\beta(z) = \rho_h(z/d_\beta)(\sigma_1(z-d_\beta)-1). \tag{2.34}$$

Now we want to show more details on how to find out β neighbors $(\hat{q}_{i,k},\ \hat{p}_{i,k})$ generated by each α agent. Firstly, we have the following assumption regarding the obstacles.

Assumption 3. Obstacles are the convex regions in R^m with boundaries being smooth manifolds.

Based on this assumption, we can choose obstacles to be circles (two dimensions, m = 2) or spheres (three dimensions, m = 3) with radius R_k at center y_k. We project each sensor to obstacles and find out which shadow of that sensor on obstacles satisfies the condition $\|\hat{q}_{i,k}-q_i\| \leq r^{'}$, and the obtained results of $\hat{q}_{i,k}$ are neighbors of sensor i. Equation (2.35) illustrates the projection method to find the positions and velocities of β neighbors generated by sensor i.

$$\hat{q}_{i,k} = \mu q_i + (1-\mu)y_k,\ \hat{p}_{i,k} = \mu P p_i \tag{2.35}$$

where $\mu = R_k/\|q_i-y_k\|$. $P = I - a_k a_k^T$ is the projection matrix with $a_k = (q_i - y_k/\|q_i-y_k\|)$ and an unit matrix or identity matrix I.

Example 1. In this case, we have three obstacles O_1, O_2 and O_3 as shown in Figure 2.4. After projecting α-sensor i on all obstacles, we see that only two shadows (β-neighbors) on the obstacles O_1 and O_2 satisfying the condition (2.23). The obstacle O_3 is out of active range $r^{'}$, hence there is no shadow of α-sensor i on it. Consequently, we found out two β-neighbors $(\hat{q}_{i,1},\ \hat{p}_{i,1})$ and $(\hat{q}_{i,2},\ \hat{p}_{i,2})$ of α-sensor i.

The third component of (2.21) f_i^γ is a distributed navigational feedback.

$$f_i^\gamma = -c_1^\gamma \sigma_1(q_i-q_\gamma) - c_2^\gamma(p_i-p_\gamma) \tag{2.36}$$

where $\sigma_1(q_i-q_\gamma) = (q_i-q_\gamma)/\sqrt{1+\|q_i-q_\gamma\|^2}$, and the γ - sensor (q_γ, p_γ) is the virtual leader (more information of virtual leader, see [110]) defined as follows

$$\begin{cases} \dot{q}_\gamma = p_\gamma \\ \dot{p}_\gamma = f_\gamma(q_\gamma, p_\gamma) \end{cases} \tag{2.37}$$

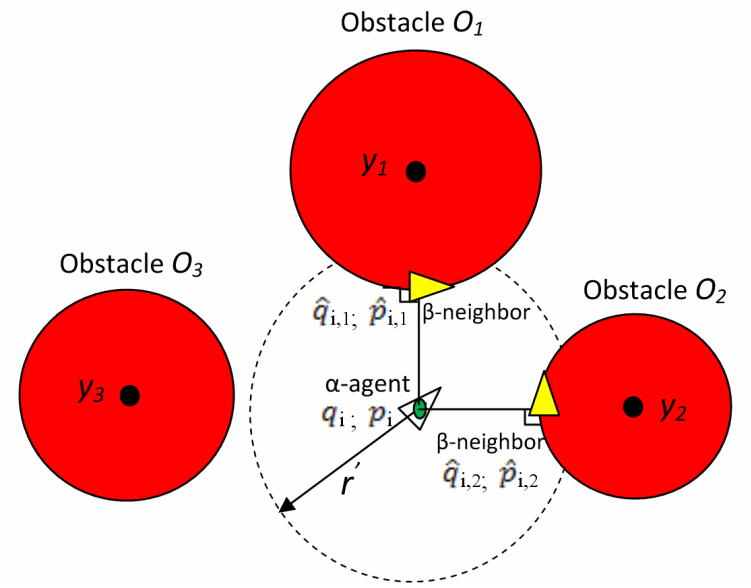


Figure 2.4: The projection method for finding the positions and velocities of β- neighbors of each α - sensor.

The constants of three components used in (2.21) are chosen as $c_1^\alpha < c_1^\gamma < c_1^\beta$, and $c_2^\nu = 2\sqrt{c_1^\nu}$. Here c_η^ν are positive constants for $\forall \eta = 1,2$ and $\nu = \alpha, \beta, \gamma$.

2.2.2 Algorithm description

In this section, we will extend the above described flocking algorithm with obstacle avoidance [23]. Two problems, named *Single-CoM* and *Multi-CoM*, respectively, will be investigated. In the *Single-CoM* problem, the CoM of positions and velocities of all sensors is controlled to track the moving target. In this case, each sensor need to know the positions and velocity of all other sensors, or it requires the global knowledge of the whole network. To address the scalability problem the *Multi-CoM* (CoM of positions and velocities of each sensor's local neighborhood, respectively) problem is studied, where each sensor only need to know the positions and velocity of its neighbors.

In the following algorithms we assume if one of the sensors in the network can estimate the position and velocity of the target, it will broadcast this obtained information to all other nodes. Consequently all sensors in the network can get the knowledge of target.

Single-CoM tracking

Firstly, based on Olfati-Saber's flocking algorithm we design an algorithm with a dynamic γ-agent. In this scenario, the dynamic γ-agent is considered as the moving target.

$$\begin{aligned} u_i &= c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ &\quad + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ &\quad - c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) \end{aligned} \tag{2.38}$$

here the pair $(q_{mt},\ p_{mt})$ is the position and velocity of the moving target, respectively, and c_1^{mt}, c_2^{mt} are positive constants, and $c_2^{mt} = 2\sqrt{c_1^{mt}}$.

By observing the control protocol (2.38), we see that the CoM is difficult to reach the target in the presence of obstacles. This creates the difficulty for sensors to track and observe the target. Therefore this protocol should be extended with more constraint on the CoM as follows:

$$u_i = f_i^{\alpha} + f_i^{\beta} + f^{mt} \tag{2.39}$$

where f^{mt} is a tracking feedback applied to sensor i by a moving target with position and velocity $(q_{mt},\ p_{mt})$, respectively.

$$\begin{aligned} f_i^{mt} &= -c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) \\ &\quad - c_1^{mt}(\overline{q} - q_{mt}) - c_2^{mt}(\overline{p} - p_{mt}) \end{aligned} \tag{2.40}$$

where the pair $(\overline{q}, \overline{p})$ is the center of mass (CoM) of positions and velocities of all sensors, respectively, as defined in (2.41).

$$\begin{cases} \overline{q} = \frac{1}{n} \sum_{i=1}^{n} q_i \\ \overline{p} = \frac{1}{n} \sum_{i=1}^{n} p_i. \end{cases} \tag{2.41}$$

The *Single-CoM* tracking is illustrated in Figure 2.5 (a). The CoM of the whole network (red dot) is created to track the target.

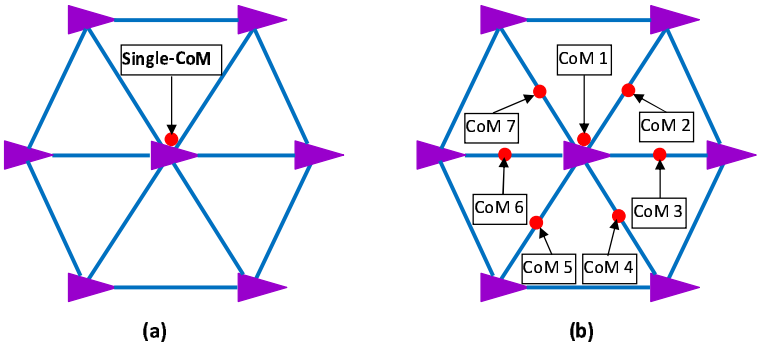


Figure 2.5: (a) A mobile sensor network with a single CoM (Single-CoM), (b) A mobile sensor network with multiple CoMs (Multi-CoM).

Consequently, the extended control protocol (2.39) is explicitly specified as follows:

$$\begin{aligned} u_i &= c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ &\quad + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ &\quad - c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) \\ &\quad - c_1^{sc}(\overline{q} - q_{mt}) - c_2^{sc}(\overline{p} - p_{mt}) \end{aligned} \tag{2.42}$$

here c_1^{sc}, c_2^{sc} are positive constants.

In control protocol (2.42), each mobile sensor at each time t need to know the position and velocity of all other sensors for computing the CoM $(\overline{q}, \overline{p})$. This means that this protocol is limited by the number of sensors, or the scalability is limited because at each time t all other sensors have to send their positions to sensor i. Hence the communication problem is a big challenge and need to be considered when implementing this protocol in real sensor networks.

Multi-CoM tracking

To make the algorithm scalable we implement a distributed flocking algorithm called *Multi-CoM* tracking in which the CoM of each sensor's local neighborhood is controlled to track the target. The *Multi-CoM* tracking is illustrated in Figure 2.5 (b). In this figure each mobile

sensor creates its own CoM, as a result multiple CoMs are created as a virtual network to track a taeget. The *Multi-CoM* tracking algorithm is presented as follows.

$$\begin{aligned} u_i &= c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ &\quad + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ &\quad - c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) \\ &\quad - c_1^{mc}(\overline{q}_{(N_i^{\alpha} \cup \{i\})} - q_t) - c_2^{mc}(\overline{p}_{(N_i^{\alpha} \cup \{i\})} - p_t), \end{aligned} \tag{2.43}$$

here $(c_1^{mc},\ c_2^{mc})$ are the positive constants, and the pair $(\overline{q}_{(i+N_i^{\alpha})},\ \overline{p}_{(i+N_i^{\alpha})})$ is defined as (2.44).

$$\begin{cases} \overline{q}_{(N_i^{\alpha} \cup \{i\})} = \frac{1}{|N_i^{\alpha} \cup \{i\}|} \sum_{i=1}^{|N_i^{\alpha} \cup \{i\}|} q_i \\ \overline{p}_{(N_i^{\alpha} \cup \{i\})} = \frac{1}{|N_i^{\alpha} \cup \{i\}|} \sum_{i=1}^{|N_i^{\alpha} \cup \{i\}|} p_i, \end{cases} \tag{2.44}$$

here $|N_i^{\alpha} \cup \{i\}|$ is the number of agents in agent i's local neighborhood including agent i itself.

In control protocol (2.43), each mobile sensor only need local knowledge, or it means that each sensor only requires the position and velocity knowledge of itself and its neighbors. In α-lattice configuration [23] the maximum number of each sensor's neighbors is 6. Therefore this protocol can scale up to lager mobile sensor networks.

2.2.3 Stability analysis

In this sub-section we will analyze the stability of our algorithms, flocking control with *Single-CoM* and *Multi-CoM*, respectively, in free space, and we will explain why the tracking performance in the presence of CoM constraint is better than without CoM constraint in obstacle space.

Theorem 2. In free space, by controlling the CoM based on the control protocol (2.42), the CoM of positions and velocities of all sensors in the network will exponentially converge to the target. In addition, the formation of all mobile sensors will maintain in the process of the moving target tracking.

Proof:

In free space, this means that $\sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)=0$. Hence we can rewrite control protocol (2.42) with ignoring constants c_η^ν (for $\forall\eta=1,2$ and $\nu=\alpha,\beta$) as follows:

$$\begin{aligned} u_i &= -\sum_{j\in N_i^\alpha}\nabla_{q_i}\psi_\alpha(\|q_j-q_i\|_\sigma)+\sum_{j\in N_i^\alpha}a_{ij}(q)(p_j-p_i) \\ &\quad -c_1^{mt}(q_i-q_{mt})-c_2^{mt}(p_i-p_{mt}). \\ &\quad -c_1^{sc}(\overline{q}-q_{mt})-c_2^{sc}(\overline{p}-p_{mt}). \end{aligned} \tag{2.45}$$

where $\psi_\alpha(z)=\int_{d_\alpha}^{z}\phi_\alpha(s)ds$ is the pairwise attractive/repulsive potential function. From (2.45), we can compute the average of the control law u as follows:

$$\begin{aligned} \overline{u}=\frac{1}{n}\sum_{i=1}^{n}u_i &= \frac{1}{n}\sum_{i=1}^{n}(-\sum_{j\in N_i^\alpha}\nabla_{q_i}\psi_\alpha(\|q_j-q_i\|_\sigma)+\sum_{j\in N_i^\alpha}a_{ij}(q)(p_j-p_i)) \\ &\quad -(c_1^{mt}+c_1^{sc})(\overline{q}-q_{mt})-(c_2^{mt}+c_2^{sc})(\overline{p}-p_{mt}). \end{aligned} \tag{2.46}$$

Obviously, we see that the pair $(\psi_\alpha,a(q))$ is symmetric. Hence we can rewrite (2.46) as:

$$\overline{u} = -(c_1^{mt}+c_1^{sc})(\overline{q}-q_{mt})-(c_2^{mt}+c_2^{sc})(\overline{p}-p_{mt}) \tag{2.47}$$

Equation (2.47) implies that

$$\begin{cases} \dot{\overline{q}}=\overline{p} \\ \dot{\overline{p}}=-(c_1^{mt}+c_1^{sc})(\overline{q}-q_{mt})-(c_2^{mt}+c_2^{sc})(\overline{p}-p_{mt}). \end{cases} \tag{2.48}$$

The solution of (2.48) indicates that the position and velocity of the CoM will exponentially converge to those of the target.

The formation or collision-free and velocity matching among mobile sensors will be maintained in the free space tracking because the gradient-based term and the consensus term are considered in this situation.

For the *Multi-CoM* flocking control algorithm, we have the following statement for the stability properties.

In cluttered environments, consider a system of n mobile agents, that have dynamics (2.18) and are controlled by the *Multi-CoM* flocking algorithm (2.43). Then based on our observations which are shown in the simulation results we see that:

1. The CoM of positions and velocities of all agents in the network will exponentially converge to the target in the free space.

2. The error between the CoM's position and the target's position is reduced in the obstacle space.

The results of the *Multi-CoM* flocking algorithm are similar to the *Single-CoM* flocking algorithm. However, the benefit of the *Multi-CoM* flocking algorithm is that each agent is controlled locally instead of globally as in the *Single-CoM* flocking algorithm.

2.2.4 Simulation results

In this section we test our theoretical results in simulation with different trajectories of the moving target. First of all we test the case where target moves with a sine wave trajectory. Parameters used in this simulation are specified as follows:

- Parameters of flocking: number of sensors = 120; the initial positions of sensors are randomly distributed in a box with a size of [0 90]x[0 90]; the initial velocities of sensors are set to zero. Parameters $a = b = 5$; the interaction range $r = 1.2d = 9$; $\varepsilon = 0.1$ for the σ-norm; h = 0.2 for the bump function ($\phi_\alpha(z)$); h = 0.9 for the bump function ($\phi_\beta(z)$).

- Parameters of target movement: The target moves in the sine wave trajectory: $q_{mt} = [50 + 35t,\ 295 - 35sin(t)]^T$ with $0 \leq t \leq 8.5$, and $p_{mt} = (q_{mt}(t) - q_{mt}(t-1))/\Delta_t$ with $\Delta_t = 0.002$.

Second we test the case where the target moves in a circle trajectory. Parameters used in this simulation are specified as follows:

- Parameters of flocking: parameters used in this case are the same with those in the sine trajectory case.

- Parameters of target movement: The target moves in a circle trajectory: $q_{mt} = [310 - 160cos(t),\ 255 + 160sin(t)]^T$ with $0 \leq t \leq 5$, and $p_{mt} = (q_{mt}(t) - q_{mt}(t-1))/\Delta_t$.

To compare three algorithms *No-CoM* (2.38), *Single-CoM* (2.42) and *Multi-CoM* (2.43) we use the same initial state (position and velocity) of mobile sensors. Figures 2.6 repre-

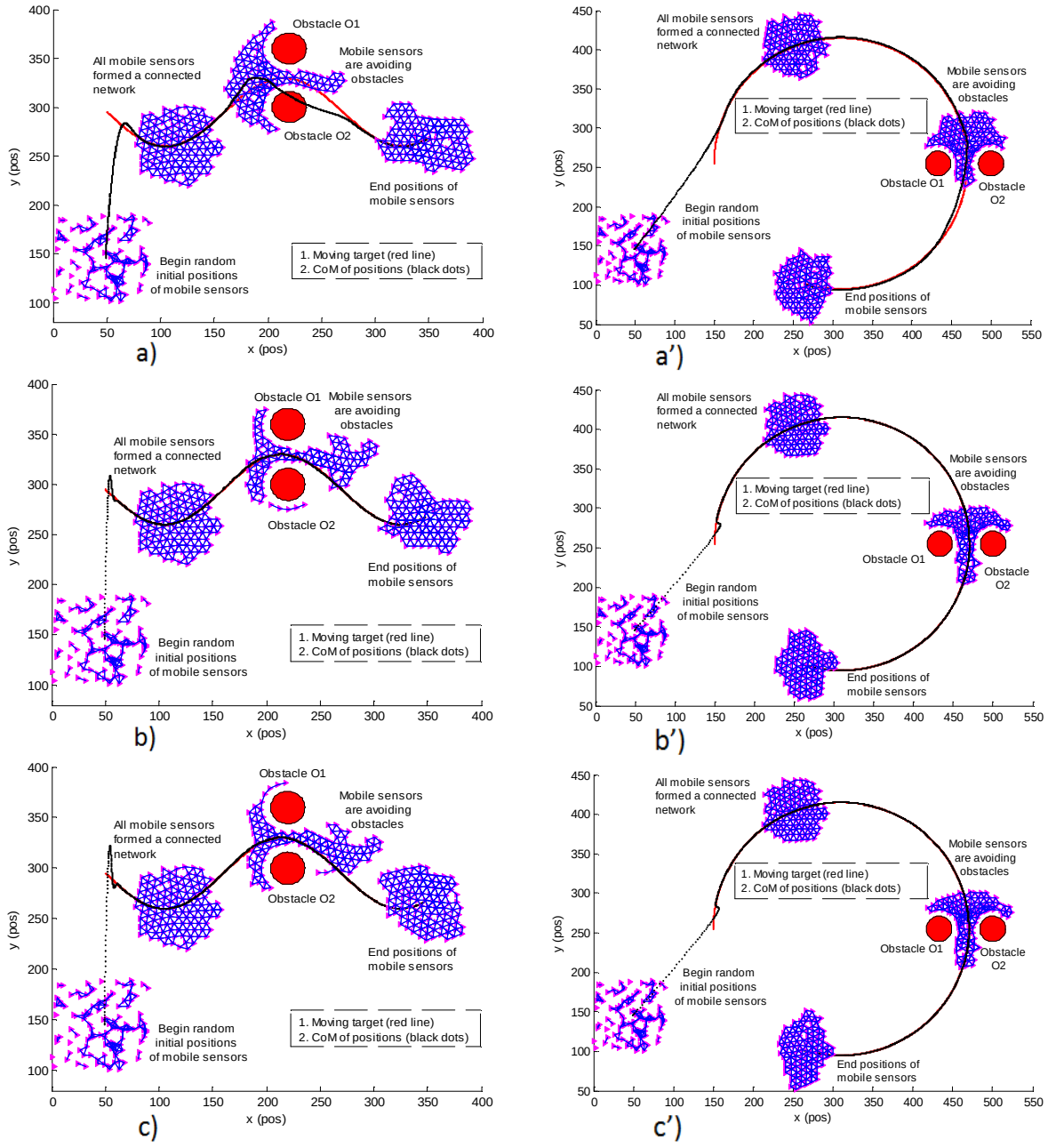


Figure 2.6: Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, forming a network, avoiding obstacles, and at the ending positions, respectively. (a, b, c) the mobile sensor network is tracking the target moving in the sine wave trajectory, and (a', b', c') the mobile sensor network is tracking the target moving in the circle trajectory using flocking control algorithms with *No-CoM* (2.38), *Single-CoM* (2.42) and *Multi-CoM* (2.43), respectively.

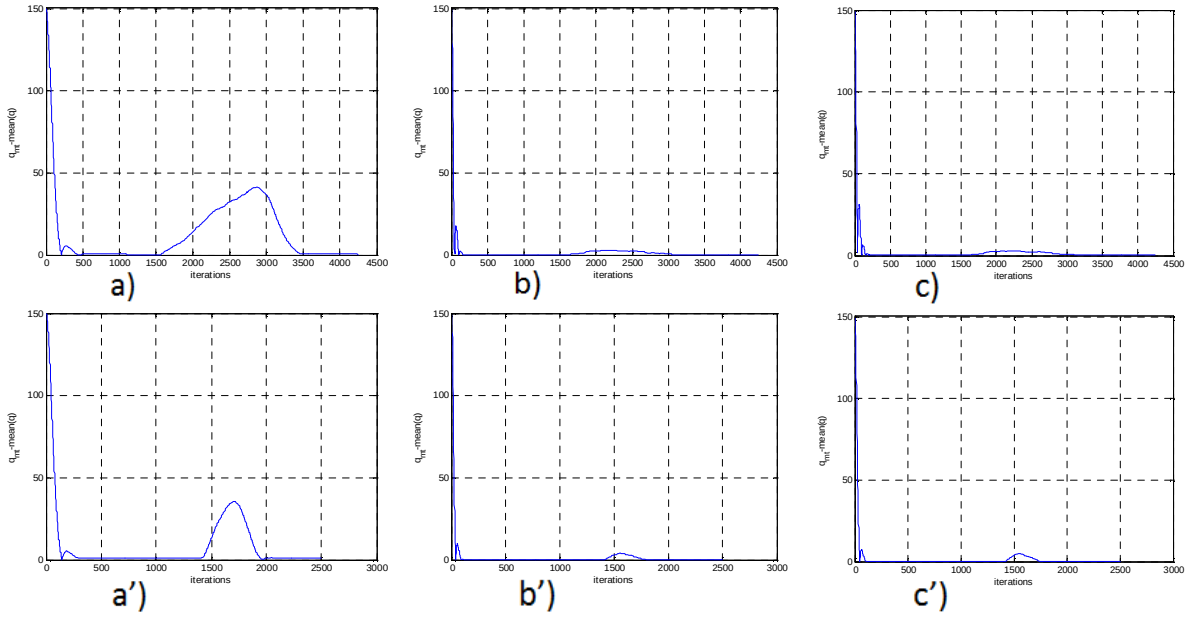


Figure 2.7: Position errors between the CoM's positions and the moving target in the sine wave trajectory (a, b, c) and the circle trajectory (a', b', c') using flocking control algorithms with *No-CoM* (2.38), *Single-CoM* (2.42) and *Multi-CoM* (2.43), respectively.

sents the snapshots of mobile agents tracking the target moving in the sine wave and circle trajectories using three algorithms, *No-CoM*, *Single-CoM* and *Multi-CoM*, respectively. Figures 2.7 represents the error between the CoM's positions and the target (tracking performance) in the sine wave and circle trajectories using three algorithms, *No-CoM*, *Single-CoM* and *Multi-CoM*, respectively. We see that the results of tracking performance in Figure 2.7 (b, b', c, c') for both trajectories of the target using *Single-CoM* and *Multi-CoM* algorithms, respectively, are better than that in Figure 2.7 (a, a') using *No-CoM* algorithm. In addition, we can see the snapshots of mobile robots avoiding obstacle taken at the same time, but in Figures 2.6 (b, b', c, c') more agents (sensors) passed through the narrow space between two obstacles than that in Figures 2.6 (a, a'). This means that the CoM in the algorithms *Single-CoM* and *Multi-CoM* (Figures 2.7 b, b', c, c') is closer to the target than that in the *No-CoM* algorithm (Figures 2.7 a, a').

2.3 Summary

This chapter first studied the problem of single moving target tracking using a mobile robot based on the artificial potential field approach. The simulation results were collected to show the effectiveness of the proposed approach. Then, this approach is extended to target tracking in mobile sensor networks based on flocking control. We designed a flocking control algorithm with *Single-CoM* and *Multi-CoM* to enable mobile sensors to track and observe the moving target more effectively while maintaining their formation and no collision among them. We prove that the CoM of positions and velocities of all mobile sensors exponentially converges to the target. By controlling the CoM explicitly, the mobile sensor network can track and observe the moving target better. This means that all mobile sensors in the network can surround the target closely which will allow them to see the target easily for recognition purpose. In addition, flocking control with *No-CoM*, flocking control with *Single-CoM*, and flocking control with *Multi-CoM* are compared. Several simulations are conducted with different target trajectories to demonstrate our theoretical results.

CHAPTER 3

COOPERATIVE CONTROL BASED FLOCKING FOR MSNs IN NOISE-FREE ENVIRONMENTS

In this chapter we study the cooperative control for MSNs in noise-free environments in which each mobile sensor node can sense the location and velocity of itself and its neighbors precisely. Three cooperative control algorithms are proposed. The first one is the flocking control algorithm for MSNs to track a target in the case of a small subset of informed agents while maintaining the network connectivity. The second one is the adaptive flocking control for MSNs to track a moving target in complex environments where the MSNs have to change the size of their formation to adapt to the environment in order to maintain the network connectivity and similar topology. The last one is the multiple dynamic target tracking algorithm which is designed for MSNs to track multi-target simultaneously.

This chapter is organized as follows. Section 3.1 presents the decentralized flocking control with a minority of informed agents. Section 3.2 presents the adaptive flocking control for MSNs to track a moving target. Section 3.3 describes multi-target tracking algorithm for MSNs. Finally, Section 3.4 concludes this chapter.

3.1 Decentralized Flocking Control with a Minority of Informed Agents

In this section we study the flocking control in the case of a small subset of informed agents. In nature, only few agents in the group have information of the target, such as knowledge about the location of a food source, or of a migration route, but they can still flock together in a group to find the source of food (target) based on local information.

Inspired by this natural phenomenon, a flocking control algorithm is designed to coordinate the motion of multiple agents. Based on our algorithm, all agents can form a network, maintain connectivity and track the target even only a few agents know the information of the target.

3.1.1 Introduction

Early work on flocking control includes [37, 38, 23]. Tanner *et al*. [37], [38] studied flocking control of a system of multiple mobile agents with double integrator dynamics in the case of fixed and dynamic topologies. In [23], the theoretical framework for design and analysis of distributed flocking algorithm was proposed. This established a foundation for flocking control design for a group of agents. As an extension of the flocking algorithm in [23], flocking control of agents with a virtual leader in the case of a minority of informed agents and varying velocity of virtual leader was presented in [46]. However, in their work the network can not maintain its connectivity because some agents may fall out of the network.

In this section we study how to utilize a minority of informed agents to lead the whole network to track the target while maintaining the connectivity. The main differences with the above related work are:

1. We adopt a target navigation term in order to reduce the large tracking force at the initial tracking time so that the connectivity is maintained.

2. We use a damping force term to reduce the tracking overshoot.

Overall, we propose a new flocking control algorithm that allows the flock to preserve connectivity, avoid collision, and track the target without overshooting. We demonstrate that by applying our algorithm the agents can flock together and maintain connectivity better compared to existing flocking control algorithms.

Most of existing flocking control algorithms [37, 38, 23] are designed under the assumption that all agents need information of the position and velocity of the target in order

to flock together. However, in reality this assumption is not valid. It can be seen in many cases that only very few agents have information of the target due to their limited sensing range. For example, in fish schools and bird flocks, only some agents have knowledge about the location of a food source, or of a migration route [41, 42]. Motivated by these observations we will study how to design a distributed flocking control algorithm which can still maintain good tracking performance and connectivity when only few agents have information of the target.

3.1.2 Decentralized Flocking Control with a Minority of Informed Agents (MIA)

In this subsection, we design a distributed flocking control algorithm for multi-agent systems in the case that only a few agents are informed with the position and velocity of the target. We call these agents as informed agents. Let us define N_I as a subset of informed agents and N_{UI} as a subset of uninformed agents with $N_I << N_{UI}$. Hence we have $N_I \cup N_{UI} = N$, here N is the set of all agents (uninformed and informed agents).

Paper [46] proposed the following flocking control algorithm based on the algorithm (2.38):

$$\begin{aligned} u_i &= \sum_{j\in N_i} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + \sum_{j\in N_i} a_{ij}(q)(p_j - p_i) \\ &\quad -c_1^t(q_i - q_t)I_i - c_2^t(p_i - p_t)I_i. \end{aligned} \tag{3.1}$$

here if $I_i = 1$ the agent i has information (position and velocity) of the target. Otherwise $I_i = 0$ agent i does not have information of the target.

We implemented the algorithm (3.1) in which we let some agents closest to the target have the information (position and velocity) of the target. The result is shown in Figure 3.1.

In this figure we clearly see that the network is broken, and only the agents which have information of the target can track the target. Additionally, we find that the target tracking performance has big overshoot. In order to solve these two problems we introduce two

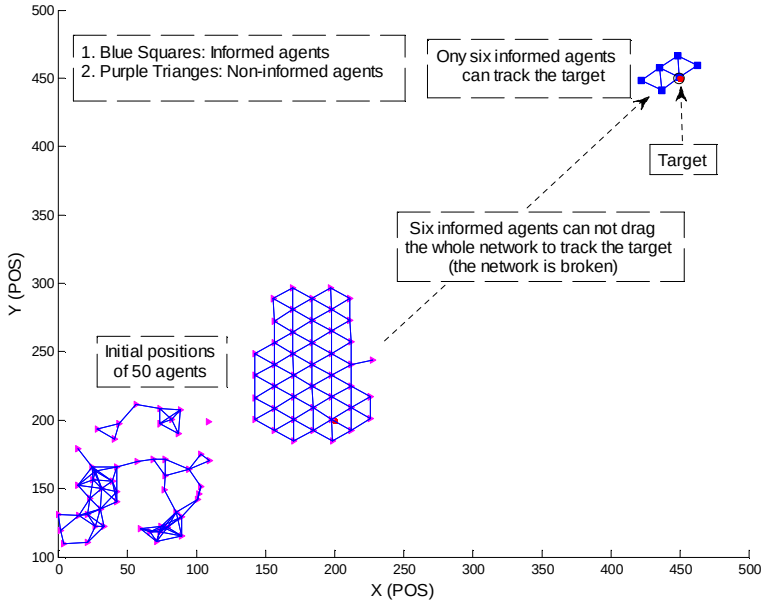


Figure 3.1: Snapshots of the agents when applying the flocking control algorithm (3.1). We select 6 out of 50 agents which are closest to the target to have the information (position and velocity) of the target.

terms. The first term is a navigation term, and the second one is a damping force term. The main purpose of the navigation term is to maintain the connectivity among agents, and the purpose of the damping force term is to reduce the tracking overshoot.

Navigation Term

The navigation term allows the agents to stay together. The main idea behind this term is that if we let the informed agents keep strong cohesion to uninformed agents at the initial time of the target tracking process, the connectivity can be maintained. In order to do this, we have to reduce the initial momentum of the attractive force to the target for the informed agents. This means that we should have small attractive force at the initial time when the distance between the informed agent and the target is large. Based on this analysis we design the navigation term as shown in Algorithm 1. In this algorithm the constant K_1 chosen between 0.9 and 1 is to ensure that a small attractive force is applied at the initial

time of the target tracking process. The weights, $\frac{K_2}{\|q_i^{inf}(t)-q_t(t)\|}$ and $\frac{K_3}{\|q_i^{inf}(t)-q_t(t)\|}$ are designed so that the attractive force is small enough at the initial time, and then it becomes bigger when the distance $\|q_i^{inf}(t)-q_t(t)\|$ decreases.

Algorithm 1: Design of the Navigation Term

for *each informed agent* j, $j \in N_I$ **do**

  if $\|q_i^{inf}(t)-q_t(t)\| > K_1\|q_i^{inf}(0)-q_t(0)\|$ **then**

$$f_j^t = -\frac{K_2}{\|q_i^{inf}(t)-q_t(t)\|}(q_j^{inf}-q_t) - \frac{K_3}{\|q_i^{inf}(t)-q_t(t)\|}(p_j^{inf}-p_t)$$

    here, $0.9 < K_1 < 1$, $K_2 > 0$ and $K_3 > 0$,

  else

$$f_i^t = -c_1^t(q_j^{inf}-q_t) - c_2^t(p_j^{inf}-p_t)$$

  end

end

Damping Force Term

Since only the informed agents N_I have the information of the target, the damping force can be only applied to these agents. The idea behind this damping force is to reduce the tracking overshoot when the informed agents are close to the target. That is, the damping force for the informed agents is only effective if the distance between the informed agent and the target is less than a certain threshold. This threshold is designed based on the active range r. This means that when the target is inside the active range of the informed agent j the damping force f_j^{dam} is applied, otherwise $f_j^{dam} = 0$. In order to do that the constant K_4 is used $(0 < K_4 < 1)$. When the damping force f_j^{dam} is applied, the informed agent j will reduce its speed gradually to approach the target. Hence, the tracking overshoot is reduced.

Algorithm 2: Design of the Damping Force Term

for *each informed agent* j, $j \in N_I$ **do**

  if $\|q_i^{inf}(t) - q_t(t)\| < K_4 r$ **then**

    $$f_j^{dam} = -K_{dam} p_j^{inf}$$

    here, $0 < K_4 < 1$ and $K_{dam} > 0$,

  else

    $$f_j^{dam} = 0$$

  end

end

Overall, the damping force is designed in Algorithm 2.

Finally the whole decentralized flocking control algorithm is proposed in Algorithm 3. In this algorithm we have two options of the initial network configuration, and both options are to allow the network of agents to be connected at the beginning.

Algorithm 3: Decentralized Flocking Control Algorithm with a MIA

Input : Position and velocity of each agent (q_i, p_i); Position and velocity of the target (q_t, p_t) for the informed agents (N_I).

Output: Control law for each agent u_i

Initialization phase: -**Option 1.** Deploy the agents to form a connected network; -**Option 2.** All agents are programmed based on flocking algorithm (2.38) to go to the rendezvous point so that they can form a connected network.

Implementation phase:

for *each agent i* **do**

Compute: $f_i^{\alpha} = \sum_{j\in N_i} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + \sum_{j\in N_i} a_{ij}(q)(p_j - p_i)$.

end

for *each informed agent j, j* $\in N_I$ **do**

if $\|q_i^{inf}(t) - q_t(t)\| > K_1 \|q_i^{inf}(0) - q_t(0)\|$ **then**

$$f_j^t = -\frac{K_2}{\|q_i^{inf}(t) - q_t(t)\|}(q_j^{inf} - q_t) - \frac{K_3}{\|q_i^{inf}(t) - q_t(t)\|}(p_j^{inf} - p_t),$$

else

$f_i^t = -c_1^t(q_j^{inf} - q_t) - c_2^t(p_j^{inf} - p_t)$.

end

if $\|q_i^{inf}(t) - q_t(t)\| < K_4 r$ **then**

$f_j^{dam} = -K_{dam} p_j^{inf}, (0 < K_4 < 1 \text{ and } K_{dam} > 0)$.

else

$f_j^{dam} = 0$.

end

end

for *each uninformed agent k, k* $\in N_{UI}$ **do**

$f_k^{dam} = 0,\ f_k^t = 0$.

end

Update the control law for each agent *i*: $u_i = f_i^{\alpha} + f_i^{dam} + f_i^t$.

3.1.3 Experimental and Simulation Results

In this section we test our proposed flocking control Algorithm 3 and compare it with the existing flocking control algorithm (3.1) in the case of a minority of informed agents. First we test our algorithm with 7 real robots. Then to show the effectiveness and the scalability of our algorithm we test it with 50 robots in simulation. In addition, we show a metric to evaluate the network connectivity of our algorithm and the existing algorithm.

Experimental Setup

In this experiment we use 7 Rovio robots [111] that have omni-directional motion capability. Basically, these robots can freely move in 6 directions. The dynamic model of the Rovio robot can be approximated by Equation (2.18). However, the accuracy of the localization of the Rovio robot is low, and the robot does not have any sensing device to sense the pose (position and velocity) of its neighbors or the obstacles. Hence we use a VICON motion capture system [1] in our lab (Figure 3.2) that includes 12 cameras to track objects. This tracking system can give the location and velocity of each moving object with over 95 percent of accuracy.

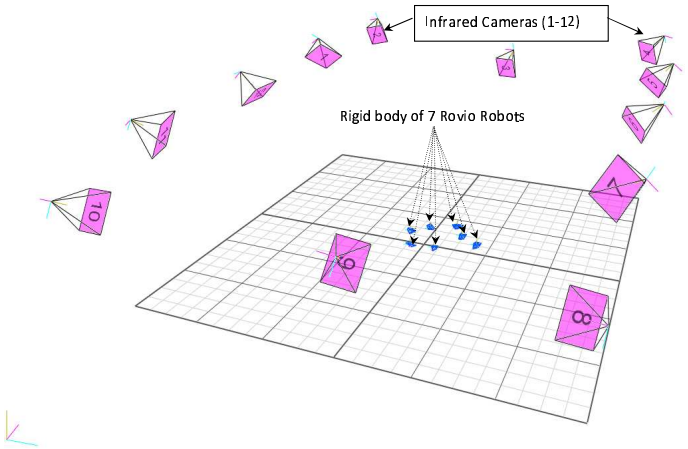


Figure 3.2: Motion Capture System from VICON [1] in the experimental setup.

We use the following parameters:

- Parameters of flocking: $a = b = 5$; $d = 600mm$; the scaling factor $k_c = 1.2$; the active range $r = k_c.d$; $\varepsilon = 0.1$ for the σ norm; h = 0.2 for the bump function ($\phi_\alpha(z)$); h = 0.9 for the other bump function ($\phi_\beta(z)$).

- Parameters of the target: The target location is at $[0,\ 500mm]$ for the experiment. The velocity vector $p_t = [5,\ 5]$.

Simulation Setup

In the simulation 50 agents are randomly distributed in the square area of 120×120 size, and we use the following parameters:

- Parameters of flocking: the constants $a = b = 5$ for the sigmoidal function ($\phi(z)$); the distance among agents $d = 16$ units; the scaling factor $k_c = 1.2$; the active range $r = k_c * d$; $\varepsilon = 0.1$ for the σ norm; h = 0.2 for the bump function ($\phi_\alpha(z)$); h = 0.9 for the other bump function ($\phi_\beta(z)$).

- Parameters of the target: The target location is at $[450,\ 450]$. The velocity vector $p_t = [5,\ 5]$.

Network Connectivity Evaluation

To evaluate the the network connectivity maintenance, first we know that the link (connectivity) between node i and node j is maintained if the distance $0 < \|q_i - q_j\| \leq r$. Otherwise this link is considered broken. For graph connectivity, a dynamic graph $G(\vartheta, E)$ is connected at time t if there exists a path between any two vertices. An example of graph connectivity is shown in Figure 3.3.

Based on the above analysis, to analyze the connectivity of the network we define a connectivity matrix $[c_{ij}(t)]$ as follows:

$$[c_{ij}(t)] = \begin{cases} 1, & if \quad j \in N_i(t), i \neq j \\ 0, & if \quad j \notin N_i(t), i \neq j \end{cases}$$

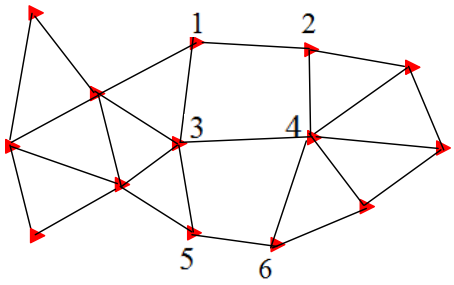


Figure 3.3: If one or two of the links (1,2), (3,4), (5,6) is broken the graph connectivity is still remained, but if all of that links is broken the graph connectivity is lost.

and $c_{ii} = 0$. Since the rank of Laplacian of a connected graph [23] $[c_{ij}(t)]$ of order n is at most $(n-1)$ or $rank([c_{ij}(t)]) \leq (n-1)$, the relative connectivity of a network at time t is defined as: $C(t) = \frac{1}{n-1} rank([c_{ij}(t)])$. If $0 \leq C(t) < 1$ the network is broken, and if $C(t) = 1$ the network is connected. Based on this metric we can evaluate the network connectivity in our proposed flocking control Algorithm 3 and the existing flocking control algorithm (3.1).

Experimental Results

Initially, the seven Rovio robots are randomly deployed so that they can form a connected network (see Option 1 in Algorithm 3). Then, two robots which are closest to the target are selected to be the informed agents (the two robots with cameras facing up as shown in snapshot (d) in Figure 3.5). We obtained the results of our flocking control Algorithm 3 in Figures 3.4, 3.5 and 3.6. Specially, Figure 3.5 (a, b, c) show the snapshots of simulation results for seven robots, and Figure 3.5 (d, e, f) show the snapshots of experiment results for seven robots. In Figure 3.6 we compare the trajectories of three out of seven robots in both simulation and experiment, and we see that the experimental trajectories have small difference with the ones in simulation since the motion of the robots is limited to only six directions. In addition, Figure 3.4 shows the connectivity result, and we clearly see that the seven robots can flock together even only two of them know the information of the target.

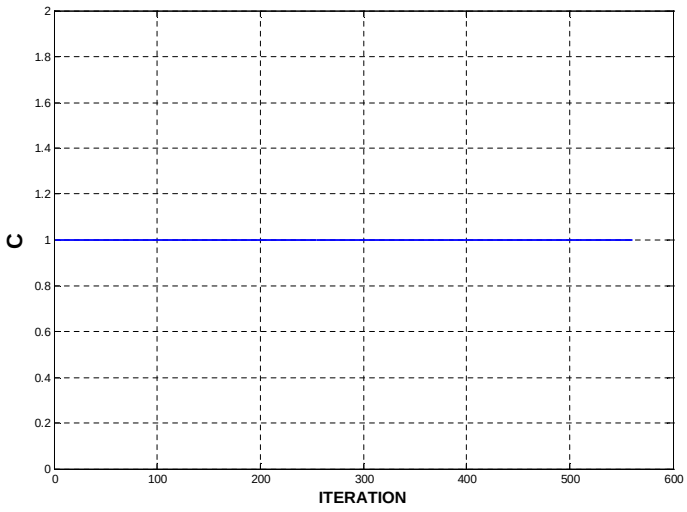


Figure 3.4: Connectivity evaluation in experiment of 7 Rovio robots when applying our proposed flocking control algorithm 3.

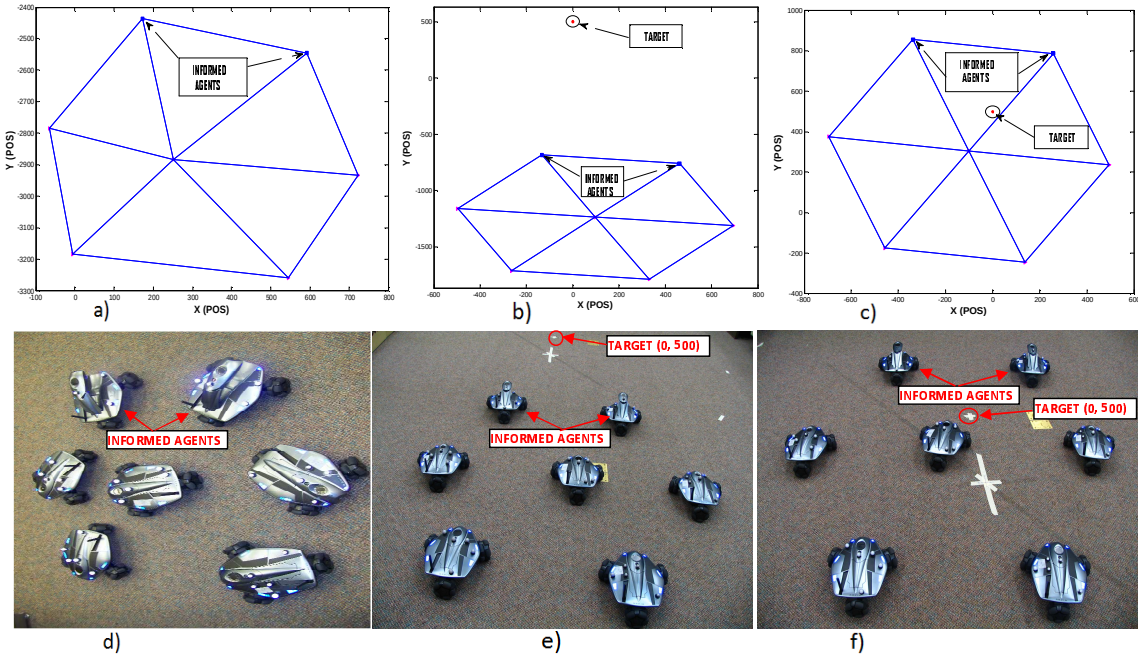


Figure 3.5: Snapshots of 7 Rovio robots flocking together when applying our proposed flocking control algorithm 3.

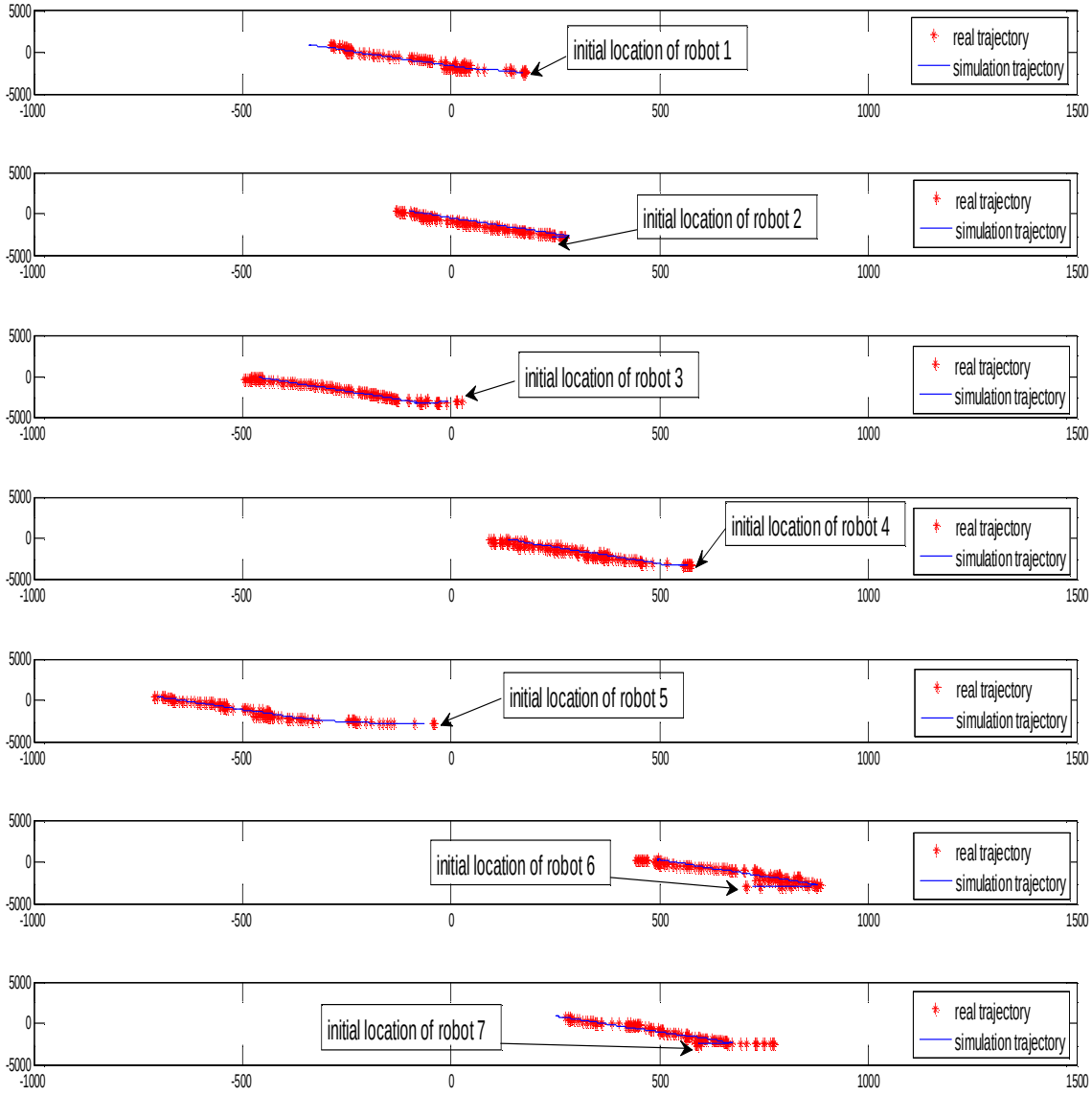


Figure 3.6: Trajectories of simulation and real robots when applying our proposed flocking control algorithm 3.

Simulation Results

In simulation, we test our proposed Algorithm 3 with fifty robots which are randomly deployed so that they do not form a connected network initially. Then, these robots are programed based on the flocking algorithm (2.38) to go to the rendezvous point (see Option 2 in Algorithm 3). This step is to make sure that the fifty robots form a connected network at the rendezvous point. After that we let two robots (blue squares) which are closest to the target know the position and velocity of the target. By observing Figure 3.7 we can see that the two informed robots can drag all 48 other robots (purple triangles) to flock together. The connectivity for the proposed Algorithm 3 and the algorithm (3.1) is shown in Figure 3.9, and from this figure we can see that the connectivity is maintained for Algorithm 3 while it is broken when applying algorithm (3.1). The tracking overshoot is evaluated in Figure 3.8, and we see that without the damping force term the tracking overshoot is big, and the network oscillates around the target.

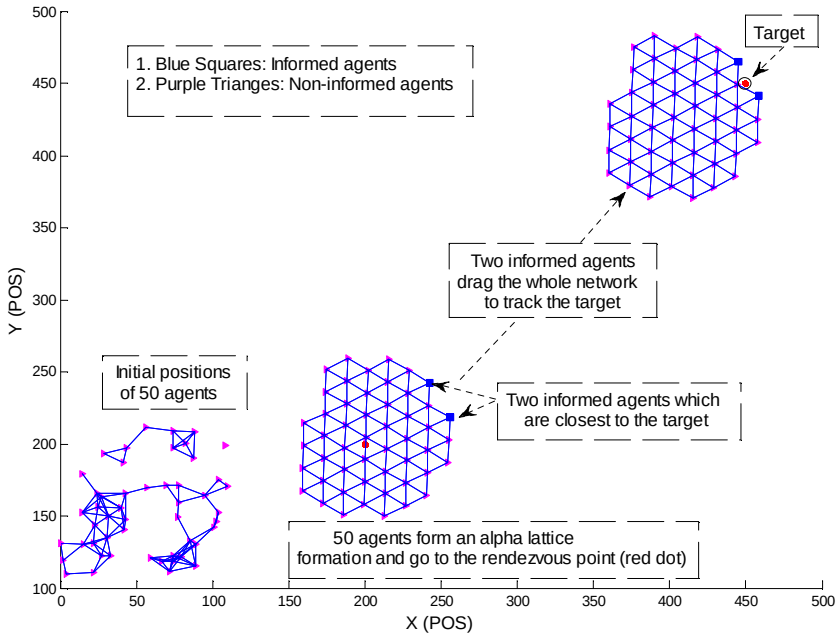


Figure 3.7: Snapshots of 50 robots flocking together (simulation) with two of them knowing the information of the target.

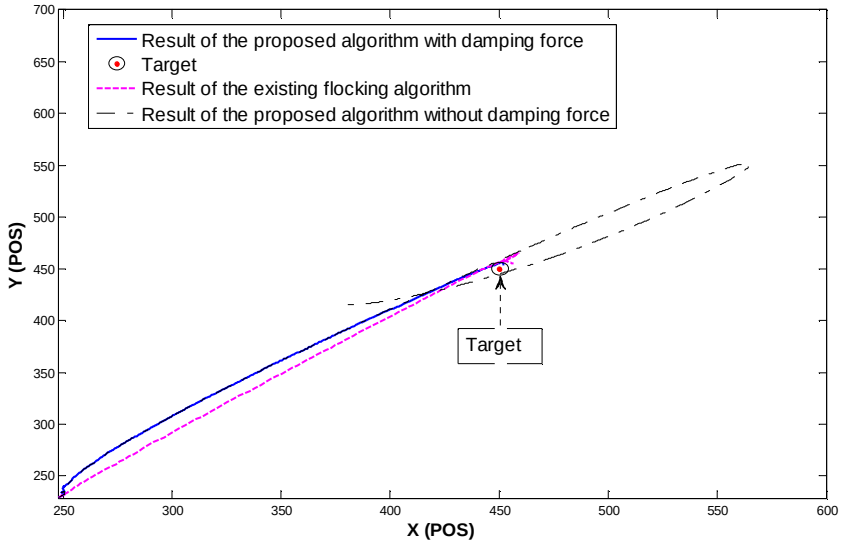


Figure 3.8: Average of positions of 2 informed robots (tracking performance) in simulation.

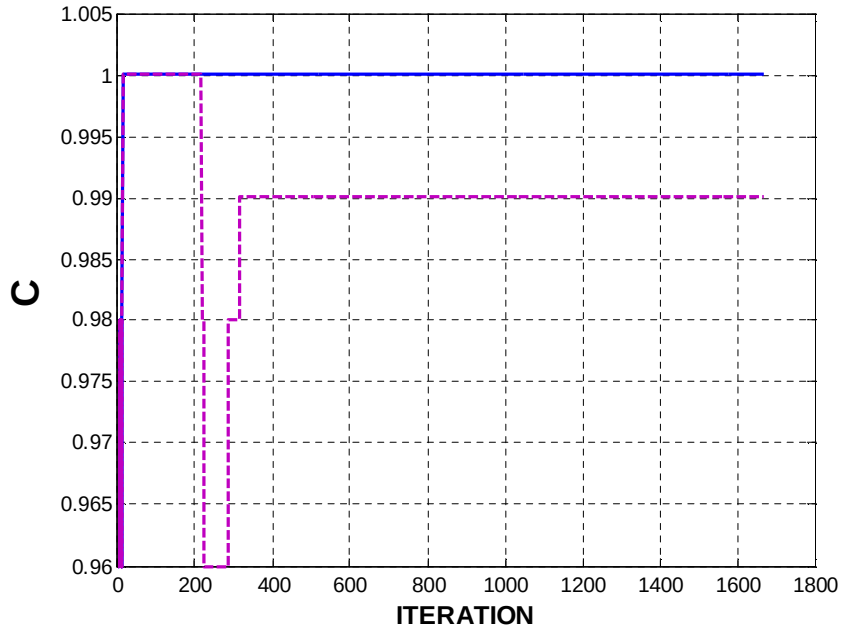


Figure 3.9: Connectivity evaluation in simulation of 50 robots. Solid line is for our proposed algorithm 3, and dash line is for the existing algorithm (3.1)

3.2 Adaptive Flocking Control for Moving Target Tracking

In this section, an adaptive flocking control algorithm is designed to allow an MSN to deal with complex environments while maintaining connectivity, tracking performance and similar formation. The stability analysis of the adaptive flocking control is provided. In addition, simulations and experiments are conducted to compare the adaptive flocking control and regular flocking control.

3.2.1 Problem Formulation

In reality, a mobile sensor network has to deal with changing or complex environments. For example the agents have to pass through a narrow space among obstacles. In that situation the existing flocking control algorithms have some limitations such as:

1. Formation of the network is totally changed.
2. Connectivity is lost because of the fragmentation phenomenon.
3. Low speed or getting stuck causes poor tracking performance.

Therefore designing an adaptive flocking control algorithm to deal with these problems is a challenging task. In this section, we present a novel approach to flocking control of a mobile sensor network to track a moving target with changing environments. In this approach, each agent cooperatively and adaptively learns the network's parameters to decide its's size in a decentralized fashion so that the connectivity, tracking performance and formation can be improved when avoiding obstacles. The reason for maintaining the connectivity and similar formation is that when the network shrinks to deal with changing environments the neighborhood of each agent can be maintained. This allows the network to keep the same topology that reduces the complexity of control during the tracking process. Computer simulations are conducted to prove our theoretical results.

The problem is how to cooperatively control the size of the network which forms an α-lattice configuration in an adaptive fashion while maintaining the network's connectivity,

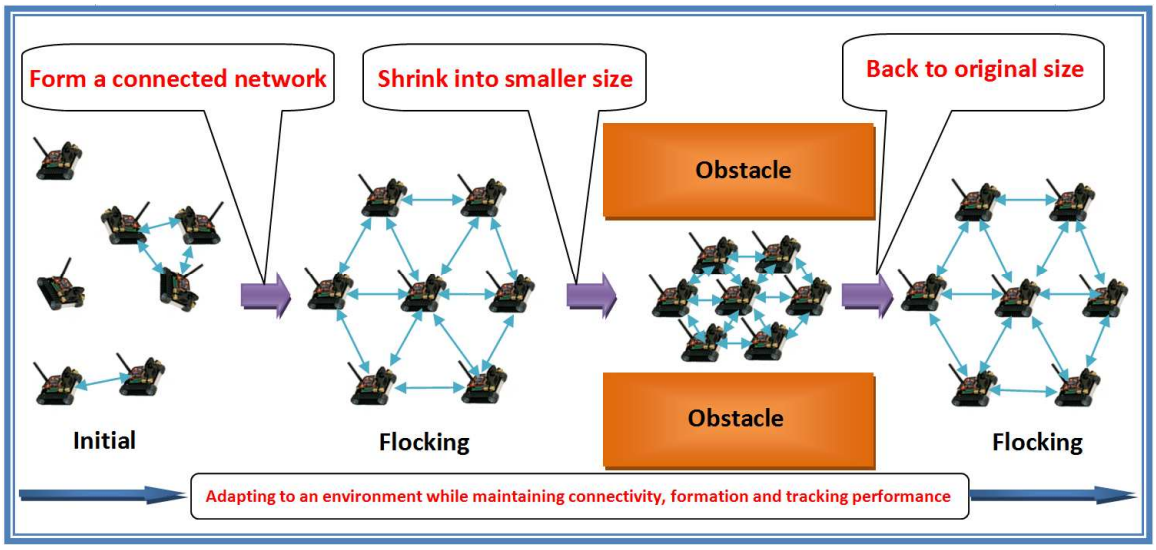


Figure 3.10: Illustration of the adaptive flocking control.

tracking performance and similar formation in the presence of obstacles. Here, the similar formation is understood as the neighbors of each agent in the whole tracking process are kept. One example of such flocking control is illustrated in Figure 3.10.

3.2.2 Adaptive Flocking Control

To control the size of the network, we need to control the set of algebraic constraints in Equation (2.20), which means that if we want the size of the network to be smaller to pass the narrow space then d^{α} should be smaller. This raises the question of how small the size of network should be reduced and how to control the size in a decentralized and dynamic fashion.

To control the constraint d^{α} one possible method is based on the knowledge of obstacle obtained by any agent in the network, which will broadcast a new d^{α} to all other agents, then the network will shrink into small size to pass through the narrow space between the obstacles. However, it is difficult for a single agent to learn the obstacles due to its limited sensing range. Therefore, one agent is not able to know the whole environment to determine the size of the network. To overcome this problem we propose the second method based

on the repulsive force, $\sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)$, which is generated by the β-agent projected on the obstacles. If any agent of the network gets this repulsive force it will shrink its own d_i^α. If this repulsive force is big (agent is close to obstacle(s)) d_i^α will be further reduced. Then, in order to maintain the neighbors (topology) the active range of each agent is re-designed. To create the agreement on the relative distance and active range among agents in a decentralized way, a consensus or a local average update law is proposed. Furthermore, to maintain the connectivity each agent is designed with an adaptive weight of attractive force from the target and an adaptive weight of interaction force from its neighbors so that the network reduces or recovers the size gradually. That is if an agent has weak connection to the network it should have big weight of attraction force to the target and small weight of interaction force from its neighbors.

Firstly, we control the set of algebraic constraints as in Equation (3.2)

$$\|q_j-q_i\|_\sigma = d_i^\alpha, j\in N_i \tag{3.2}$$

and let each agent have its own d_i^α, which is designed as in Equation (3.3)

$$d_i^\alpha = \begin{cases} d^\alpha, & if \quad \sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)=0 \\ \frac{c_a}{\sum_{k\in N_i^\beta}|\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)|+1}, & if \quad \sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)\neq 0. \end{cases} \tag{3.3}$$

here c_a is the positive constant.

From Equation (3.3) we see that if the repulsive force generated from the obstacles $\sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)=0$ or $N_i^\beta\in\emptyset$ (empty set) then the agent will keep its original d^α. When the agent senses the obstacles it reduces its own d_i^α, and how small d_i^α depends on the repulsive force that the agent gets from obstacles.

In order to control the size of network each sensor need its own r_i^α that relates to d_i^α as follows: $r_i^\alpha=\|kd\|_\sigma$ with $\|d\|_\sigma=d_i^\alpha$ or $d=\sqrt{\frac{(\varepsilon d_i^\alpha+1)^2-1}{\varepsilon}}$. Explicitly, r_i^α is computed as in Equation (3.4).

$$r_i^\alpha = \begin{cases} r^\alpha, & if \quad \sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)=0 \\ \frac{1}{\varepsilon}[\sqrt{k^2\frac{(\varepsilon d_i^\alpha+1)^2-1}{\varepsilon}+1}-1], & if \quad \sum_{k\in N_i^\beta}\phi_\beta(\|\hat{q}_{i,k}-q_i\|_\sigma)\neq 0. \end{cases} \tag{3.4}$$

Similar to computing r_i^α, r_i which also relates to r_i^α is computed as

$$r_i = \begin{cases} r, & if \quad \sum_{k\in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma) = 0 \\ \sqrt{\frac{1}{\varepsilon}[(\varepsilon r_i^\alpha + 1)^2 - 1]}, & if \quad \sum_{k\in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma) \neq 0. \end{cases} \tag{3.5}$$

It should be pointed out that the active range r_i is different from the physical communication (sensing) range. Namely, the active range is the range that each agent decides its neighbors to talk with, but the physical communication range is the range defined by the RF module. This implies that even a robot can communicate with all other robots in the network, it will only talk (interact) with robots in its active range. That is why we want to control the active range of each robot in order to reduce the communication and maintain the similar formation when the network shrinks into smaller sizes.

To achieve agreement on d_i^α, r_i^α and r_i among agents in the connected network we use the following update law based on local average for d_i^α, r_i^α and r_i:

$$\begin{cases} d_i^\alpha = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} d_j^\alpha \\ r_i^\alpha = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} r_j^\alpha \\ r_i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} r_j \end{cases} \tag{3.6}$$

here $|N_i^\alpha \cup \{i\}|$ is the number of agents in agent i's local neighborhood including agent i itself.

In addition, to better maintain the network connectivity each agent should have an adaptive weight of attractive force from the target and interaction force from its neighbors as discussed before. Firstly, in the control protocol (2.38), the first two terms are used to control the formation (velocity matching, collision avoidance among robots). The third and fourth terms are used to allow robots to avoid obstacles, and the last term is used for target tracking. If the last term is absent the control will lead to the network fragmentation [23]. The coefficients of the interaction forces (c_1^α, c_2^α), (c_1^β, c_2^β) and attractive force (c_1^{mt}, c_2^{mt}) which deliver desired swarm-like behaviour are used to adjust the weight of interaction forces and attractive force. Namely, the pair (c_1^α, c_2^α) is used to adjust the attractive/repulsive forces

among agent i and its actual neighbors (α-agent), and the pair $(c_1^{\beta},\ c_2^{\beta})$ is used to adjust the repulsive forces among agent i and its virtual neighbors (β-agent) that is generated from agent i when it see the obstacles, and the pair $(c_1^{mt},\ c_2^{mt})$ is used to adjust the attractive forces between agent i and its target. The bigger $(c_1^{mt},\ c_2^{mt})$ the faster convergence to the target. However if $(c_1^{mt},\ c_2^{mt})$ is too big the center of mass (CoM) as defined in Equation (2.41) oscillates around the target, and the formation of network is not guaranteed. In addition, in order to guarantee that no agent hit obstacle the pair $(c_1^{\beta},\ c_2^{\beta})$ is selected to be bigger than the other two pairs, $(c_1^{\alpha},\ c_2^{\alpha})$ and $(c_1^{mt},\ c_2^{mt})$. Finally we have the relationship among these pairs as: $(c_{1,2}^{\alpha} < c_{1,2}^{mt} < c_{1,2}^{\beta})$.

From the above analysis of choosing the coefficients of the interaction forces and attractive force we see that these adaptive weights allow the network to reduce and recover the size gradually. This also allows the network to maintain the connectivity during the obstacle avoidance. We will let each sensor have its own weight of the interaction forces as in Equation (3.7) and attractive force as in Equation (3.8). Keep in mind that in the α-lattice configuration if the sensor has less than 3 neighbors it is considered as having a weak connection to the network. This means that this sensor is on the border of network, or far from the target hence it should have bigger weight of attractive force from its target and smaller weight of interaction forces from its neighbors to get closer to the target. This design also has the benefit for the whole network to track the target faster. From this analysis $c_{1,2}^{\alpha}$ and $c_{1,2}^{mt}$ of each agent are designed as follows:

$$c_1^{\alpha}(i) = \begin{cases} c_1^{\alpha}, & if \quad |N_i^{\alpha}| \geq 3 \\ c_1^{\alpha'}, & if \quad |N_i^{\alpha}| < 3 \end{cases} \tag{3.7}$$

here $c_1^{\alpha'} < c_1^{\alpha}$, $c_2^{\alpha}(i) = 2\sqrt{c_1^{\alpha}(i)}$, and $i = 1, 2, ..., n$.

$$c_1^{mt}(i) = \begin{cases} c_1^{mt}, & if \quad |N_i^{\alpha}| \geq 3 \\ c_1^{mt'}, & if \quad |N_i^{\alpha}| < 3 \end{cases} \tag{3.8}$$

here $c_1^{mt'} > c_1^{mt}$, $c_2^{mt}(i) = 2\sqrt{c_1^{mt}(i)}$, and $i = 1, 2, ..., n$.

Hence, the neighbor set of sensor i at time t ($N_i^{'\alpha}(t)$), the new adjacency matrix $a_{ij}(q)$ and the new action function $\phi_\alpha(z)$ are defined as follows:

$$N_i^{'\alpha}(t) = \left\{ j \in \vartheta : \|q_j - q_i\| \leq r_i,\ \vartheta = \{1,2,...,n\},\ j \neq i \right\} \tag{3.9}$$

$$a_{ij}^{'}(q) = \begin{cases} \rho_h(\|q_j - q_i\|_\sigma / r_i^\alpha), & if \quad j \neq i \\ 0, & if \quad j = i \end{cases} \tag{3.10}$$

$$\phi_\alpha^{'}(\|q_j - q_i\|_\sigma) = \rho_h(\|q_j - q_i\|_\sigma / r_\alpha)\phi(\|q_j - q_i\|_\sigma - d_i^\alpha). \tag{3.11}$$

Finally, the adaptive flocking control law for dynamic target tracking is as follows,

$$\begin{aligned} u_i \quad = \quad & c_1^\alpha(i) \sum_{j \in N_i^{'\alpha}} \phi_\alpha^{'}(\|q_j - q_i\|_\sigma) n_{ij} \\ & + c_2^\alpha(i) \sum_{j \in N_i^{'\alpha}} a_{ij}^{'}(q)(p_j - p_i) \\ & + c_1^\beta \sum_{k \in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma)\hat{n}_{i,k} + c_2^\beta \sum_{k \in N_i^\beta} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ & - c_1^{mt}(i)(q_i - q_{mt}) - c_2^{mt}(i)(p_i - p_{mt}). \end{aligned} \tag{3.12}$$

3.2.3 Stability Analysis

By applying the control protocol (3.12), the CoM (defined in Equation (2.41)) of positions and velocities of all mobile sensors in the network will exponentially converge to the target in both free space and obstacle space. In addition, the formation or no collision and velocity matching among mobile sensors will maintain in the process of the moving target tracking.

Let us consider two cases of adaptive flocking control in free space and obstacle space, respectively.

Case 1 (Free space): In free space, this means that $\sum_{k \in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma) = 0$. Hence we can rewrite the control protocol (3.12) with ignoring constants c_η^ν (for $\forall \eta = 1,2$ and $\nu = \alpha, \beta$) as follows:

$$\begin{aligned} u_i \quad = \quad & -\sum_{j \in N_i^\alpha} \nabla_{q_i} \psi_\alpha(\|q_j - q_i\|_\sigma) + \sum_{j \in N_i^\alpha} a_{ij}(q)(p_j - p_i) \\ & - c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) \end{aligned} \tag{3.13}$$

where $\psi_\alpha(z) = \int_{d_\alpha}^{z} \phi_\alpha(s)ds$ is the pairwise attractive/repulsive potential function. From (3.13), we can compute the center of mass of control law u as follows:

$$\begin{aligned} \overline{u} = \frac{1}{n}\sum_{i=1}^{n} u_i &= \frac{1}{n}\sum_{i=1}^{n}(-\sum_{j\in N_i^\alpha} \nabla_{q_i}\psi_\alpha(\|q_j - q_i\|_\sigma) \\ &\quad + \sum_{j\in N_i^\alpha} a_{ij}(q)(p_j - p_i)) \\ &\quad -c_1^{mt}(\overline{q} - q_{mt}) - c_2^{mt}(\overline{p} - p_{mt}). \end{aligned} \tag{3.14}$$

Obviously, we see that the pair $(\psi_\alpha, a(q))$ are symmetric. Hence we can rewrite (3.14) as:

$$\overline{u} = -c_1^{mt}(\overline{q} - q_{mt}) - c_2^{mt}(\overline{p} - p_{mt}). \tag{3.15}$$

Equation (3.15) implies that

$$\begin{cases} \dot{\overline{q}} = \overline{p} \\ \dot{\overline{p}} = -c_1^{mt}(\overline{q} - q_{mt}) - c_2^{mt}(\overline{p} - p_{mt}). \end{cases} \tag{3.16}$$

The solution of (3.16) indicates that the position and velocity of the CoM exponentially converge to those of target.

The formation or collision-free and velocity matching among mobile sensors are kept in the free space tracking because the gradient-based term and the consensus term are considered in this situation (more details please see [23]).

Case 2 (Obstacle space): d_i^α is designed to be reduced when each agent senses the obstacles. Therefore, when the sensor network has to pass through the narrow space between two obstacles it will shrink the size gradually, and when the network already passed this narrow space it grows back to the original size gradually. This reduces the impact of the obstacle on the network hence the speed of agents can be maintained or the CoM keeps tracking the target. Also, the connectivity and similar formation can be maintained in this scenario.

3.2.4 Simulation and Experiment Results

The parameters used in the simulation and experiment of the adaptive flocking are specified as follows:

- Parameters of flocking in simulation: we use 50 mobile sensor nodes which are randomly distributed in the box of 100x100 size. Other parameters are $a = b = 5$; the active range $r = 8.5$; the desired distance $d = 7$; $\varepsilon = 0.1$ for the σ-norm; h = 0.2 for the bump functions ($\phi_\alpha(z)$, $\phi'_\alpha(z)$); h = 0.9 for the bump function ($\phi_\beta(z)$).

Parameters of target movement for simulation: The target moves in the line trajectory: $q_t = [100 + 130t,\ t]^T$.

- Parameters of flocking in experiment:

$a = b = 5$; $d = 1100mm$; the scaling factor $k_c = 1.2$; the active range $r = k_c * d$; $\varepsilon = 0.1$ for the σ-norm; h = 0.2 for the bump functions ($\phi_\alpha(z)$, $\phi'_\alpha(z)$); h = 0.9 for the bump function ($\phi_\beta(z)$).

Parameters of target movement for experiment: The virtual target moves in the line trajectory: $q_t = [230 + t,\ -3000 + 130t]^T$.

- Experimental setup: In this experiment we use 7 Rovio robots [111] that have omni-directional motion capability. Basically, these robots can freely move in 6 directions. The dynamic model of the Rovio robot can be approximated by Equation (2.18). However, the localization accuracy of the Rovio robot is low, and the robot does not has any sensing device to sense the pose (position and velocity) of its neighbors or the obstacles. Hence we use a VICON motion capture system setup [1] in our lab (Figure 3.11) that includes 12 infrared cameras to track moving objects. This tracking system can provide the location and velocity of each moving object with high accuracy.

Figures 3.12 represents the results of moving target (red/dark line) tracking in the line trajectory using the existing flocking control algorithm (2.38). Figure 3.13 represents the results of moving target tracking in the line trajectory using the adaptive flocking control algorithm (3.12). Figure 3.14 shows the results of velocity matching among agents (a, a'), connectivity (b, b') and error positions between the CoM (black/darker line) and the target (tracking performance) (c, c') of both flocking control algorithms (3.12) and (2.38), respectively. To compare these algorithms we use the same initial state (position and velocity) of

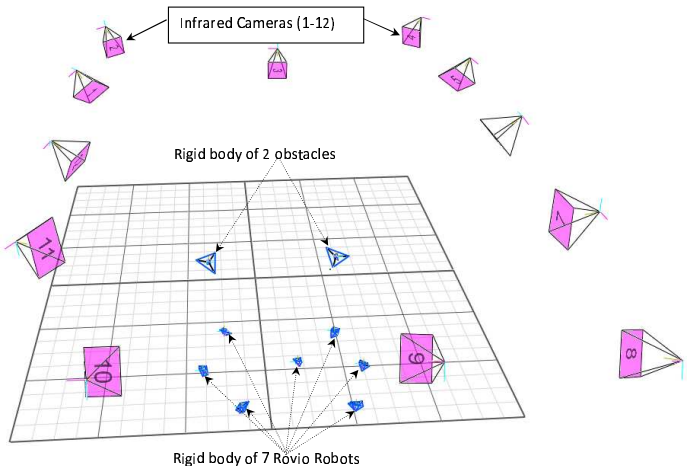


Figure 3.11: Experimental setup for adaptive flocking control.

mobile agents. By comparing these figures we see that by applying the adaptive flocking control algorithm (3.12) the connectivity, similar formation and tracking performance are maintained when the network passes through the narrow space between two obstacles (two red/dark circles) while the existing flocking control algorithm (2.38) could not handle these problems. In Figures 3.13 when the network enters the small gap between two obstacles its size is shrunk gradually in order to pass this space, then the network size grows back gradually when it passed. Therefore the connectivity and similar formation are maintained.

Figure 3.15 shows the snapshots (a to f) of the experiment result for 7 Rovio robots using our adaptive flocking algorithm (3.12). The results look similar with the simulation result in Figure 3.13. Figure 3.16 (Left) shows the trajectories of 7 robots in simulation, and Figure 3.16 (Right) compares the trajectories of 7 robots in both simulation and experiment.

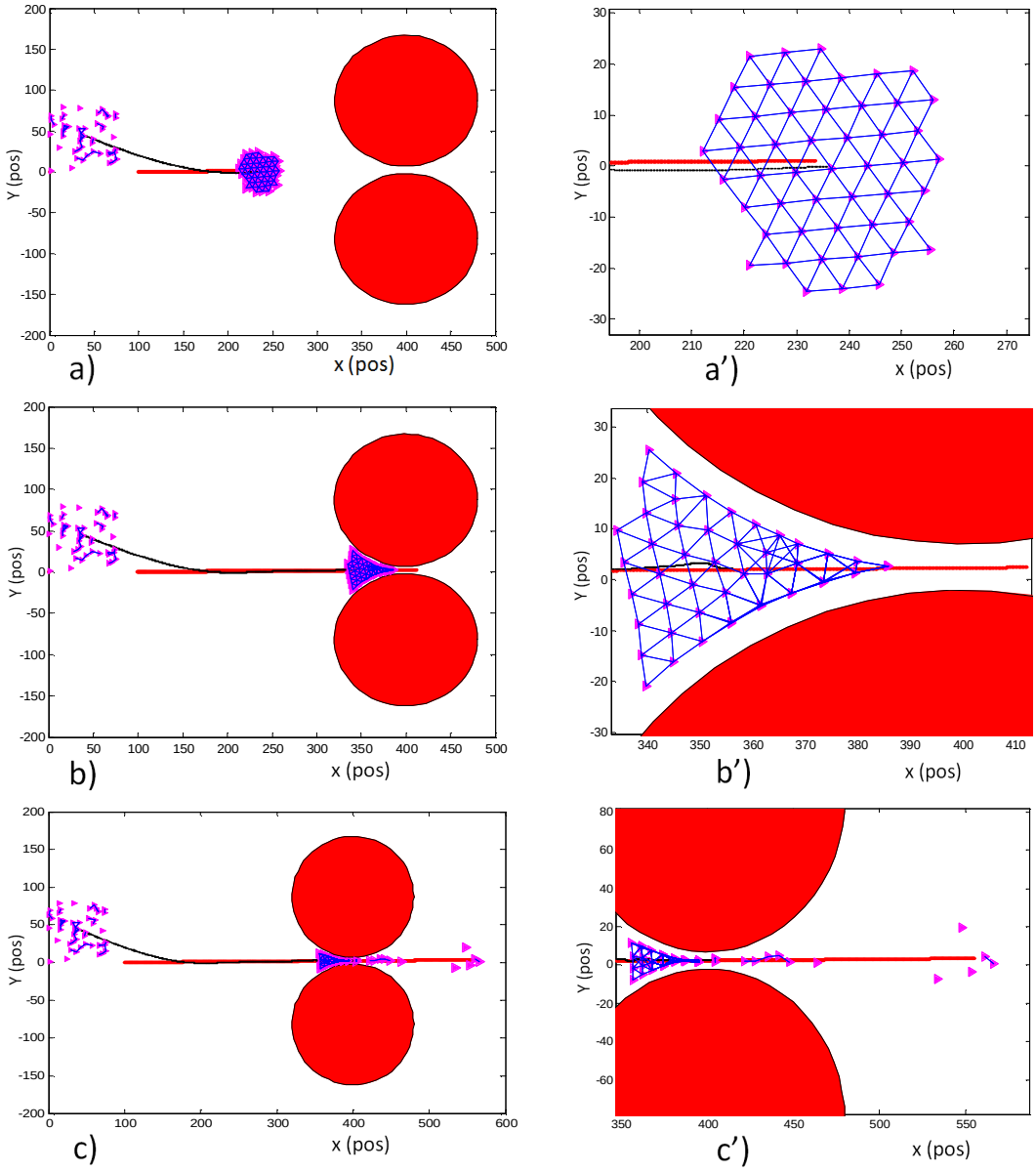


Figure 3.12: Snapshots of the mobile sensor network (a) when the mobile sensors form a network, (b) when the mobile sensors avoid obstacles, (c) when the mobile sensors get stuck in the narrow space between two obstacles. (a', b', c') are closer look of (a, b, c), respectively. These results is obtained by using algorithm (2.38).

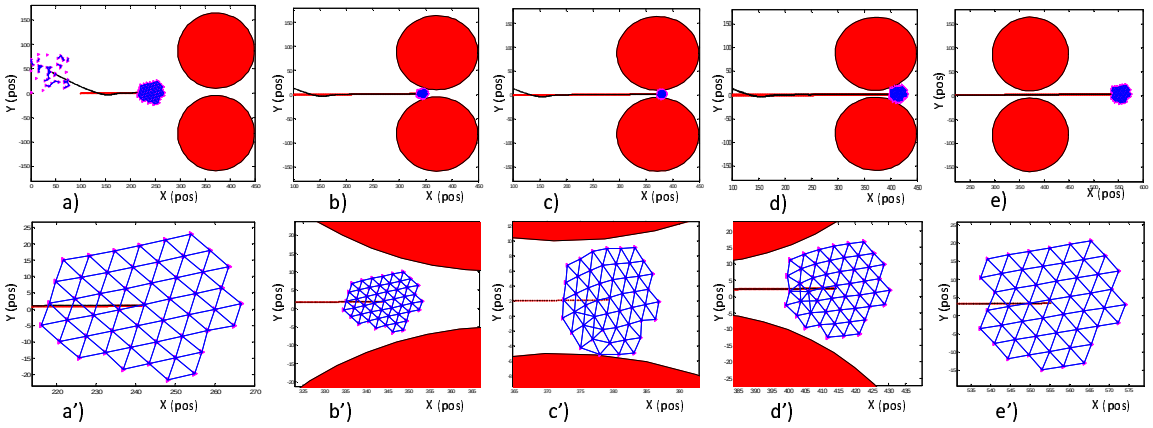


Figure 3.13: Snapshots of the mobile agent network (a) when the mobile agents form a network, (b, c) when the mobile agent network shrinks to avoid obstacles, (d) when the mobile agents successfully passed through the narrow space between two obstacles, (e) when the mobile agents recover the original size. (a', b', c', d', e') are closer look of (a, b, c, d, e), respectively. These results are obtained by using our adaptive flocking control algorithm (3.12).

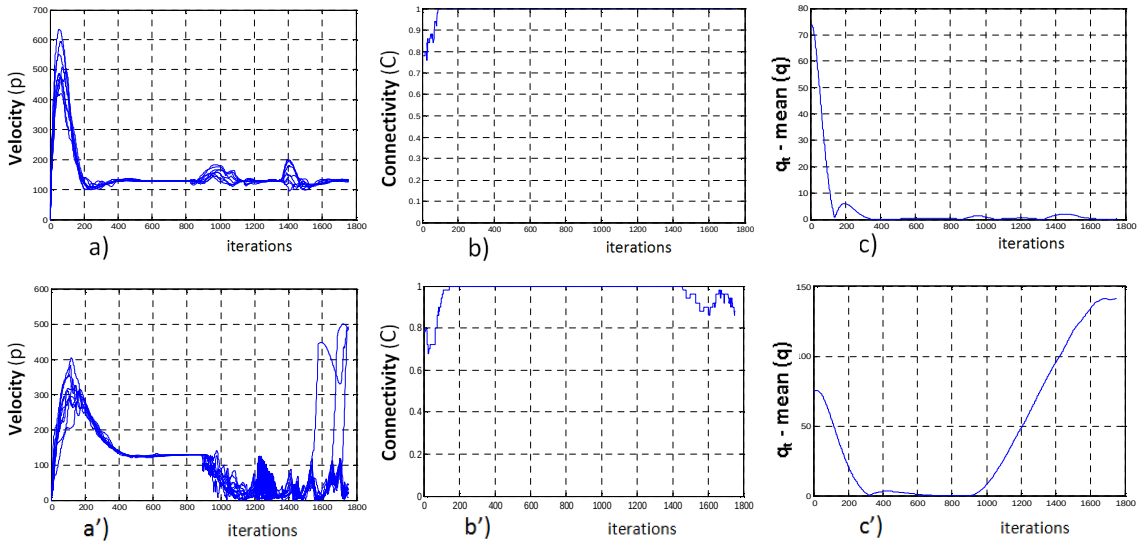


Figure 3.14: Velocity matching among agents, connectivity, and error of positions between the CoM and the moving target in (a, b, c), respectively using our adaptive flocking control algorithm (3.12), (a', b', c') using the algorithm (2.38).

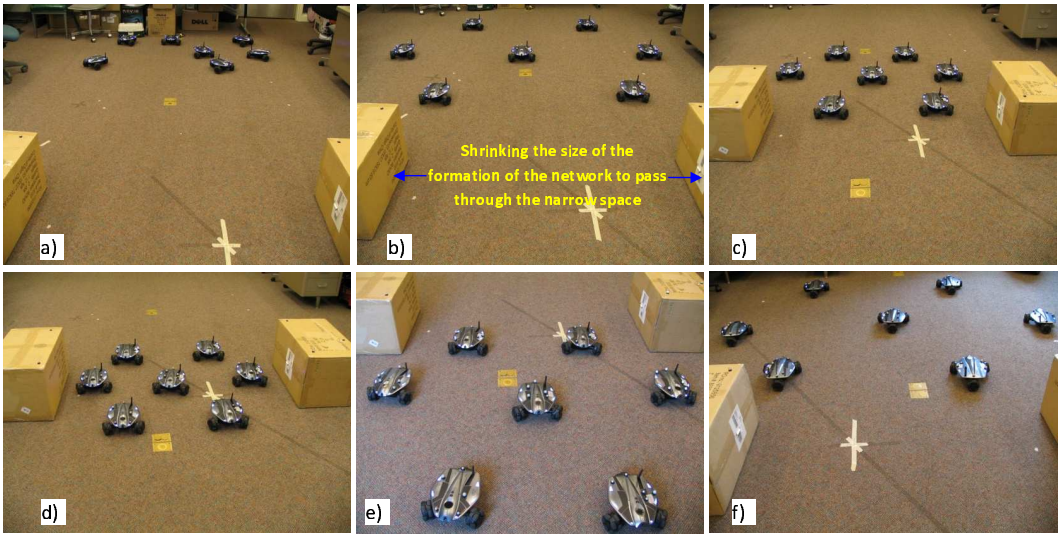


Figure 3.15: Snapshots of adaptive flocking control with 7 Rovio robots using our adaptive flocking control algorithm (3.12). (a) 7 robots are randomly distributed. (b) 7 robots form a lattice formation. (c) 7 robots begin to shrink the size of the network. (d) 7 robots pass through the narrow space between 2 obstacles. (e) 7 robots begin to recover the size of the network. (f) 7 robots completely recover the size of the network.

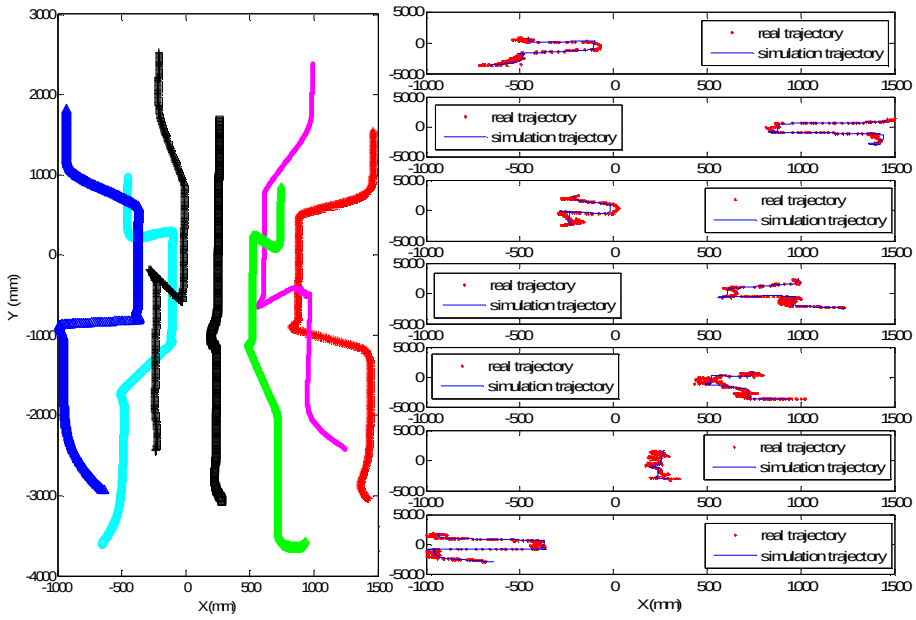


Figure 3.16: Trajectories of 7 robots are obtained by using the adaptive flocking control algorithm (3.12).

3.3 Multiple Dynamic Targets Tracking

In many surveillance applications MSNs have to deal with the dynamic situation of targets appearing and disappearing in the field. In this section we first address the problem of sensor network partitioning and then discuss multiple dynamic targets tracking through sensor splitting and merging.

3.3.1 Sensor Network Partitioning

To deal with a new emerging target, the sensor network should automatically decompose into equal sub-groups and then each sub-group will be assigned to track one target. For example, consider M targets existing at time t and M sensor groups $(G_1, G_2, ..., G_M)$ which are tracking these targets (each group has about n/M sensors). If the $(M+1)$th target appears then $\frac{n}{M+1}$ sensors should split off from M existing groups to form a new group to track the new target. On the other hand to deal with a disappearing target, the sensors which are tracking this target should split and merge with the existing groups.

As discussed in Chapter 2, the mobile sensor network can be considered as a dynamic graph (dynamic topology). Hence we can apply some graph partitioning algorithms to decompose the graph into sub-graphs (sub-groups). However, some existing methods for graph partitioning are centralized methods, which means that each sensor need global knowledge of the whole network's state to split from the network. There are also some distributed graph partitioning or distributed graph clustering methods, but they are usually based on the density of node's distribution (see *Literature review section*). Hence the size of sub-groups is not predetermined, or the number of sensors in each sub-group is different.

Based the above analysis, this section proposes a seed growing graph partition (SGGP) algorithm to decide which sensor in the network should track new targets. The main idea of this algorithm is based on seed growing. This means that the mobile sensor which is closest to the new target will initiate the growth of the sensors into a new group by broadcasting

the message to its sons in a recursive fashion until the number of sensors in the subgroup is equal to a predetermined threshold (Θ_S). By growing the number of sensors in each generation from the seed sensor (the sensor closest to the new target), the formation of each sub-group is maintained during splitting. This leads to minimized total energy and time consumption.

Assume all mobile sensors already formed a network with an α-lattice configuration (see Figure 3.17). In this configuration if the sensor has 5 or 6 neighbors (6 is the maximum number of neighbors in this configuration) this sensor will be inside the network. If the sensor has less than or equal to 4 neighbors it will be on the border of the network. This sensor is called a border sensor. Based on this fact, the SGGP algorithm is summarized as follows:

Step 1. Each sensor checks to find how many neighbors it has and decides if it is a border sensor.

Step 2. Each border sensor computes the distance to the new target and forwards this distance information to the other border sensors, and receives the distances from other border sensors.

Step 3. Each border sensor compares its distance with the received distances from other border sensors and finds the sensor with smallest distance to be set as the Seed Sensor (SS).

Step 4. The SS counts its sons and broadcasts the predetermined size of the new group to its sons. If the size of the new group is less than the predetermined size the sons will continue passing the message to their sons. This process is repeated until the size of the new group is equal to the predetermined size.

Remark 2. In the SGGP algorithm, the number of sons of sensor i is defined as:

$$|S_i| = |N_i| - |F_i| - |DB_i| \tag{3.17}$$

here $|S_i|$, $|N_i|$, $|F_i|$ and $|DB_i|$ are the number of sons, neighbors, fathers and direct brothers of sensor i, respectively. For example in Figure 3.17, SS is the father of sensors 2, 3 and 4. Sensor 3 is the direct brother of sensor 2, hence the sons of sensor 2 are only sensors 5

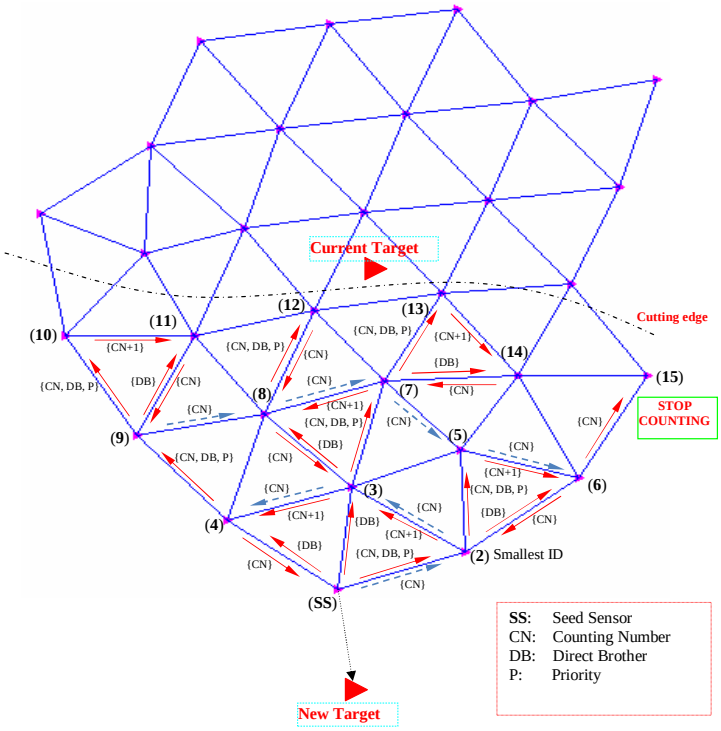


Figure 3.17: Example of seed growing graph partition.

and 6. Sensor 2 can know sensor 3 being its direct brother because its father (SS) sends a message $\{DB\}$ to tell which sensor is its direct brother. In addition, two or more sensors can have the same son, but if a sensor has the priority $\{P\}$ to count this same son first the remaining sensors will not count this son again. For an example of this situation, sensors 2 and 3 have the same son, sensor 5, but because of its smaller ID sensor 2 receives a message consisting of $\{P\}$ from its father (SS) hence it has priority to count sensor 5 as its son first then it sends the counting number (CN) to its direct brother sensor 3.

Figure 3.17 shows the message exchange when applying the SGGP algorithm. The slashed green arrows represent the counting number (CN) which is sent after counting, and the solid red arrows represent the message exchange. In this scenario assuming that we have 30 sensors (n=30), and they already formed a network with α-lattice configuration. This sensor network is tracking the current target. When a new target appears, by applying the SGGP algorithm 15 sensors (Θ_S = n/2) split from the network to track the new target with the total distance of all n/2 sensors to the new target being minimized.

3.3.2 Multiple Dynamic Targets Tracking

In the multiple targets scenario, we assume that each sensor is integrated with the flocking control algorithms with *No-CoM* (3.18) and *Multi-CoM* (3.19), respectively, which deal with each different target (q_{mt_l}, p_{mt_l}) with $l = 1, 2, ..., M$ described as below.

$$\begin{aligned} u_i &= c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ &\quad + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ &\quad - c_1^{mt}(q_i - q_{mt_l}) - c_2^{mt}(p_i - p_{mt_l}). \end{aligned} \tag{3.18}$$

$$\begin{aligned} u_i &= c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ &\quad + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ &\quad - c_1^{mt}(q_i - q_{mt_l}) - c_2^{mt}(p_i - p_{mt_l}) \\ &\quad - c_1^{mc}(\overline{q}_{(N_i^{\alpha} \cup \{i\})} - q_{mt_l}) - c_2^{mc}(\overline{p}_{(N_i^{\alpha} \cup \{i\})} - p_{mt_l}). \end{aligned} \tag{3.19}$$

As discussed in Chapter 2, the dynamic target (q_{mt_l}, p_{mt_l}) in (3.18) or (3.19) is exactly the navigation term that makes the flocks (mobile sensors) move together. Without this term the sensor network leads to fragmentation. This means that if sensor i is assigned to track another target it only need switch to another navigation term. This also means that if the new target appears one by one the sensors which are selected by the SGGP algorithm will switch to another navigation term (another target).

On the other hand in the merging case, three sensor subgroups are tracking three targets. If one of these targets disappears then this subgroup will decompose into two equal parts and each one will merge into one of remaining subgroups to track the existing targets by switching to the another navigation term.

3.3.3 Experimental Tests

SGGP Algorithm and Flocking Control (with *No-CoM*)

In this sub-section we will test the SGGP algorithm and flocking control (with *No-CoM*) (3.18) in two different cases of sensor splitting and merging. Parameters used in this simulation are specified as follows:

Case1. Two targets appear one by one and no target disappears.

- Parameters of flocking: Number of sensors = 120 (randomly distributed in the square area with the size of 90x90). Positions of obstacles
y_k= $[220\ 300;\ 220\ 360; 250\ 120;\ 250\ 60]^T$; Radii of obstacles R_k= [16; 16; 16; 16], and the communication range $r = 1.2 * d$ with $d = 7.5$; $\varepsilon = 0.1$ for the σ-norm.

- Parameters of target movement: The targets move in the sine wave trajectory: For the target 1, $q_{mt_1} = [50 + 35t,\ 295 - 35sin(t)]^T$ with $0 \leq t \leq 8.5$, and for the target 2, $q_{mt_2} = [85 + 35t,\ 55 - 35sin(t)]^T$ with $1.26 \leq t \leq 8.5$, and $\Delta_t = 0.002$ is the step size.

In this case, the SGGP algorithm will be compared with a Random Selection (RS) algorithm. In the RS algorithm when the new target appears a half of the sensors in the network which are tracking the existing target are selected randomly to track the new target.

Case2. Two targets appear one by one and one target disappears.

- Parameters of flocking: these parameters are the same with the Case 1.

- Parameters of target movement: Parameters are set up the same as in Case 1, but the target 1 is set to run in the interval time $0 \leq t \leq 12.5$, and the target 2 appears at time $t = 1.26$ (at iteration 840) and disappears at time $t = 8.4$ (at iteration 4200).

Figure 3.18 (a) displays the result of tracking of Case 1 where the targets appear one by one and move in a sine wave trajectory. Firstly, the whole group of 120 mobile sensors form an α-lattice configuration and track target 1. Then, at iteration 840 target 2 appears and the network decides which sensors will split and track this target. By applying the SGGP algorithm, the sensor network automatically decomposes into 2 equal sub-groups

(60 sensors in each sub-group). The second sub-group which is closest to target 2 tracks target 2, and the first sub-group keep tracking target 1. The SGGP algorithm allows two sub-groups to maintain their formation when they split. Figure 3.18(b) represents the error between the average of positions in the whole network and target 1 (from iteration 1 to 839), and the error between the average of positions in sub-group 1 and target 1 (from iteration 840 to the end). Figure 3.18(c) represents the error between the average of positions in sub-group 2 and target 2. We see that at iteration 840, the average of positions of sensors slightly changes because at this time the average sensors's positions in sub-group 1 will replace that of the whole network. In this figure we see that all tracking errors are very small in free space. This means that all sensors in the whole network or in each sub-group can surround the target closely to observe it easily. However in the presence of obstacles, the errors are significant because the repulsive forces generated from obstacles push the sensors away from them.

Figures 3.19 shows the results of tracking in Case 2 where the targets appear one by one and then one disappears. When target 2 appears at iteration 840 the results are similar with Figures 3.18. When target 2 disappears at iteration 4200 sub-group 2 which is tracking this target will rejoin to sub-group 1 and continue to track target 1. The tracking result of the whole group after merging is good with small tracking error between the average of sensors's positions and target 1 in the free space as shown in Figure 3.19 (b) (from iteration 4200 to the end).

Comparison Between the SGGP Algorithm and the RS Algorithm

In this subsection we will compare two algorithms, SGGP and RS, in term of tracking time, formation time, and total distance of all sensors in each sub-group to its target. These comparisons also imply the time consumption and power consumption in each sub-group.

Similar to Figures 3.18, Figures 3.20 also shows the results of tracking to Case 1 where the targets appear one by one and move in the sine wave trajectory. However, the difference

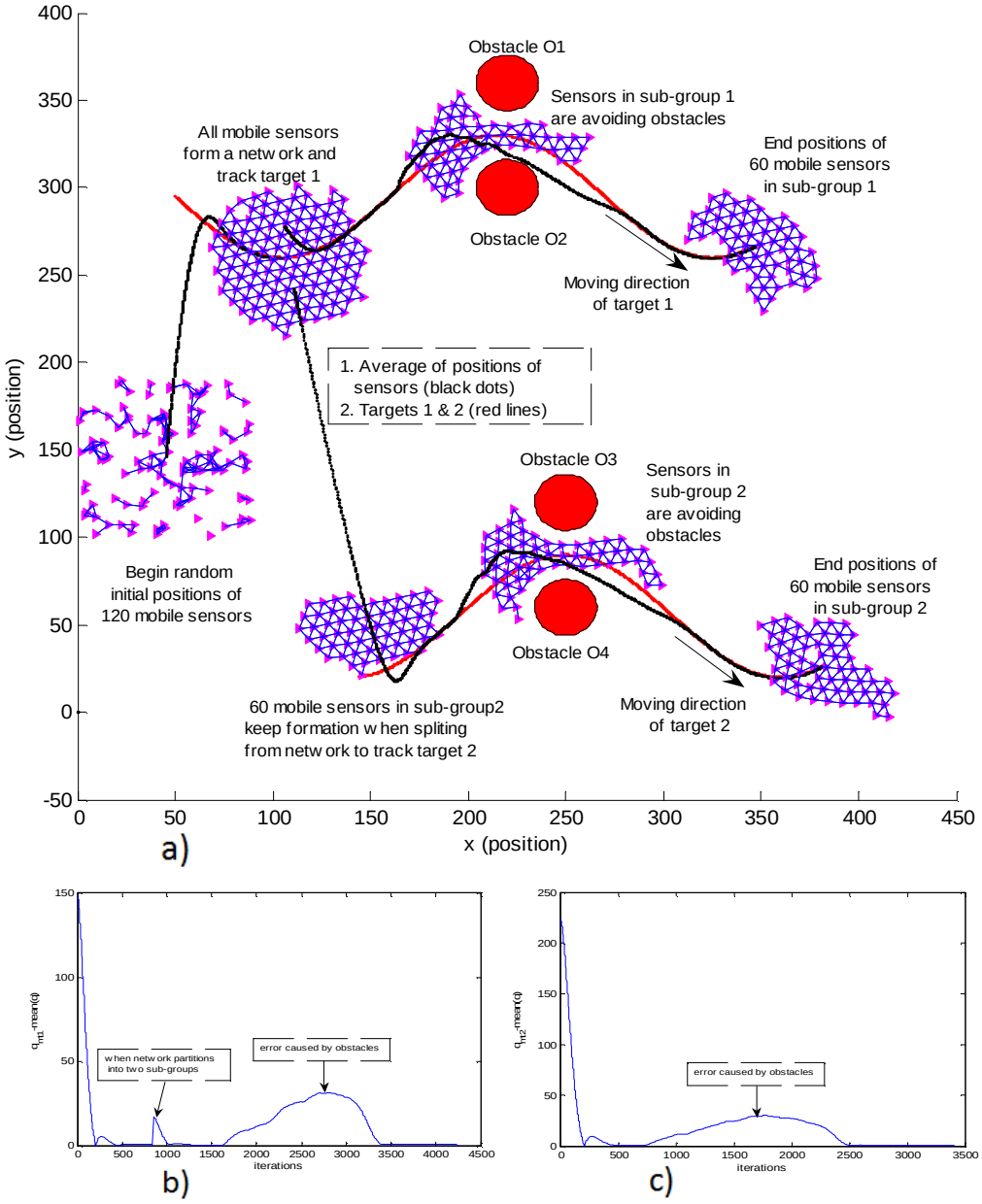


Figure 3.18: (a)- Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, forming a network at time $t = 1.26$, and decomposing into two sub-groups, respectively to track the targets moving in the sine wave trajectories, (b)- Error between the average of sensors's positions in the whole network and the moving target 1 (iteration 1 to 839), and between average of sensors's positions in sub-group 1 and the moving target 1 (iteration 839 to the end), (c)- Error between the average of sensors's positions in sub-group 2 and the moving target 2. This result is obtained by using the flocking control *No-CoM* (3.18) and SGGP algorithm

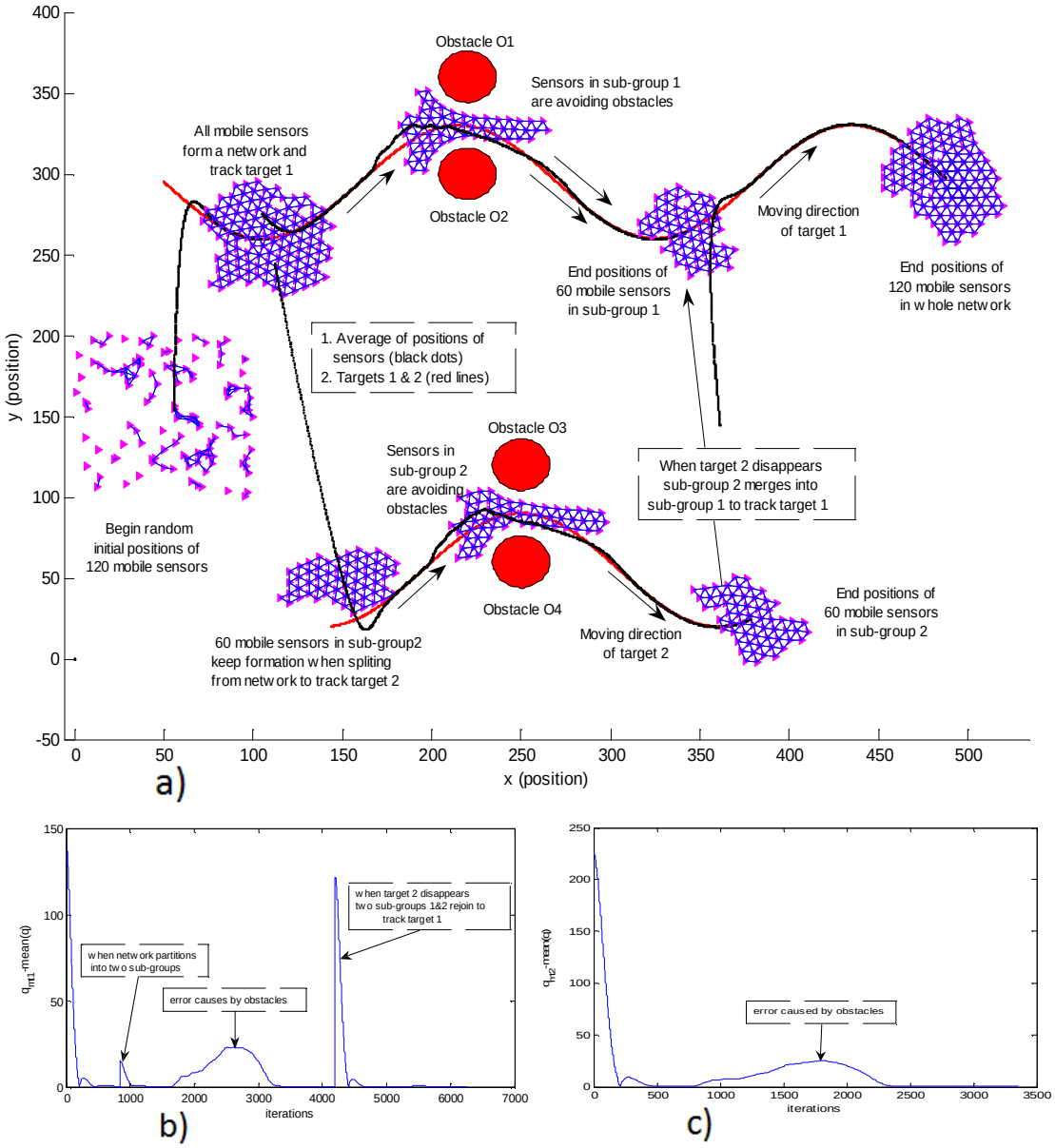


Figure 3.19: (a)- Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, when the mobile sensors form a network at time $t = 1.26$, when the mobile sensors decompose into two sub-groups, and when two sub-groups merge, (b)- Error between the average of sensors's positions in the whole network and the moving target 1 (iteration 1 to 839, and iteration 4200 to the end), and between the average of sensors's positions in sub-group 1 and the moving target 1 (iteration 840 to 4200), (c)- Error between the average of sensors's positions in sub-group 2 and the moving target 2. This result is obtained by using the flocking control with *No-CoM* (3.18) and SGGP algorithm.

here is that when target 2 appears a half of the sensors in the whole network are split to track this target by using the RS algorithm. With this algorithm two sub-groups do not maintain their formation, and all sensors in each sub-group need certain time to reform a network. This is the main drawback of this algorithm, and some data are collected to compare the SGGP and the RS algorithms which is shown in Table 3.1.

Table 3.1: Comparison between two algorithms (SGGP and RS).

Algorithms	D_{tt} (units)	t_T (s)	t_F (s)
RS (G_1)	1184.7	1.000801	8.345623
RS (G_2)	14194	11.770489	11.125117
SGGP(G_1)	1185.6	1.203569	0.0
SGGP(G_2)	13126	9.007456	0.0

Parameters in the Table 3.1 are computed as follows:

D_{tt} is the total travel distance between all sensors in the each group and its target, and it is computed when the network is decomposed into sub-groups to when the average of positions of sensors in each sub-group reaches the target (this is evaluated based on the same condition as used to compute t_T below).

t_T is the tracking time which is computed based on the condition: $\|\frac{1}{n_{G_l}}\sum_{i=1}^{n_{G_l}} q_i - q_{t_l}\| \leq \Theta_T$, $l = 1,2$; here n_{G_l} is number of sensors in each sub-group G_1 and G_2, respectively, and Θ_T is a given threshold.

t_F is the formation time representing the time that it costs all mobile sensors to form a network. This formation time is computed based on the following condition:

$Var(\|q_i - q_j\|) = \frac{1}{|E_l|}\sum(\|q_i - q_j\| - \frac{1}{n_{G_l}}\sum_{(i,j)\in E_l}\|q_i - q_j\|)^2 \leq \Theta_3$ with $i, j = 1,2,...,$ n_{G_l}; $l = 1,2$; here Θ_F is a given threshold, and $i \neq j$.

In the RS algorithm, the values of D_{tt}, t_T, and t_F are obtained based on the average value of 50 running times.

Comparison between the RS and the SGGP algorithms: The maximum of the track-

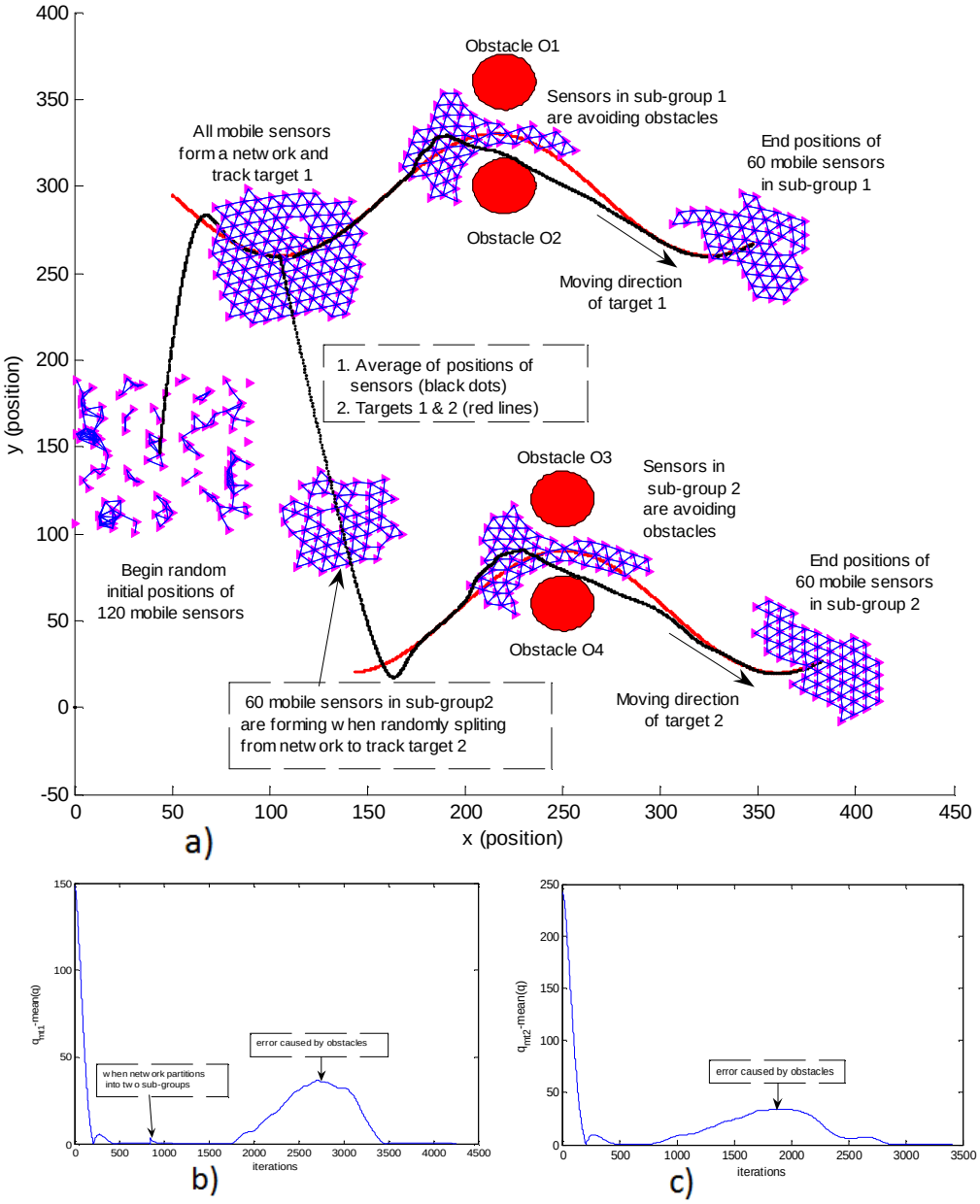


Figure 3.20: (a)- Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, forming a network at time $t = 1.26$, and decomposing into two sub-groups, respectively to track the targets moving in the sine wave trajectories, (b)- Error between the average of sensors's positions in the whole network and the moving target 1 (iteration 1 to 839), and between average of sensors's positions in sub-group 1 and the moving target 1 (iteration 840 to the end), (c)- Error between the average of sensors's positions in sub-group 2 and the moving target 2. This result is obtained by using the flocking control with *No-CoM* (3.18) and RS algorithm.

ing time and formation time in SGGP algorithm $t_{SGGP}^{max} = max(t_T, t_F)_{G_1} + max(t_T, t_F)_{G_2} = 10.211(s)$ while in RS algorithm $t_{RS}^{max} = 20.1161(s)$, or t_{SGGP}^{max} is 49.28 % less than t_{RS}^{max}. The total distance in SGGP algorithm $D_{SGGP}^{t} = D_{tt}^{G_1} + D_{tt}^{G_2} = 14311.6(units)$ while in RS algorithm $D_{RS}^{t} = 15378.7(units)$, or D_{SGGP}^{t} is 7% shorter than D_{RS}^{t}.

SGGP Algorithm and Flocking Control (with *Multi-CoM*)

In this sub-section we will test the SGGP algorithm and flocking control (with *Multi-CoM*) (3.19) in two different cases of sensor splitting and merging. Parameters used in this simulation are specified as follows:

Case1. Two targets appear one by one and no target disappears.

- Parameters of flocking: Number of sensors = 60 (randomly distributed in the box with the size of 50x50). Positions of obstacles y_k= $[190\ 720;\ 150\ 330; 200\ 106;\ 200\ 10]^T$; Radii of obstacles R_k= [16; 16; 16; 16], and other parameters $a = b = 5$; the communication range $r = 7.8$ with $d = 6.5$; $\varepsilon = 0.1$ for the σ-norm; h = 0.2 for the bump function ($\phi_\alpha(z)$); h = 0.9 for the bump function ($\phi_\beta(z)$).

- Parameters of target movement: The targets move in the sine wave trajectory: For the target 1, $q_{mt_1} = [50 + 35t,\ 295 - 35sin(t)]^T$ with $0 \leq t \leq 6$, and $p_{mt_1} = (q_{mt_1}(t) - q_{mt_1}(t-1))/\Delta_t$, and for the target 2, $q_{mt_2} = [85 + 35t,\ 55 - 35sin(t)]^T$ with $1.26 \leq t \leq 6$, and $p_{mt_2} = (q_{mt_2}(t) - q_{mt_2}(t-1))/\Delta_t$.

In this case, the SGGP algorithm will be compared with the Random Selection (RS) algorithm where the sensors are selected randomly to track targets.

Case2. Two targets appear one by one and one target disappears.

- Parameters of flocking: these parameters are the same with the Case 1.

- Parameters of target movement: Parameters are set up the same with the Case 1, but the target 1 is set to run in the interval time $0 \leq t \leq 7.5$, and the target 2 appears at time $t = 1.26$ and disappears at time $t = 4.95$.

Figure 3.21 represents the result of tracking of Case 1 where the targets appear one by

one and move in the sine wave trajectory. Firstly, the whole group of 60 mobile sensors form the network with α-lattice configuration and track the target 1. Then, at time $t = 1.26$ the target 2 appears and the network should decide which sensor will split and track this target. By applying the SGGP algorithm, the sensor network automatically decomposes into 2 equal sub-groups (30 sensors in each sub-group). The second sub-group which is closest to the target 2 will go to track this target, and the first sub-group keep tracking the target 1. The SGGP algorithm allows two sub-groups maintaining their formation when they split from the network to track targets. Figure 3.22 represents the errors between the CoM of positions and target. Here Figure 3.22(b) is the error between the CoM of positions of the whole network and target 1 (from iteration 1 to 839), and the error between the CoM of positions of sub-group1 and target 1. Figure 3.22(a) is the zoom in of Figure 3.22(b) at iterations from 1 to 100 for ease to see. We see that at time $t = 1.26$ or iteration = 840, the CoM slightly changes because at this time the CoM of sub-group 1 will be replaced that of the whole network. Here Figure 3.22(d) is the error between the CoM of positions of sub-group 2 and target 2. Figure 3.22(c) is the zoom in of Figure 3.22(d) at iteration from 1 to 100. In this figure we see that all the errors are very small. This means that all sensors in the whole network or in each sub-group can surround the target closely. Similar with Figures 3.21 and 3.22, Figures 3.23 and 3.24 also represent the results of tracking of the case 1 where the targets appear one by one and move in the sine wave trajectory. However, the difference here is that when target 2 appears each sensor in the whole network is split to track this target by using the RS algorithm. With this algorithm two sub-group do not maintain their formation, and all sensor in each sub-group need the certain time to form a network.

Figures 3.25 and 3.26 also represent the results of tracking of Case 2 where the targets appear one by one and one then disappears. When target 2 appears at time $(t = 1.26)$ the results are similar with Figures 3.21 and 3.22. When target 2 disappears at time $(t = 4.95)$ sub-group 2 which are tracking this target will rejoin to sub-group 1 and continue to track

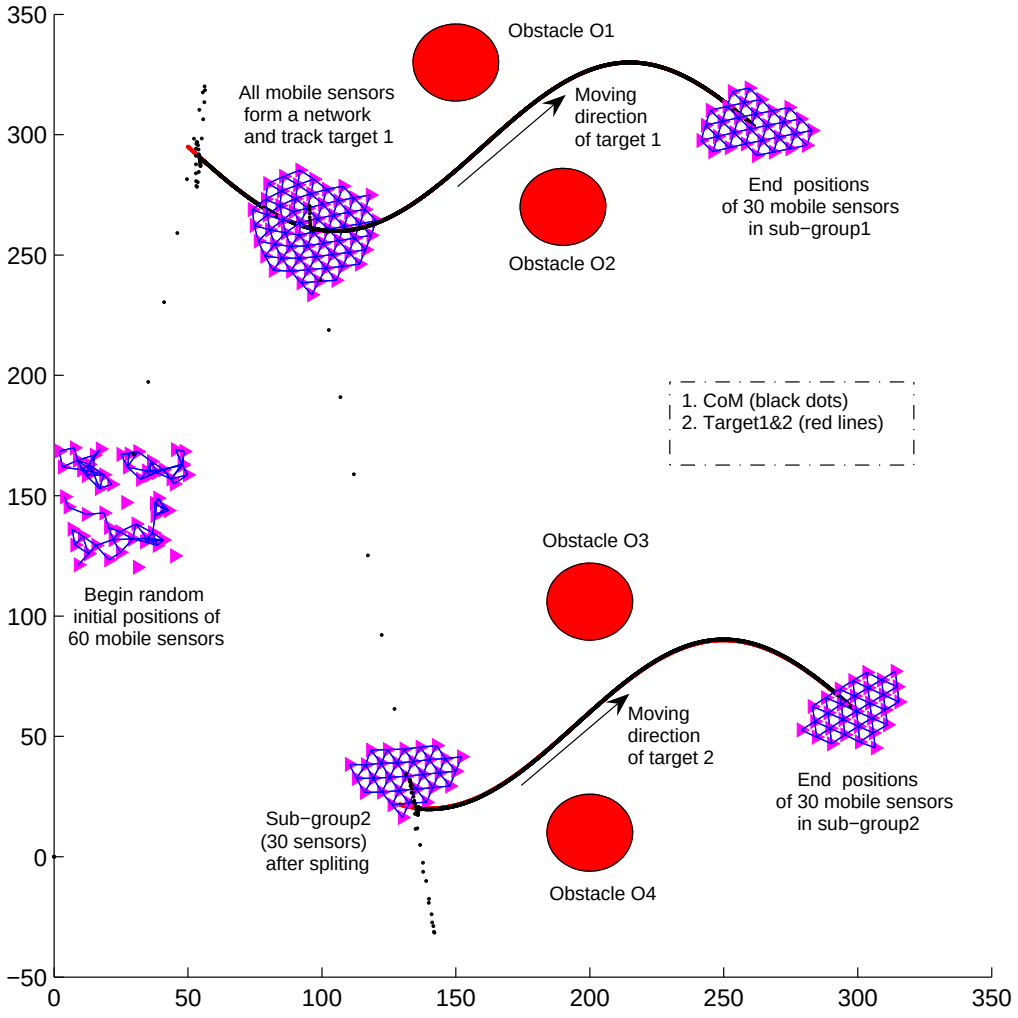


Figure 3.21: Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, forming a network at time $t = 1.26$, and decomposing into two sub-groups, respectively to track the targets moving in the sine wave trajectories. This result is obtained by using the *Multi-CoM* flocking control and SGGP algorithms.

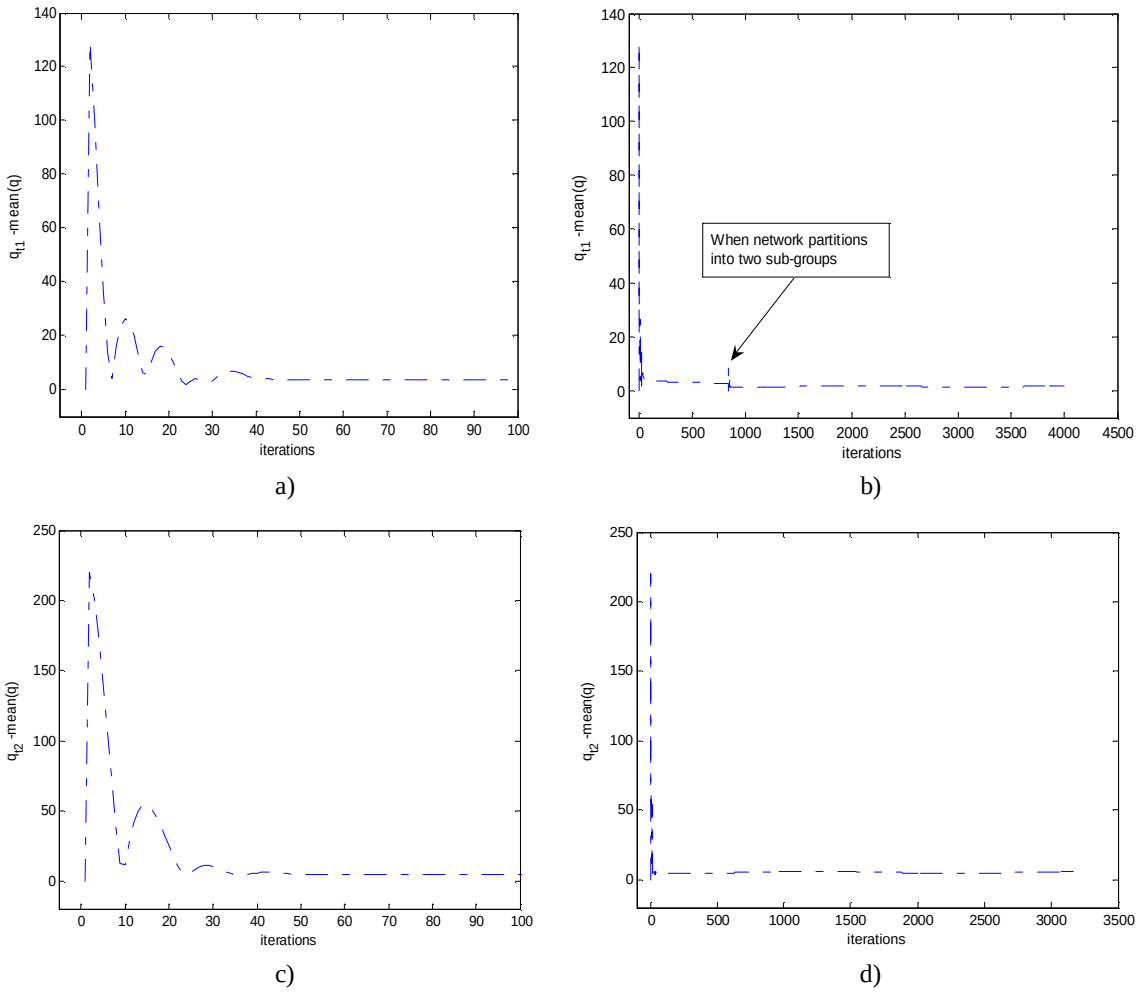


Figure 3.22: (a, c) are closer look of (b, d) at iterations from 1 to 100. (b) Position errors between the CoM of the whole network and target 1 (from iteration 1 to 839), and between the CoM of the sub-group 1 and target 1 (from iteration 840 to the end). (d) Position errors between the CoM of the sub-group 2 and target 2. This result is obtained by using the *Multi-CoM* flocking control and SGGP algorithms.

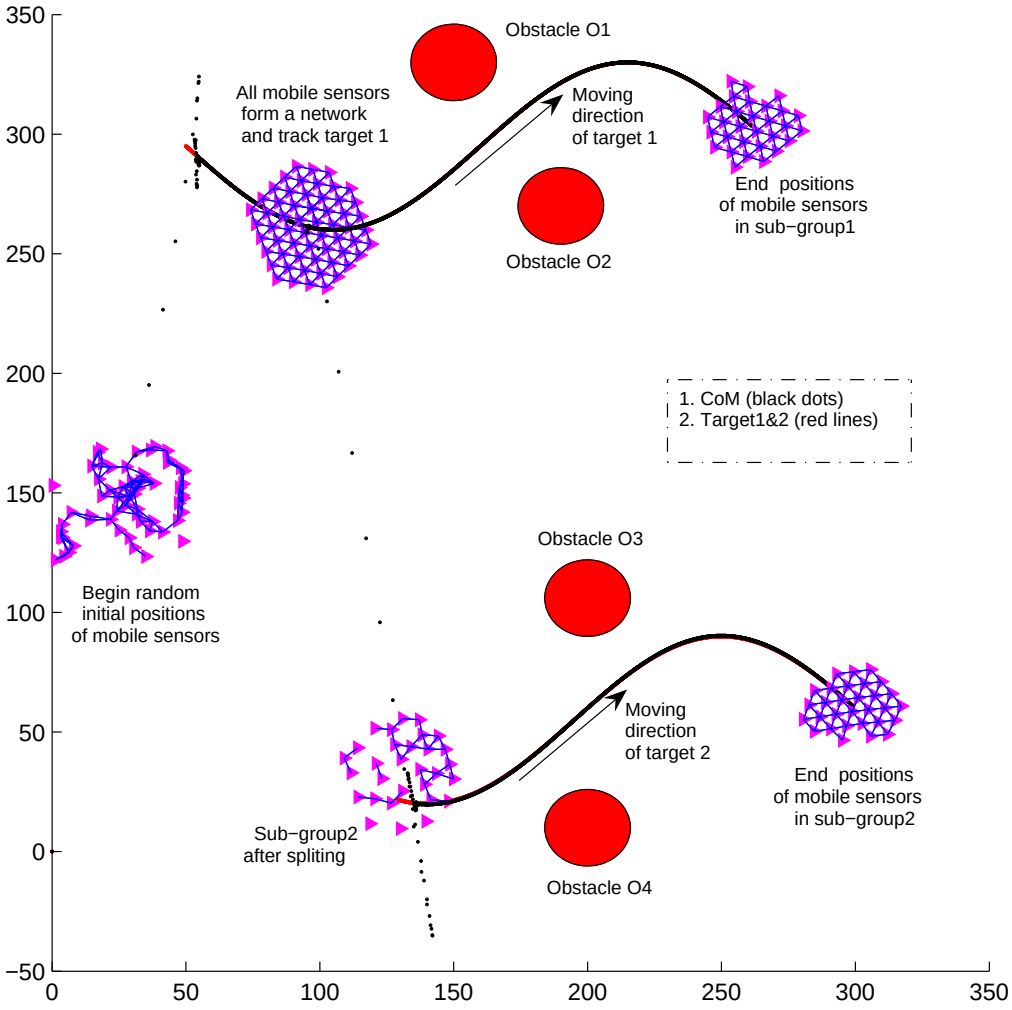


Figure 3.23: Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, forming a network at time $t = 1.26$, and decomposing into two sub-groups, respectively to track the targets moving in the sine wave trajectories. This result is obtained by using the *Multi-CoM* flocking control and RS algorithms.

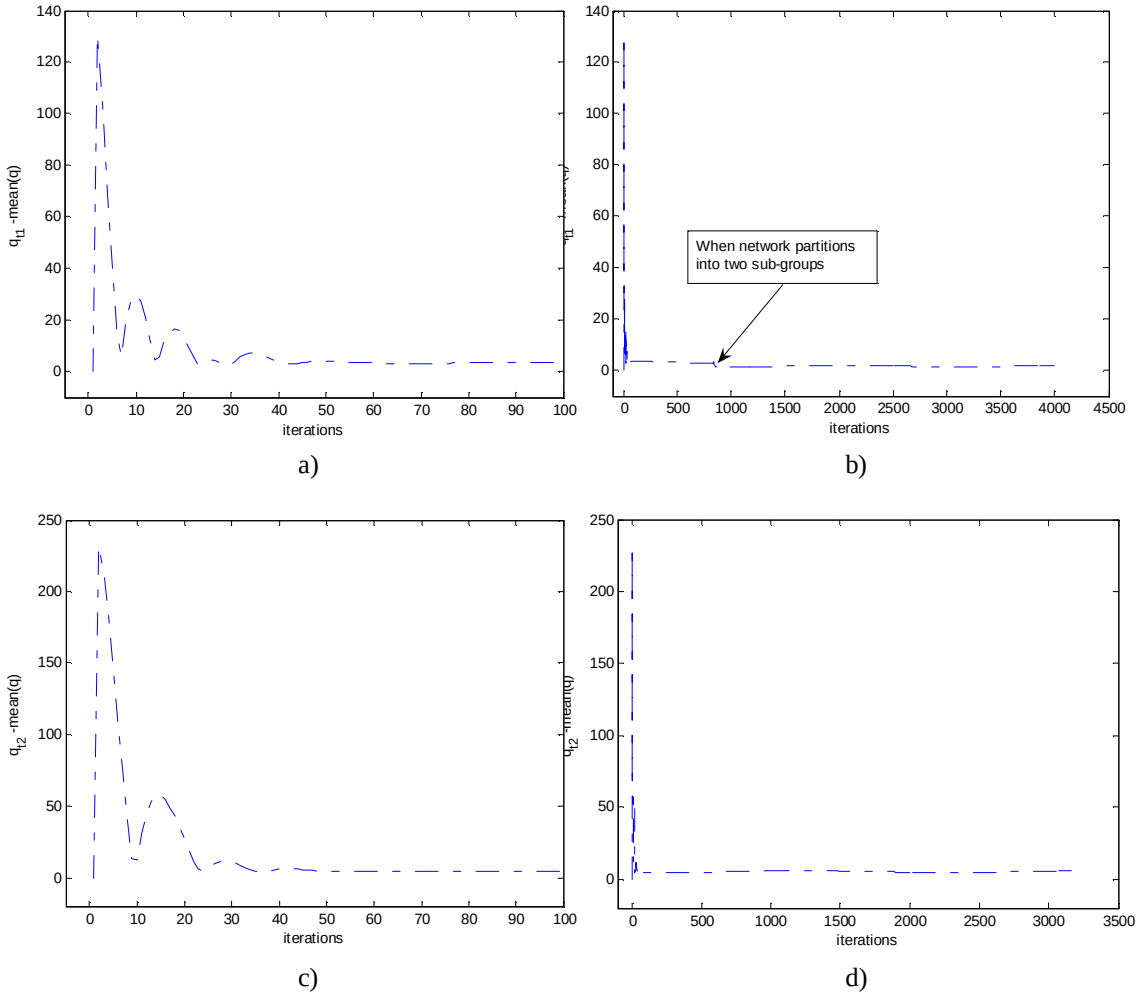


Figure 3.24: (a, c) are closer look of (b, d) at iterations from 1 to 100. (b) Position errors between the CoM of the whole network and target 1 (from iteration 1 to 839), and between the CoM of the sub-group 1 and target 1 (from iteration 840 to the end). (d) Position errors between the CoM of the sub-group 2 and target 2. This result is obtained by using the *Multi-CoM* flocking control and RS algorithms.

target 1. The result of the whole group after merging is good with small error between CoM and target 1 as shown in Figure 3.26 (from iteration 3301 to 5001, or $t \in [4.95, 7.5]$). In all the above simulation results, all sensors keep their formation (excepting in the case of the RS algorithm) and no collision occurs among them while tracking the moving target, and all sensors avoid obstacles successfully in a narrow space.

In summary, we see that the SGGP algorithm combining the flocking control with *Multi-CoM* is better than the SGGP algorithm combining the flocking control with *No-CoM* in terms of the tracking performance. Namely, in the SGGP algorithm with *No-CoM* the CoM could not converge to the target in the obstacle space, but this was not the case in the SGGP algorithm with *Multi-CoM*.

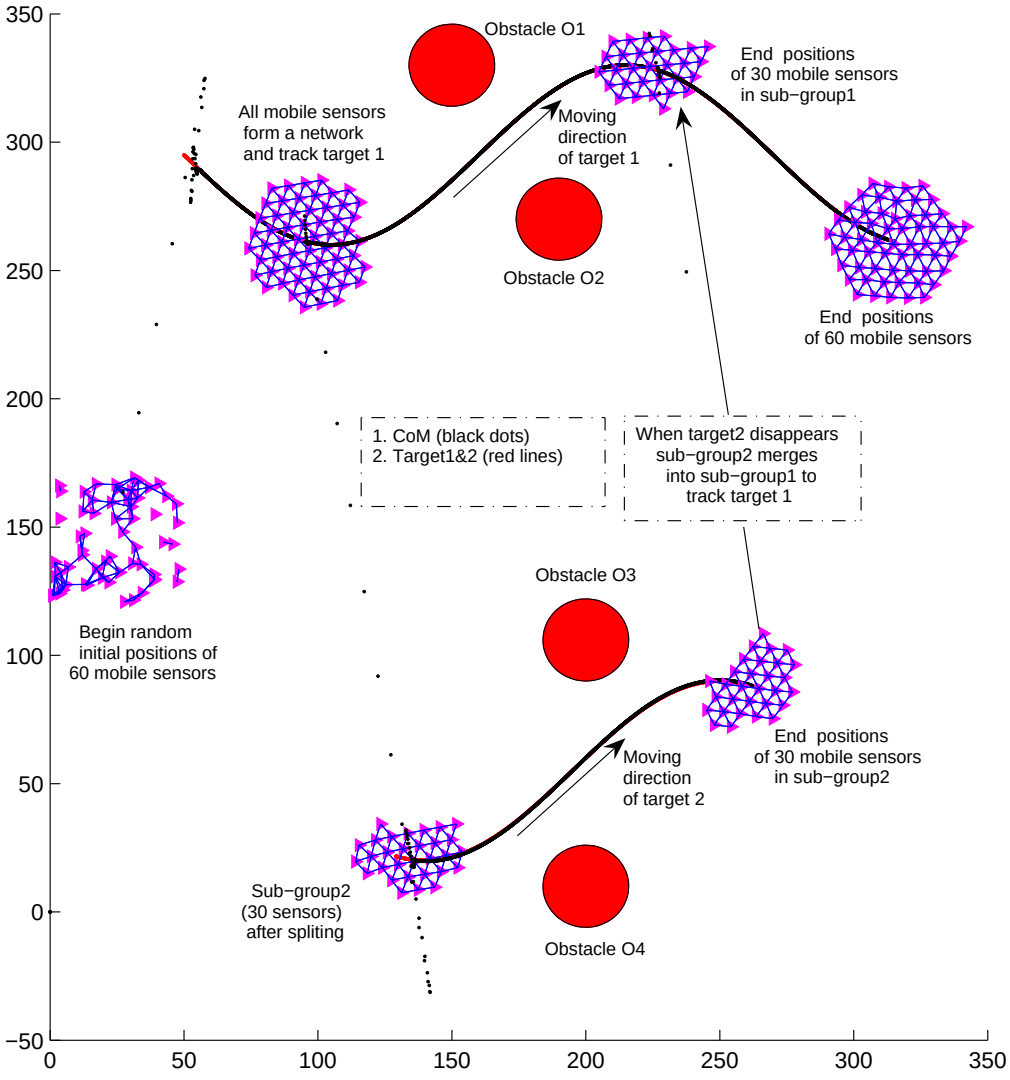


Figure 3.25: Snapshots of the mobile sensor network when the mobile sensors are at the initial positions, when the mobile sensors form a network at time $t = 1.26$, when the mobile sensors decompose into two sub-groups, and when two sub-groups merge. This result is obtained by using the *Multi-CoM* flocking control and SGGP algorithms.

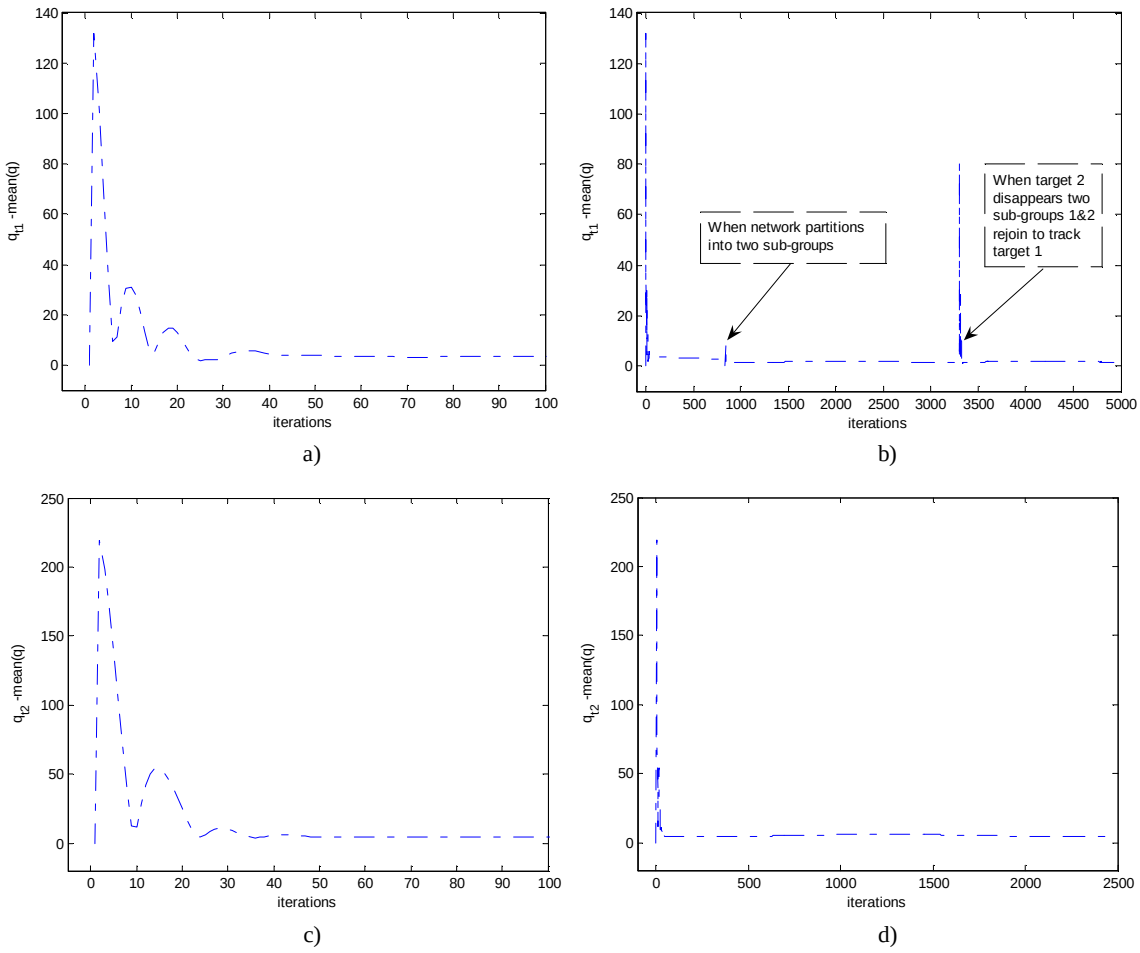


Figure 3.26: (a, c) are closer look of (b, d) at iterations from 1 to 100. (b) Position errors between the CoM of the whole network and target 1 (from iteration 1 to 839, and 3301 to the end), between the CoM of the sub-group 1 and target 1 (from iteration 840 to 3300). (d) Position errors between the CoM of the sub-group 2 and target 2. This result is obtained by using the *Multi-CoM* flocking control and SGGP algorithms.

3.4 Summary

In this chapter, we considered the behavior of a group of agents when only a subset of them have the information of the target. We proposed a decentralized flocking control algorithm to deal with the network partition and reduce the overshoot of the tracking. Our algorithm is based on considering the effect of the target tracking term and damping term. As a result, the network connectivity preservation is improved, the overshoot is eliminated, and the collision avoidance among agents is guaranteed. Both simulation and experimental results are collected to demonstrate the effectiveness of our proposed flocking control.

We studied the approach to flocking control of a mobile sensor network to track and observe a moving target in changing environments. We designed an adaptive flocking control algorithm that can cooperatively learn the network's parameters in a decentralized fashion to change the size of the network in order to maintain connectivity, formation and tracking performance when passing through obstacles. In addition, to see the benefit of the adaptive flocking algorithm we compared it with the normal flocking control algorithm, and we found that the connectivity, similar formation and tracking performance in the adaptive flocking control algorithm are better than those in the existing flocking control algorithm. The simulations and experiments on real Rovio robots verified our theoretical results.

We developed an approach to flocking control of a mobile sensor network to track and observe multiple dynamic targets. The SGGP algorithm is proposed to solve the problem of splitting/merging the sensor agents. To see the benefit of this algorithm we compared it with a random selection (RS) algorithm, and the results are promising. The maximum of the convergent distance and formation time in the SGGP algorithm is faster than that in the RS algorithm. In addition, the distance in the SGGP algorithm is shorter than that in the RS algorithm. Several experimental tests were done with two different cases of splitting and merging sensor agents to demonstrate our theoretical results.

CHAPTER 4

COOPERATIVE CONTROL BASED FLOCKING FOR MSNs IN NOISY ENVIRONMENTS

In this chapter in order to deal with noisy measurements we propose two flocking control algorithms, *Multi-CoM-Shrink* and *Multi-CoM-Cohesion*. Based on these algorithms, all agents can form a network and maintain connectivity, even with noisy measurements. We also investigate the stability and scalability of our algorithms. Simulation results are conducted to demonstrate the effectiveness of the proposed algorithms.

This chapter is organized as follows. Section 4.1 gives the motivation and problem formulation of flocking control in noisy environments. Section 4.2 presents our flocking control algorithms, *Multi-CoM-Shrink* and *Multi-CoM-Cohesion*, for tracking a moving target in noisy environments. Section 4.3 shows the main results on stability analysis of flocking control in noisy environments. Section 4.4 demonstrates the experimental results. Finally, Section 4.5 concludes this chapter.

4.1 Introduction

In real flocking control environments, noise handling is always an important issue since the noise usually causes broken network or connectivity loss. This problem exists in most of the previous work on flocking control [112, 23, 46, 17]. Namely, most of flocking control algorithms [112, 23, 46, 17] work under the following assumptions:

- Each agent can sense its own position and velocity precisely (without noises).

- Each agent can obtain its neighbor's position and velocity via sensing or communication precisely.

- Each agent can sense the target position and velocity precisely.

However, in reality these assumptions are not valid because sensing errors always exist. Motivated by these observations we will study how to design distributed flocking control algorithms which can still perform well when the measurements are affected by noises.

In this chapter we propose two new flocking control algorithms to deal with more realistic environments. To make the flocking control more applicable in real applications we consider the effect of position and velocity measurement errors of the agent itself, the agent's neighbors and the target. None of the flocking control algorithms in the above related work considers this noise issue. We propose two flocking control algorithms, *Multi-CoM-Shrink* and *Multi-CoM-Cohesion*, which are based on the extensions of the *Multi-CoM* flocking control algorithm in the previous chapters. Our algorithms allow the flocks to preserve connectivity, avoid collision, and follow the target in such noisy environments. We demonstrate that by applying our algorithms the agents can flock together in the presence of noise with better performances such as connectivity and tracking performance.

4.2 Flocking Control Algorithm in Noisy Environments

In this section we are going to design two algorithms in noisy environments. The first one is the *Multi-CoM-Shrink* flocking control algorithm. The main idea of this algorithm is to shrink the size of the network in oder to keep the connectivity. The second one is the *Multi-CoM-Cohesion* flocking control algorithm, and its main idea is based on the position and velocity cohesion feedbacks to create the strong cohesion between the agent and the network. Both algorithms are based on the *Multi-CoM* flocking control algorithm presented

in our previous chapter. The *Multi-CoM* flocking control algorithm is shown below

$$\begin{aligned} u_i &= c_1^\alpha \sum_{j\in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j\in N_i^\alpha} a_{ij}(q)(p_j - p_i) \\ &\quad -c_1^t(q_i - q_t) - c_2^t(p_i - p_t) - c_1^l(\overline{q}_i - q_t) - c_2^l(\overline{p}_i - p_t), \end{aligned} \tag{4.1}$$

here c_1^l and c_2^l are positive constants. $\overline{q}_i$ and $\overline{p}_i$ are the local average of position and velocity, respectively for each agent i defined as:

$$\begin{cases} \overline{q}_i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} q_j \\ \overline{p}_i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} p_j. \end{cases} \tag{4.2}$$

In this control algorithm, the first two terms are used to control the formation (α-lattice configuration) and to allow agents to avoid collision [23]. The terms $-c_1^t(q_i - q_t) - c_2^t(p_i - p_t)$ and $-c_1^l(\overline{q}_i - q_t) - c_2^l(\overline{p}_i - p_t)$ allow each agent and its neighbors to closely follow the target.

4.2.1 Multi-CoM-Shrink Algorithm

Assume that the estimates of the position and velocity of agent i are: $\hat{q}_i = q_i + \varepsilon_q^i$ and $\hat{p}_i = p_i + \varepsilon_p^i$, where ε_q^i and ε_p^i are the position and velocity measurement errors, respectively. Then we have:

$\hat{q}_i - \hat{q}_j = q_i - q_j + \varepsilon_q^{ij}$; $\hat{p}_i - \hat{p}_j = p_i - p_j + \varepsilon_p^{ij}$, here $\varepsilon_q^{ij} = \varepsilon_q^i - \varepsilon_q^j$ and $\varepsilon_p^{ij} = \varepsilon_p^i - \varepsilon_p^j$.

Similarly, the estimates of the position and velocity of the target are: $\hat{q}_t = q_t + \varepsilon_q^t$ and $\hat{p}_t = p_t + \varepsilon_p^t$, where ε_q^t and ε_p^t are the position and velocity measurement errors, respectively. Then we have:

$\hat{q}_i - \hat{q}_t = q_i - q_t + \varepsilon_q^{it}$; $\hat{p}_i - \hat{p}_t = p_i - p_t + \varepsilon_p^{it}$, here $\varepsilon_q^{it} = \varepsilon_q^i - \varepsilon_q^t$ and $\varepsilon_p^{it} = \varepsilon_p^i - \varepsilon_p^t$.

If all noises are bounded, one possible method to maintain connectivity in noisy environments is to shrink the size of the network. We assume that the noise ε_q^i satisfies $\|\varepsilon_q^i\| \leq r_w$ as shown in Figure 4.1.

Let us denote $d_a = \|q_i - q_j\|$ to be the actual distance between agent i and agent j. Then

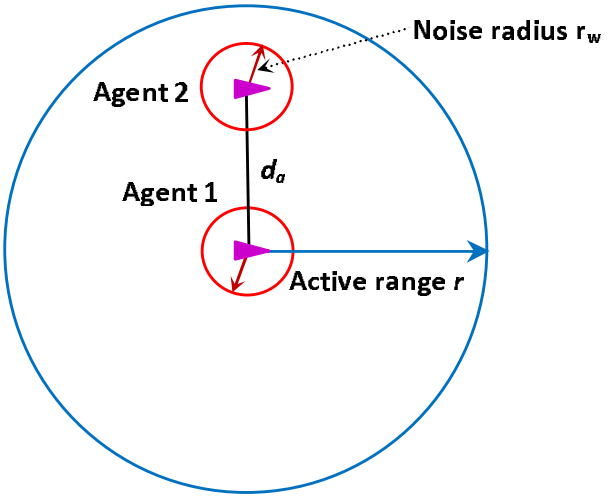


Figure 4.1: Agent 2 is considered as a neighbor of agent 1 because the estimated distance $\hat{d}_a$ is less than the active range r.

to maintain the connectivity and no collision among agents we need

$$0 < d_a \leq r. \tag{4.3}$$

Denote $\hat{d}_a$ to be the estimate of the actual distance d_a, then we have

$$\hat{d}_a = \|\hat{q}_i - \hat{q}_j\| \leq \|q_i - q_j\| + \|\varepsilon_q^{ij}\|. \tag{4.4}$$

Since $\|\varepsilon_q^i\| \leq r_w$ we have $\|\varepsilon_q^{ij}\| \leq 2r_w$, and we obtain

$$\|q_i - q_j\| - 2r_w \leq \hat{d}_a \leq \|q_i - q_j\| + 2r_w. \tag{4.5}$$

With $\|q_i - q_j\| = d_a$ we have

$$d_a - 2r_w \leq \hat{d}_a \leq d_a + 2r_w, \tag{4.6}$$

or,

$$\hat{d}_a - 2r_w \leq d_a \leq \hat{d}_a + 2r_w. \tag{4.7}$$

Since the control algorithm (2.38) guarantees that $\hat{d}_a$ converges to the desired distance d. Then from (4.7) we obtain

$$d - 2r_w \leq d_a \leq d + 2r_w. \tag{4.8}$$

From (4.3) and (4.8) we should have

$$\begin{cases} d - 2r_w > 0 \\ d + 2r_w \leq r. \end{cases} \tag{4.9}$$

Hence from (4.9) we obtain d to be

$$2r_w < d \leq r - 2r_w. \tag{4.10}$$

Equation (4.10) shows that we need to design the distance d within the range $(2r_w,\ r-2r_w]$ to maintain connectivity and no collision among agents. However if we select d to be smaller than $r-2r_w$ then each agent will have more neighbors than necessary. Hence, we choose $d = r - 2r_w$.

Now, from (2.24) we obtain d^{α}_{new} as

$$d^{\alpha}_{new} = \|d\|_{\sigma} = \frac{1}{\varepsilon}[\sqrt{1+\varepsilon(r-2r_w)^2} - 1]. \tag{4.11}$$

From (2.25) we obtain a new action function $\phi^{new}_{\alpha}(\|\hat{q}_j - \hat{q}_i\|_{\sigma})$ as follows:

$$\phi^{new}_{\alpha}(\|\hat{q}_j - \hat{q}_i\|_{\sigma}) = \rho_h(\|\hat{q}_j - \hat{q}_i\|_{\sigma}/r_{\alpha})\phi(\|\hat{q}_j - \hat{q}_i\|_{\sigma} - d^{\alpha}_{new}). \tag{4.12}$$

From (4.2) we have the local average of position and velocity for each agent i, $\hat{\overline{q}}_i$ and $\hat{\overline{p}}_i$ with noise computed as

$$\begin{cases} \hat{\overline{q}}_i = \frac{1}{|N^{\alpha}_i \cup \{i\}|} \sum_{j=1}^{|N^{\alpha}_i \cup \{i\}|} \hat{q}_j \\ \hat{\overline{p}}_i = \frac{1}{|N^{\alpha}_i \cup \{i\}|} \sum_{j=1}^{|N^{\alpha}_i \cup \{i\}|} \hat{p}_j, \end{cases} \tag{4.13}$$

From (2.28) and (2.29) we obtain $\hat{n}_{ij}$ and $\hat{a}_{ij}(q)$ as

$$\hat{n}_{ij} = (\hat{q}_j - \hat{q}_i)/\sqrt{1+\varepsilon\|\hat{q}_j - \hat{q}_i\|^2} \tag{4.14}$$

$$\hat{a}_{ij}(q) = \begin{cases} \rho_h(\|\hat{q}_j - \hat{q}_i\|_{\sigma}/r_{\alpha}), & if \quad j \neq i \\ 0, & if \quad j = i, \end{cases} \tag{4.15}$$

Now, we propose a *Multi-CoM-Shrink* algorithm with d^{α}_{new} as

$$\begin{aligned} u_i \;=\; & c^{\alpha}_1 \sum_{j \in N^{\alpha}_i} \phi^{new}_{\alpha}(\|\hat{q}_j - \hat{q}_i\|_{\sigma})\hat{n}_{ij} + c^{\alpha}_2 \sum_{j \in N^{\alpha}_i} \hat{a}_{ij}(q)(\hat{p}_j - \hat{p}_i) \\ & -c^{t}_1(\hat{q}_i - \hat{q}_t) - c^{t}_2(\hat{p}_i - \hat{p}_t) - c^{l}_1(\hat{\overline{q}}_i - \hat{q}_t) - c^{l}_2(\hat{\overline{p}}_i - \hat{p}_t). \end{aligned} \tag{4.16}$$

4.2.2 Multi-CoM-Cohesion Algorithm

In this subsection we describe the *Multi-CoM-Cohesion* algorithm. The main idea of the *Multi-CoM-Cohesion* algorithm is that each agent should have a strong cohesion to the network so that the connectivity is maintained. In order to do that we introduce local position and velocity cohesion feedbacks to each agent.

Before presenting the algorithm, we have the following definitions:

$d_{il} = q_i - \overline{q}_i$ is the relative distance between node i and its local average of position;

$v_{il} = p_i - \overline{p}_i$ is the relative velocity between node i and its local average of velocity;

However, because agent i senses its own position and velocity with noise, hence the estimates $\hat{d}_{il}$ and $\hat{v}_{il}$ are also corrupted by noise $(\varepsilon_d^i, \varepsilon_v^i)$ as:

$$\begin{cases} \hat{d}_{il} = \hat{q}_i - \hat{\overline{q}}_i = q_i + \varepsilon_q^i - (\overline{q}_i + \overline{\varepsilon}_q^i) = d_{il} + \varepsilon_d^i \\ \hat{v}_{il} = \hat{p}_i - \hat{\overline{p}}_i = p_i + \varepsilon_p^i - (\overline{p}_i + \overline{\varepsilon}_p^i) = v_{il} + \varepsilon_v^i, \end{cases} \tag{4.17}$$

here $\varepsilon_d^i = \varepsilon_q^i - \overline{\varepsilon}_q^i$ with $\overline{\varepsilon}_q^i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} \varepsilon_q^i$,
and $\varepsilon_v^i = \varepsilon_p^i - \overline{\varepsilon}_p^i$ with $\overline{\varepsilon}_p^i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} \varepsilon_p^i$.

Based on the above definitions, we design a distributed flocking control law, *Multi-CoM-Cohesion* , in noisy environments as:

$$\begin{aligned} u_i \;=\; & c_1^\alpha \sum_{j \in N_i^\alpha} \phi_\alpha(\|\hat{q}_j - \hat{q}_i\|_\sigma)\hat{n}_{ij} + c_2^\alpha \sum_{j \in N_i^\alpha} \hat{a}_{ij}(q)(\hat{p}_j - \hat{p}_i) \\ & - c_{pos}\hat{d}_{il} - c_{ve}\hat{v}_{il} \\ & - c_1^t(\hat{q}_i - \hat{q}_t) - c_2^t(\hat{p}_i - \hat{p}_t) - c_1^l(\hat{\overline{q}}_i - \hat{q}_t) - c_2^l(\hat{\overline{p}}_i - \hat{p}_t), \end{aligned} \tag{4.18}$$

here $\hat{d}_{il}$, $\hat{v}_{il}$ are the estimates of d_{il} and v_{il}, respectively, and c_{pos} and c_{ve} are positive constants. The terms $-c_{pos}\hat{d}_{il}$ and $-c_{ve}\hat{v}_{il}$ are called local position and velocity cohesion feedbacks, respectively. The role of these negative feedbacks is to maintain position and velocity cohesions. This means that each agent tries to stay close to the local average of position and minimize the velocity mismatch between its velocity and the local average of velocity in noisy environments.

In this algorithm, to make it simpler in the stability analysis provided later we dropped the obstacle avoidance term. However, in real applications, to allow each agent to avoid both static and dynamic obstacles we only need to add the second component (2.30) to the control algorithm (4.18). In general, this component does not affect the properties of the global stability of the whole system.

4.3 Stability Analysis

Before analyzing the stability of the flocking control algorithm, *Multi-CoM-Cohesion*, we build the error dynamic model of the flocking system in noisy environments in the next subsection.

Error Dynamic Model

To study the stability properties, we have the error dynamics of the system given as follows:

$$\begin{cases} \dot{d}_{ig} = v_{ig} \\ \dot{v}_{ig} = u_i - \frac{1}{n}\sum_{j=1}^{n} u_j = u_i - \overline{u}, \quad i = 1,2,...,n. \end{cases} \tag{4.19}$$

here $\overline{u} = \frac{1}{n}\sum_{j=1}^{n} u_j$.

We have following definitions:

$d_{ig} = q_i - \overline{q}$ is the relative distance between node i and its global average of position;

$v_{ig} = p_i - \overline{p}$ is the relative velocity between node i and its global average of velocity;

Then we have the following relations:

$$\begin{aligned} d_{il} &= q_i - \overline{q}_i = d_{ig} + \overline{q} - \frac{1}{|N_i^{\alpha} \cup \{i\}|} \sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|} q_j \\ &= d_{ig} + \overline{q} - \frac{1}{|N_i^{\alpha} \cup \{i\}|} \sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|} (d_{jg} + \overline{q}) = d_{ig} - \frac{1}{|N_i^{\alpha} \cup \{i\}|} \sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|} d_{jg}. \end{aligned} \tag{4.20}$$

Then similar to d_{il}, v_{il} is obtained as follows:

$$v_{il} = v_{ig} - \frac{1}{|N_i^{\alpha} \cup \{i\}|} \sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|} v_{jg}. \tag{4.21}$$

The estimates of the local average of position and velocity, respectively in (4.13) is rewritten as

$$\hat{\bar{q}}_i = q_i - d_{ig} + \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} d_{jg} + \bar{\varepsilon}_q^i. \tag{4.22}$$

$$\hat{\bar{p}}_i = p_i - v_{ig} + \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} v_{jg} + \bar{\varepsilon}_p^i. \tag{4.23}$$

Now, we can rewrite the control law (4.18) with considering (4.17), (4.22) and (4.23):

$$\begin{aligned} u_i \;=\; & c_1^\alpha \sum_{j\in N_i^\alpha} \phi_\alpha(\|\hat{q}_j - \hat{q}_i\|_\sigma)\hat{n}_{ij} + c_2^\alpha \sum_{j\in N_i^\alpha} \hat{a}_{ij}(q)(\hat{p}_j - \hat{p}_i) \\ & + (c_1^l - c_{pos})(d_{ig} - \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} d_{jg}) + (c_2^l - c_{ve})(v_{ig} - \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} v_{jg}) \\ & - (c_1^t + c_1^l)(q_i - q_t) - (c_2^t + c_2^l)(p_i - p_t) - c_{pos}\varepsilon_d^i - c_{ve}\varepsilon_v^i - c_1^l\bar{\varepsilon}_q^i - c_2^l\bar{\varepsilon}_p^i \\ & - (c_1^t + c_1^l)\varepsilon_q^{it} - (c_2^t + c_2^l)\varepsilon_p^{it} \end{aligned} \tag{4.24}$$

The average of control law for composite system is

$$\begin{aligned} \bar{u} \;=\; & \frac{c_1^\alpha}{n} \sum_{i=1}^n [\sum_{j\in N_i^\alpha} \phi_\alpha(\|\hat{q}_j - \hat{q}_i\|_\sigma)\hat{n}_{ij}] + \frac{c_2^\alpha}{n} \sum_{i=1}^n [\sum_{j\in N_i^\alpha} \hat{a}_{ij}(q)(\hat{p}_j - \hat{p}_i)] \\ & + (\frac{c_1^l - c_{pos}}{n}) \sum_{i=1}^n (d_{ig} - \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} d_{jg}) \\ & + (\frac{c_2^l - c_{ve}}{n}) \sum_{i=1}^n (v_{ig} - \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} v_{jg}) \\ & - (\frac{c_1^t + c_1^l}{n}) \sum_{i=1}^n (q_i - q_t) - (\frac{c_2^t + c_2^l}{n}) \sum_{i=1}^n (p_i - p_t) \\ & - \frac{1}{n} \sum_{i=1}^n [c_{pos}\varepsilon_d^i + c_{ve}\varepsilon_v^i + c_1^l\bar{\varepsilon}_q^i + c_2^l\bar{\varepsilon}_p^i + (c_1^t + c_1^l)\varepsilon_q^{it} + (c_2^t + c_2^l)\varepsilon_p^{it}] \end{aligned} \tag{4.25}$$

Substitute u_i in (4.24) and $\overline{u}$ in (4.25) into (4.19) we obtain:

$$\begin{aligned}
\dot{v}_{ig} &= c_1^{\alpha}\sum_{j\in N_i^{\alpha}}\phi_{\alpha}(\|\hat{q}_j-\hat{q}_i\|_{\sigma})\hat{n}_{ij}-\frac{c_1^{\alpha}}{n}\sum_{i=1}^{n}[\sum_{j\in N_i^{\alpha}}\phi_{\alpha}(\|\hat{q}_j-\hat{q}_i\|_{\sigma})\hat{n}_{ij}] \\
&\quad +c_2^{\alpha}\sum_{j\in N_i^{\alpha}}\hat{a}_{ij}(q)(\hat{p}_j-\hat{p}_i)-\frac{c_2^{\alpha}}{n}\sum_{i=1}^{n}[\sum_{j\in N_i^{\alpha}}\hat{a}_{ij}(q)(\hat{p}_j-\hat{p}_i)] \\
&\quad -(\frac{c_1^l-c_{pos}}{|N_i^{\alpha}\cup\{i\}|})\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}d_{jg}-(\frac{c_2^l-c_{ve}}{|N_i^{\alpha}\cup\{i\}|})\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}v_{jg} \\
&\quad -(\frac{c_1^l-c_{pos}}{n})\sum_{i=1}^{n}(d_{ig}-\frac{1}{|N_i^{\alpha}\cup\{i\}|}\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}d_{jg}) \\
&\quad -(\frac{c_2^l-c_{ve}}{n})\sum_{i=1}^{n}(v_{ig}-\frac{1}{|N_i^{\alpha}\cup\{i\}|}\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}v_{jg}) \\
&\quad -(c_{pos}-c_1^l)d_{ig}-(c_{ve}-c_2^l)v_{ig}-(c_1^t+c_1^l)d_{ig}-(c_2^t+c_2^l)v_{ig} \\
&\quad -c_{pos}\varepsilon_d^i-c_{ve}\varepsilon_v^i-c_1^l\overline{\varepsilon}_q^i-c_2^l\overline{\varepsilon}_p^i-(c_1^t+c_1^l)\varepsilon_q^{it}-(c_2^t+c_2^l)\varepsilon_p^{it} \\
&\quad +\frac{1}{n}\sum_{i=1}^{n}[c_{pos}\varepsilon_d^i+c_{ve}\varepsilon_v^i+c_1^l\overline{\varepsilon}_q^i+c_2^l\overline{\varepsilon}_p^i+(c_1^t+c_1^l)\varepsilon_q^{it}+(c_2^t+c_2^l)\varepsilon_p^{it}] \\
&= -(c_1^t+c_{pos})d_{ig}-(c_2^t+c_{ve})v_{ig}+\Phi_i+\Omega_i(V)+\zeta_i, \qquad (4.26)
\end{aligned}$$

where

$$\begin{aligned}
\Phi_i &= c_1^{\alpha}\sum_{j\in N_i^{\alpha}}\phi_{\alpha}(\|\hat{q}_j-\hat{q}_i\|_{\sigma})\hat{n}_{ij}-\frac{c_1^{\alpha}}{n}\sum_{i=1}^{n}[\sum_{j\in N_i^{\alpha}}\phi_{\alpha}(\|\hat{q}_j-\hat{q}_i\|_{\sigma})\hat{n}_{ij}] \\
&\quad +c_2^{\alpha}\sum_{j\in N_i^{\alpha}}\hat{a}_{ij}(q)(\hat{p}_j-p_i)-\frac{c_2^{\alpha}}{n}\sum_{i=1}^{n}[\sum_{j\in N_i^{\alpha}}\hat{a}_{ij}(q)(\hat{p}_j-p_i)];
\end{aligned}$$

$$\begin{aligned}
\Omega_i(V) &= -(\frac{c_1^l-c_{pos}}{|N_i^{\alpha}\cup\{i\}|})\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}d_{jg}-(\frac{c_2^l-c_{ve}}{|N_i^{\alpha}\cup\{i\}|})\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}v_{jg} \\
&\quad -(\frac{c_1^l-c_{pos}}{n})\sum_{i=1}^{n}(d_{ig}-\frac{1}{|N_i^{\alpha}\cup\{i\}|}\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}d_{jg}) \\
&\quad -(\frac{c_2^l-c_{ve}}{n})\sum_{i=1}^{n}(v_{ig}-\frac{1}{|N_i^{\alpha}\cup\{i\}|}\sum_{j=1}^{|N_i^{\alpha}\cup\{i\}|}v_{jg});
\end{aligned}$$

$$\begin{aligned}
\zeta_i &= \frac{1}{n}\sum_{i=1}^{n}[c_{pos}\varepsilon_d^i+c_{ve}\varepsilon_v^i+c_1^l\overline{\varepsilon}_p^i+c_2^l\overline{\varepsilon}_p^i+(c_1^t+c_1^l)\varepsilon_q^{it}+(c_2^t+c_2^l)\varepsilon_p^{it}] \\
&\quad -[c_{pos}\varepsilon_d^i+c_{ve}\varepsilon_v^i+c_1^l\overline{\varepsilon}_q^i+c_2^l\overline{\varepsilon}_p^i+(c_1^t+c_1^l)\varepsilon_q^{it}+(c_2^t+c_2^l)\varepsilon_p^{it}]
\end{aligned}$$

here, we define $V_i = [d_{ig}\ v_{ig}]^T$ and $V = [V_1,\ V_2, ...,\ V_n]^T$.

Rewrite (4.26) in state space representation

$$\begin{bmatrix} \dot{d}_{ig} \\ \dot{v}_{ig} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -k_1 I & -k_2 I \end{bmatrix} \begin{bmatrix} d_{ig} \\ v_{ig} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} (\Phi_i + \Omega_i(V) + \zeta_i), \tag{4.27}$$

here $k_1 = (c_1^t + c_{pos})$, $k_2 = (c_2^t + c_{ve})$, and I is an m x m identity matrix.

Then we can rewrite (4.27) as

$$\dot{V}_i = \begin{bmatrix} 0 & I \\ -k_1 I & -k_2 I \end{bmatrix} V_i + \begin{bmatrix} 0 \\ I \end{bmatrix} (\Phi_i + \Omega_i(V) + \zeta_i) \tag{4.28}$$

Let the matrix $A_i = \begin{bmatrix} 0 & I \\ -k_1 I & -k_2 I \end{bmatrix}$, then we have the characteristic equation as:

$$det(\lambda I - A_i) = (\lambda^2 + k_2\lambda + k_1)^m = 0. \tag{4.29}$$

Since $k_1 > 0$, $k_2 > 0$, and if $k_2 < 2\sqrt{k_1}$ then all roots of the characteristic equation (4.29) have negative real parts ($Re(\lambda_i) < 0$).

Stability Analysis of the *Multi-CoM-Cohesion* algorithm

In this subsection we will analyze the stability of the flocking control algorithm, *Multi-CoM-Cohesion*, in noisy environments based on the Lyapunov approach.

We assume that the errors of sensing position and velocity have linear relationship with the magnitude of the state of the error system. That is because two agents are far away from each other, the sensing errors will usually increase. Hence, we have

$$\begin{cases} \|\varepsilon_d^i(t)\| \le c_{ed_1}^i \|V_i(t)\| + c_{ed_2}^i \\ \|\varepsilon_v^i(t)\| \le c_{ev_1}^i \|V_i(t)\| + c_{ev_2}^i, \quad i = 1, 2, ..., n. \end{cases} \tag{4.30}$$

We also assume that the noise ε_q^{it} and ε_p^{it} on the target tracking terms (negative feedbacks) are bounded as

$$\begin{cases} \|\varepsilon_q^{it}(t)\| \le c_{eq}^i \\ \|\varepsilon_p^{it}(t)\| \le c_{ep}^i, \quad i = 1, 2, ..., n, \end{cases} \tag{4.31}$$

and the noise $\overline{\varepsilon}_q^i$ and $\overline{\varepsilon}_p^i$ on the estimates of local average of position and velocity are bounded as

$$\begin{cases} \|\overline{\varepsilon}_q^i(t)\| \leq \overline{c}_{eq}^i \\ \|\overline{\varepsilon}_p^i(t)\| \leq \overline{c}_{ep}^i, \quad \overline{i} = 1,2,...,n. \end{cases} \tag{4.32}$$

here $\overline{c}_{eq}^i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} c_{eq}^i$, and $\overline{c}_{ep}^i = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} c_{ep}^i$.

Theorem 3. Consider a system of n mobile agents with dynamics (2.18) and controlled by (4.18), and all noise are bounded by (4.30), (4.31) and (4.32) . Let

$$c_{pv}^1 = \frac{(c_{pos}+1)^2 + c_{ve}^2}{2c_{pos}c_{ve}} + \sqrt{(\frac{c_{pos}+c_{ve}^2-1}{2c_{pos}c_{ve}})^2 + \frac{1}{c_{pos}^2}},$$

and if

$$c_{pos}c_{ed_1}^i + c_{ve}c_{ev_1}^i \leq \frac{1}{c_{pv}^1},$$

and the parameters are such that

$$\sum_{j=1}^{m} \frac{2c_{pv}^1[\sqrt{(c_1^l - c_{pos})^2 + (c_2^l - c_{ve})^2} - \frac{1}{n}(c_{pos}c_{ed_1}^i + c_{ve}c_{ev_1}^i)]}{(1-\varepsilon_i)[1 - c_{pv}^1(c_{pos}c_{ed_1}^i + c_{ve}c_{ev_1}^i)]} < 1,$$

here $0 < \varepsilon_i < 1$ for $\forall i$, then the trajectories of (4.28) are bounded.

Proof:

To study the stability of the error dynamics (4.28), one possible choice is to choose the Lyapunov function for each agent as

$$L_i(V_i) = V_i^T P V_i, \tag{4.33}$$

here $P = P^T$ is a $2m$ x $2m$ positive-definite matrix ($P > 0$). Then, the Lyapunov function for the composite system is

$$L(V) = \sum_{i=1}^{n} V_i^T P V_i.$$

From (4.33) we have

$$\dot{L}_i(V_i) = V_i^T P \dot{V}_i + \dot{V}_i^T P V_i. \tag{4.34}$$

Then, substitute $\dot{V}_i$ in (4.28) into (4.34) we obtain

$$\begin{aligned}\dot{L}_i(V_i) &= V_i^T(PA_i + A_i^T P)V_i + 2V_i^T PB(\Phi_i + \Omega_i(V) + \zeta_i) \\ &= -V_i^T CV_i + 2V_i^T PB(\Phi_i + \Omega_i(V) + \zeta_i),\end{aligned}$$

here $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$, and $C = -(PA_i + A_i^T P)$.

The remaining part of this proof is to show $\dot{L}_i(V_i) < 0$. The detailed proof of $\dot{L}_i(V_i) < 0$ is similar to that in the reference [113].

4.4 Experimental Results

In this section we are going to test our proposed algorithms, adaptive flocking control (3.12), *Multi-CoM-Shrink* (4.16), and *Multi-CoM-Cohesion* (4.18). Then we compare our algorithms with the existing one (2.38), called *No-CoM* flocking control algorithm, in terms of network connectivity, formation and tracking performance. First we discuss how to evaluate the connectivity of the network in the next subsection.

4.4.1 Parameter Setup

The parameters used in this simulation are specified as follows:

- Parameters of flocking: we use 50 agents which are randomly distributed in the square area of 120 x 120 size; and other parameter are $a = b = 5$; the active range $r = 19$; $\varepsilon = 0.1$ for the σ-norm; h = 0.2 for the bump functions ($\phi_\alpha^{new}(z), \phi_\alpha(z)$); h = 0.9 for the bump function ($\phi_\beta(z)$). The desired distance for the flocking control algorithms, *No-CoM* (2.38) and *Multi-CoM-Cohesion*, $d = 16$. For the *Multi-CoM-Shrink* flocking control algorithm, $r_w = 3.4$, hence $d = r - 2r_w = 19 - 2 \times 3.4 = 12.2$.

- Parameters of target movement:

Case 1: The target moves in a sine wave trajectory: $q_t = [50 + 50t,\ 295 - 50sin(t)]^T$ with $0 \leq t \leq 6$.

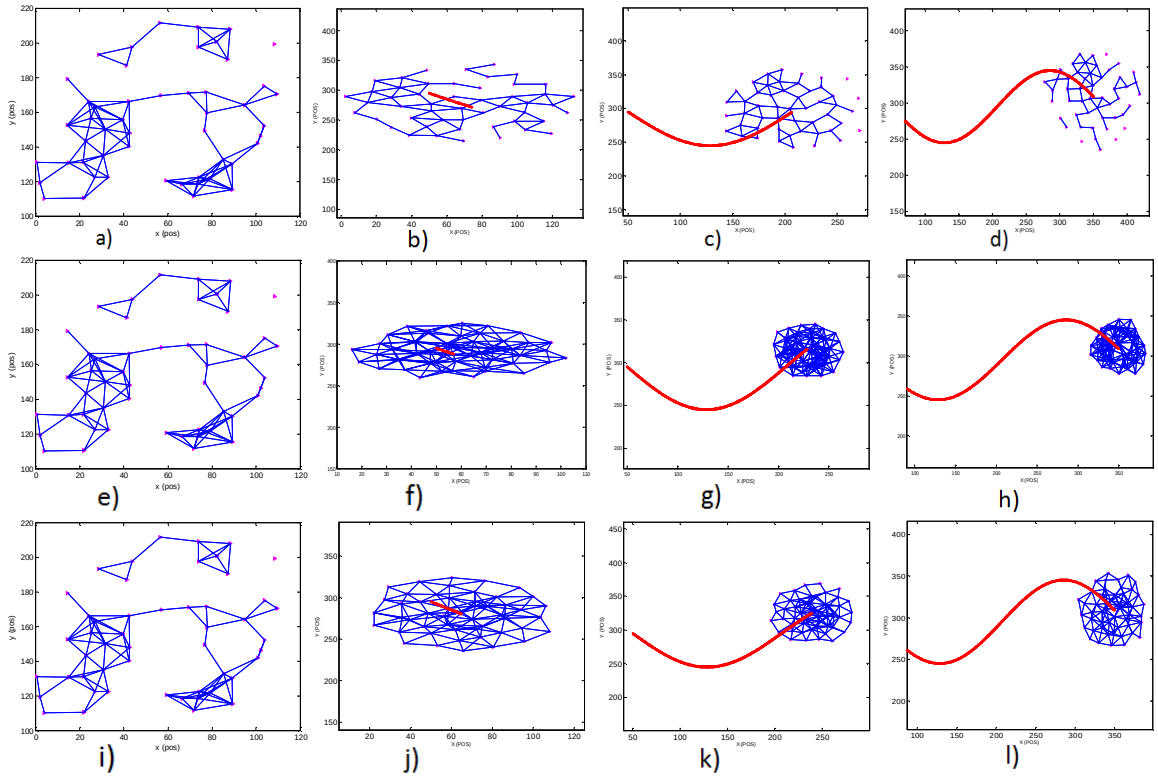


Figure 4.2: Snapshots of agents when they are randomly distributed (a, e, i), and when they form a network and track a target (red/dark line) moving in a sine wave trajectory (b, c, d; f, g, h; j, k, l), where (a, b, c, d) are for the *No-CoM* flocking control algorithm (2.38), (e, f, g, h) are for the *Multi-CoM-Shrink* flocking control algorithm, and (i, j, k, l) are for the *Multi-CoM-Cohesion* flocking control algorithm.

Case 2: The target moves in a circle trajectory: $q_t = [310 - 160cos(t),\ 255 + 160sin(t)]^T$ with $0 \leq t \leq 4$.

- The noise used in the simulation is Gaussian with zero mean and a variance of 1.

Figures 4.2 and 4.3 show the results of of the moving target (red/dark line) tracking in the sine wave and circle trajectories, respectively in noisy environments for three algorithms, *No-CoM* (2.38), *Multi-CoM-Shrink* and *Multi-CoM-Cohesion*. Especially, Figures 4.2(a, b, c, d) and 4.3(a, b, c, d) are for the *No-CoM* algorithm (2.38). Figures 4.2(e, f, g, h) and 4.3(e, f, g, h) are for the proposed flocking control algorithm *Multi-CoM-Shrink*. Figures 4.2(i, j, k, l) and 4.3(i, j, k, l) are for the proposed flocking control algorithm

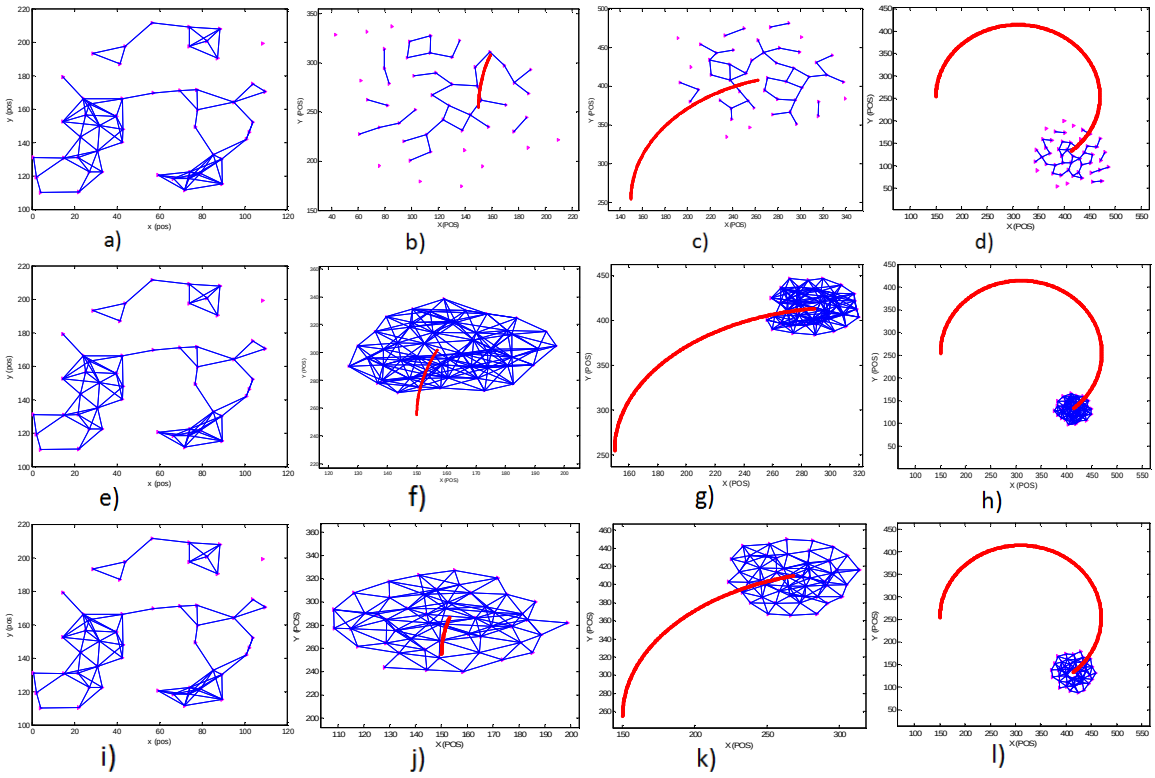


Figure 4.3: Snapshots of agents when they are randomly distributed (a, e, i), and when they form a network and track a target (red/dark line) moving in a circle trajectory (b, c, d; f, g, h; j, k, l), where (a, b, c, d) are for the *No-CoM* flocking control algorithm (2.38), (e, f, g, h) are for the *Multi-CoM-Shrink* flocking control algorithm, and (i, j, k, l) are for the *Multi-CoM-Cohesion* flocking control algorithm.

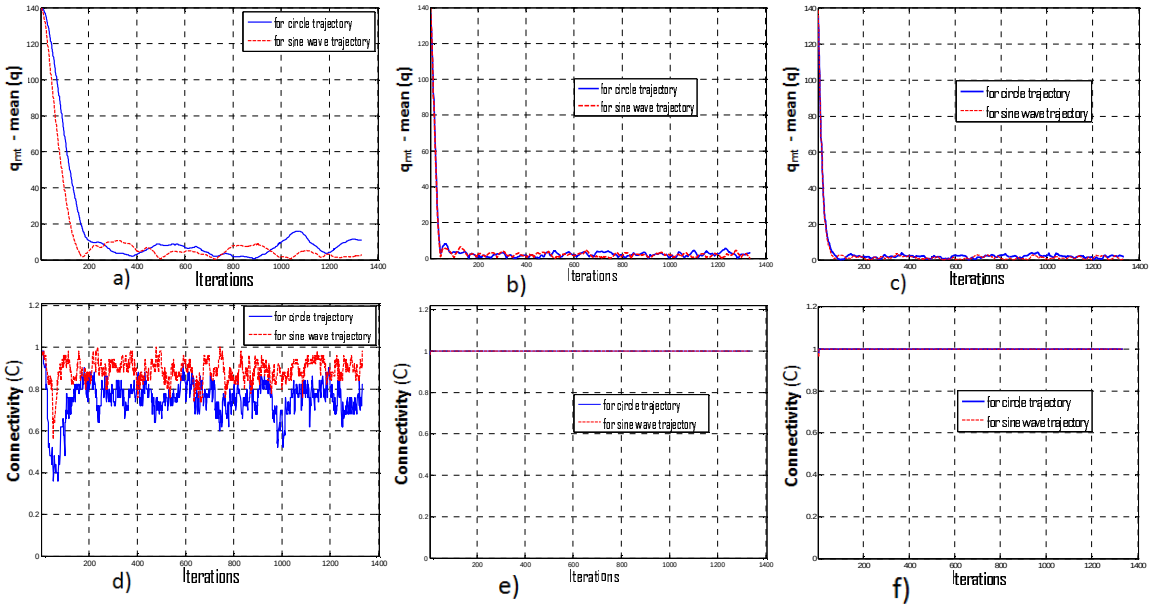


Figure 4.4: The tracking performance results (error between the CoM and target positions): (a) is for the *No-CoM* flocking control algorithm (2.38), (b) is for the *Multi-CoM-Shrink* flocking control algorithm, and (c) is for the *Multi-CoM-Cohesion* flocking control algorithm. The connectivity is evaluated by the $C(t)$ value: (d) is for the *No-CoM* flocking control algorithm (2.38), (e) is for the *Multi-CoM-Shrink* flocking control algorithm, and (f) is for the *Multi-CoM-Cohesion* flocking control algorithm.

Multi-CoM-Cohesion .

To compare our proposed flocking control algorithms, *Multi-CoM-Shrink* and *Multi-CoM-Cohesion* with the existing flocking algorithm, *No-CoM* (2.38), we use the same initial state (position and velocity) of the mobile agents. Figure 4.4 shows the results of the tracking performance and the connectivity, respectively: (a, c) are for the *No-CoM* flocking control algorithm (2.38), (b, d) are for the *Multi-CoM-Shrink* flocking control algorithm, and (e, f) are for the *Multi-CoM-Cohesion* flocking control algorithm. Comparing the results in these figures we clearly see that:

- For the *No-CoM* flocking control algorithm (2.38): The tracking performance has big errors, and it makes the target out of the center of the network. In addition, the connectivity is lost, or the network is broken ($C(t) < 1$).

- For the *Multi-CoM-Cohesion* flocking control algorithm: The tracking performance has small errors. In addition, the agents can quickly form a network (only five iterations) and then maintain connectivity ($C(t) = 1$).

- For the *Multi-CoM-Shrink* flocking control algorithm: The tracking performance also has small errors, and the connectivity is maintained after six iterations. However, the size of the network is smaller than that of the *Multi-CoM-Cohesion* flocking control algorithm, and each agent has more neighbors because each agent tries to reduce the distance to its neighbor in order to keep connection to them.

4.5 Summary

In this chapter, we considered the problem of controlling a group of mobile agents to track a target in noisy environments. Two flocking control algorithms, *Multi-CoM-Shrink* and *Multi-CoM-Cohesion*, are proposed. In the *Multi-CoM-Shrink* algorithm our approach is to shrink the size of the network by reducing the distance among agents. In the *Multi-CoM-Cohesion* algorithm our approach integrates local position and velocity cohesion feedbacks

in oder to deal with the noise. The stability of the *Multi-CoM-Cohesion* algorithm is investigated based on the Lyapunov approach. Also, the network connectivity preservation is improved, and collision avoidance among agents is guaranteed in both cluttered and noisy environments.

CHAPTER 5

COOPERATIVE LEARNING OF PREDATOR AVOIDANCE IN MSNs

In this chapter we propose a hybrid system that integrates reinforcement learning and flocking control in order to create adaptive and intelligent MSNs. We study two problems in multiple mobile sensors concurrent learning of cooperative behaviors: (1) how to generate efficient combination of high level behaviors (discrete states and actions) and low level behaviors (continuous states and actions) for multiple mobile sensors cooperation; (2) how to conduct concurrent learning in a distributed fashion. To evaluate our theoretic framework, we apply it to enable MSNs to learn avoiding predators while maintaining network topology and connectivity. We also investigate the stability and scalability of our algorithm. The simulations and experiments are performed to demonstrate the effectiveness of the proposed hybrid system.

This chapter is organized as follows. Section 5.1 presents the introduction of this chapter. Section 5.2 presents a general framework to enable cooperative learning. Section 5.3 presents the model of multiple mobile sensors learning and then proposes a cooperative learning algorithm. Section 5.4 analyzes the convergence of the proposed learning algorithm. Section 5.5 shows the simulation and experiment results. Finally, conclusion of this chapter is given in Section 5.6.

5.1 Introduction

MSNs have great potentials in many military applications such as reconnaissance, surveillance and minefield clearance, etc. [114]. When an MSN are deployed to conduct such tasks, the enemy force may react and employ predators to attack the MSN. When such

attack occurs, the MSN may break up. In this scenario, the MSN should have the ability to avoid the enemy or predator. It is desirable that the MSN can avoid predator while maintaining the network topology and connectivity. From biology we know that there is an effective anti-predator function in animal aggregations [43, 44, 45], where the fish schools and bird flocks move together to create a sensory overload on the predator's visual channel (see Figure 1.1 in Chapter 1). This chapter focuses on the distributed decision making problem where each individual has a number of options (safe places) to choose from when the predators appear. Often in these decisions there is a benefit for consensus, where all individuals choose the same safe place. However, the consensus methods [40, 49, 50, 51, 52, 53, 54, 55, 56] require a connected network in which all mobile sensors can communicate with each other. This may not be valid in real environments because some mobile sensors may not connect to the network during the escape. In that case the consensus algorithms will fail. Therefore, in this chapter we are interested in the problem of reaching consensus even when the mobile sensors cannot connect to the network sometimes, but they can still make right decisions through learning from experience. Our method is based on a novel combination of flocking control [23] and reinforcement learning [100, 99].

Flocking control for multiple mobile agents studied in [37, 38, 112, 23] and our previous work [78, 64] was inspired by the natural phenomena of bird flock and fish school [25]. Basically, flocking control law is designed based on three basic flocking rules proposed by Reynolds in [25]: flock centering (agents try to stay close to nearby flock-mates), collision avoidance (agents try to avoid collision with nearby flock-mates), and velocity matching (agents try to match their velocity with nearby flock-mates).

In recent years, machine learning techniques such as reinforcement learning have been developed for MSNs that allow mobile sensors to learn cooperation [100, 99, 101]. However, traditional reinforcement learning assumes discrete and finite state/action spaces; therefore, it is difficult to directly apply reinforcement learning to most real world applica-

tions that inherently involve with continuous space. Furthermore, even if the states can be discretized, the learned behaviors are still discrete. In addition, the switching of discrete behaviors usually causes the control of the mobile sensors to become non-smooth, which is undesirable in most applications. To tackle these issues, several methods have been proposed to make the reinforcement learning work in continuous environments. The common approach is to use a function approximator to learn a value function, and there are several examples of successful applications [115, 116, 117, 118, 119]. In this chapter, instead of following such a common approach we try to combine reinforcement learning and flocking control to create a hybrid system. Our new framework allows the proposed system to:

- generate efficient combination of high level behaviors (discrete states and actions) and low level behaviors (continuous states and actions) for multiple mobile sensors cooperation.
- coordinate the concurrent learning process in a distributed fashion.

5.2 General Framework of Hybrid System in Multiple Mobile Sensors Domain

In this section, we build a general framework of cooperative learning in multiple mobile sensors cooperation. In this framework our goals are to:

- allow the mobile sensors to learn with continuous states and actions.
- coordinate the concurrent learning process to generate an efficient control policy in a distributed fashion.

With regard to the limitation of discrete and finite space, we propose a hybrid system of reinforcement learning in discrete space and flocking controller in continuous space as shown in Figure 5.1. This control architecture has two main parts, the reinforcement learning module (high level) and the flocking controller module (low level).

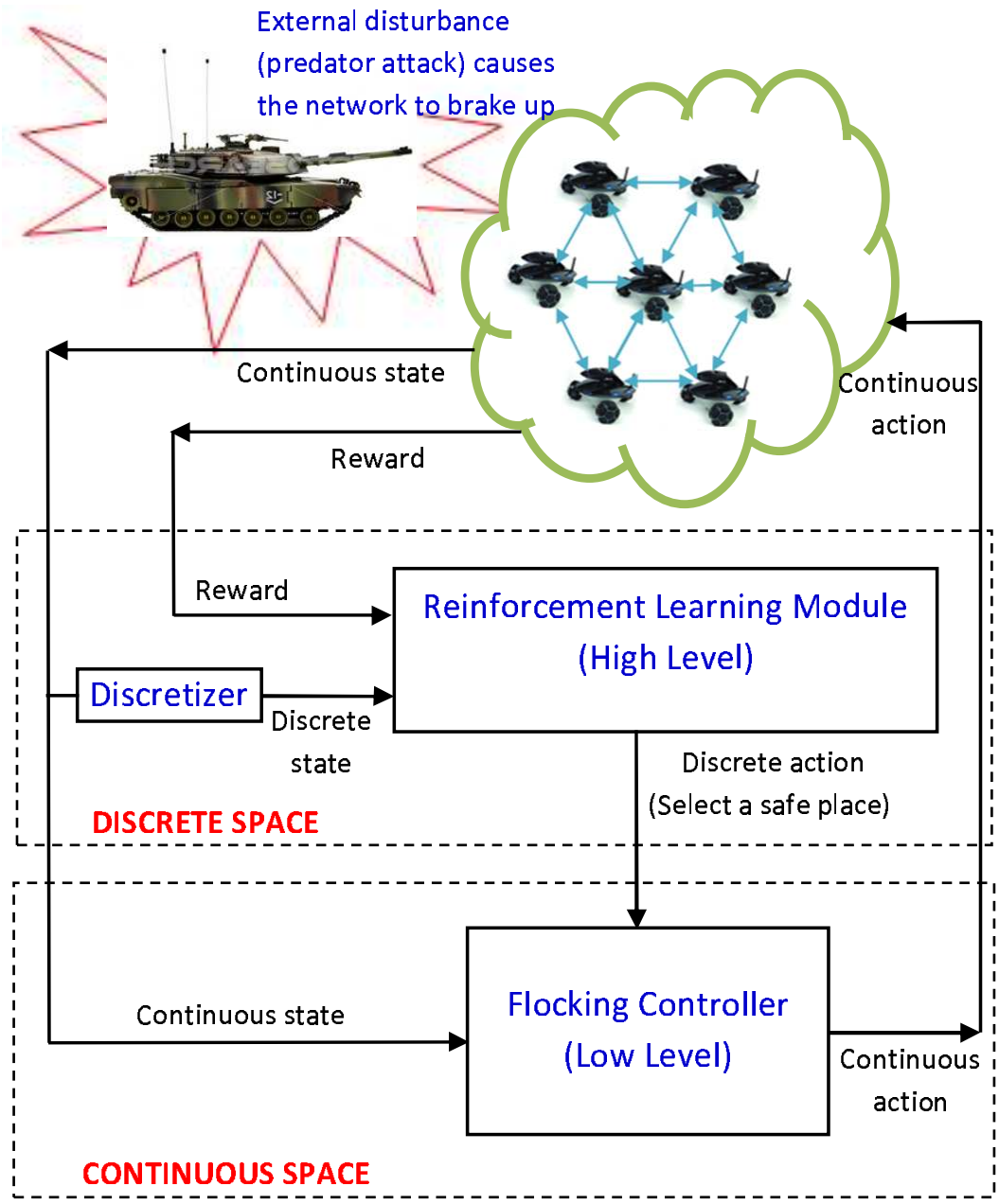


Figure 5.1: The hybrid system for reinforcement learning and flocking control in multiple mobile sensors domain.

The flocking controller (2.38), which works in a continuous space, is the network controller that controls all mobile sensors to move together without collision and track a stationary or moving target. In general, the target (q_t, p_t) is defined as follows

$$\begin{cases} \dot{q}_t = p_t \\ \dot{p}_t = f_t(q_t, p_t) \end{cases} \tag{5.1}$$

In this chapter we only consider a stationary target (a fixed point or safe place). Then q_t and p_t are considered to be constant vectors. When the predator is detected, several safe places $(q_{t_1}, q_{t_2}, ..., q_{t_N}, N \in Z)$ are generated by the prey. These safe places are generated based on the moving direction of the predator to maximize the escaping probability. For example, these safe places can be located at four corners centered at the moving trajectory of the predator.

The flocking controller also allows the mobile sensors to avoid the predators based on a repulsive force generated from an artificial potential field induced by the predators. However, this repulsive force usually breaks up the network. Therefore, we need combine both flocking control and reinforcement learning so that they can avoid the predators while maintaining network formation (topology) and connectivity.

The reinforcement learning module, which works in discrete space, is the key to the controller. The goal is to agree on one of the safe places for the flocking controller. By retrieving the states (after they are discretized) and the rewards, the reinforcement learning module finds the appropriate safe place so that the network topology and connectivity can be maintained.

Our framework is valid in real situations when the predators continuously attack the prey network, and the prey can learn this behavior of the predators in order to agree on the same decision. Since all mobile sensors in a cooperative multiple mobile sensors system can influence each other, it is important to ensure that the actions are selected by the individual mobile sensors result in effective decisions for the whole group.

5.3 Modeling and Cooperative Learning Algorithm

In this section we build a model of multiple mobile sensors learning to avoid predator and then develop the cooperative learning algorithm.

5.3.1 Model of Multiple Mobile Sensors Learning

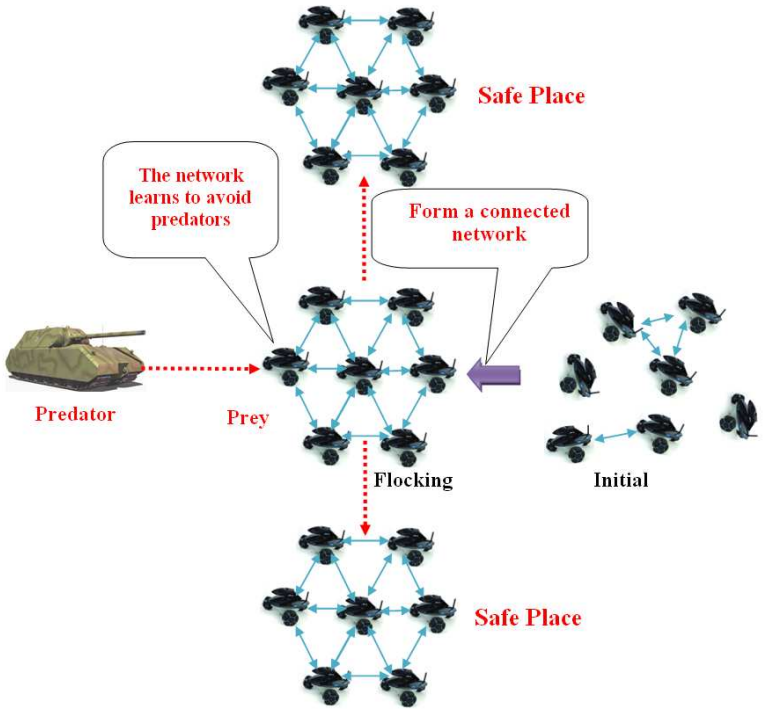


Figure 5.2: Illustration of the safe places to choose.

The multiple mobile sensors learning problem can be illustrated in Figure 5.2. In this figure, the mobile sensors learn to make the same decision (select the same safe place to go) so that the network will not break up, and the network topology and connectivity can be maintained. Based on the moving direction of the predator, the safe places are real-timely generated by the network of prey. If the prey reaches the safe places, and the predators keep attacking, then other safe places will be generated in oder to continuously avoid the predators.

We model the predators as follows:

- The predators try to go into the center of the network. This behavior of the predators is usually adopted in existing works [120, 121].

- The velocity of the predators is faster than that of the prey (mobile sensor).

Usually these behaviors of the predators will cause the prey network to break up. As a result, the prey will not flock together. This is one of the reasons that the prey have to learn in a cooperative fashion so that they can agree on the same safe place to escape the predators [43]. Therefore, we model the prey (mobile sensors) as follows:

- All mobile sensors flock together in free space and form an α-lattice formation [23] based on the distributed flocking control algorithm (2.38).

- If the predators come into the detection range (R_2), the mobile sensor (prey) can sense the location of the predators. The mobile sensor will learn and select one of the safe places to go (see Figure 5.3).

- If the predators come into the risk area (R_1), the mobile sensor will move away based on the repulsive force via the function f_i^{β} defined in Equation (2.30). Here, we can set R_1 equal to r' as defined in Section II.

5.3.2 Cooperative Learning Algorithm

In this subsection we define the state, action and reward, and then present an independent reinforcement learning algorithm. Finally, we develop a cooperative reinforcement learning algorithm based on Q-learning.

State, Action and Reward

Let the current state, action and reward of mobile sensor i be s_i, a_i, r_i, respectively, and the next state and action of mobile sensor i be $s_i^{\cdot}, a_i^{\cdot}$, respectively. At each moment, we have a partially observable environment. This means that not all mobile sensors are able to see the

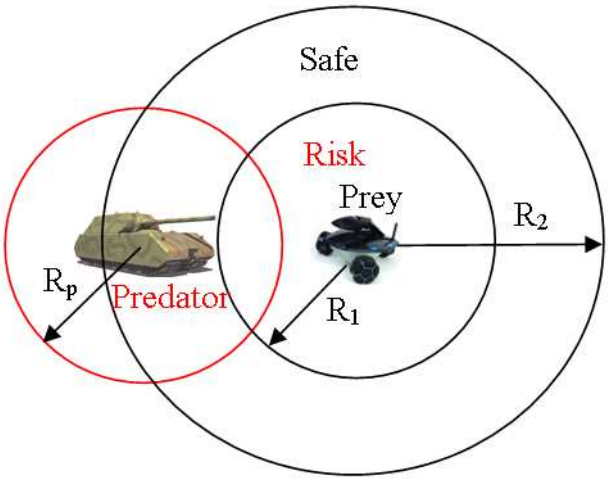


Figure 5.3: Illustration of the predator and prey detection ranges. R_1 is the active range of the mobile sensor (prey), R_2 is the predator detection range, and R_p is the prey detection range.

predators, and each mobile sensor only communicates with its neighbors to exchange local information. We have the following models for the state, action and reward.

The state: we assume that when the learning starts (all mobile sensors flocked together and formed an α-lattice formation) the state is initialized. For each mobile sensor i, the state is defined as the number of the predators n_p in the detection range R_2, and the number of neighboring mobile sensors $|N_i^\alpha|$ in its active range r, $s_i = [n_p,\ |N_i^\alpha|]^T$. For example, if one predator is in the detection range and six neighboring mobile sensors are in the active range of mobile sensor i then the state for mobile sensor i is $[1,6]^T$. If only six neighboring mobile sensors are in the active range, and no predator is in the detection range of the mobile sensor i then the state for mobile sensor i is $[0,6]^T$. If the mobile sensor i performs the action, i.e., selecting one safe place, it will keep moving until the state changes to a different state, $s_i^{'} \neq s_i$. The maximum number of states depends on the number of the mobile sensors and predators. Hence, we have the maximum number of states or the state list (S_i) of mobile sensor i in the case of a single predator to be

$$S_i = [1,n-1]^T, [0,n-1]^T, [1,n-2]^T, [0,n-2]^T, ..., [1,0]^T, [0,0]^T. \tag{5.2}$$

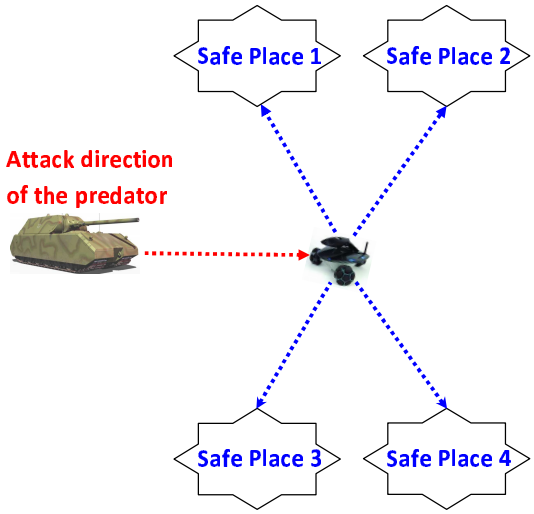


Figure 5.4: Four safe places are generated based on the moving direction of the predator.

Overall, the maximum number of states of mobile sensor i in the case of one predator equals to $2n$. Since all mobile sensors want to maintain the connection to the network, they want to avoid the states $[1,0]^T$ and $[0,0]^T$.

The action: We assume that the predators can come from any direction with different paths. However, when they detect the prey they try to come into the center of the prey network. Therefore, the desired action of the prey (mobile sensors) is to go to one of four safe places to escape. If we encode 4 safe places as numbers 1, 2, 3, 4, we have the action list for each mobile sensor $A_i = [1,\ 2,\ 3,\ 4]$. When the predators enter the risk area, the mobile sensor will generate the repulsive force to move away from them. Additional actions can be introduced if needed. The illustration of this scenario is shown in Figure 5.4. The action, selecting one of the safe places, is generated in the Reinforcement Learning module. Then, this action is implemented in the flocking controller.

The reinforcement reward: the reinforcement reward signal changes in the experiments, depending on the input data that is received. In α-lattice configuration (hexagonal lattice configuration), a mobile sensor inside the network has six neighbors, and the mobile sensor on the border of the network has one to five neighbors. Our purpose is to maintain this network configuration, hence we define the reward as: if $|N_i^\alpha| < 6$ then $r_i = |N_i^\alpha|$, else

$r_i = 6$ (keep an hexagonal lattice configuration). This reward definition basically implies that the maximum reward for each agent is six which corresponds to the hexagonal lattice configuration of the network.

Independent Learning

For comparison purpose, we implement an independent learning algorithm in which the mobile sensors ignore the actions and rewards of other mobile sensors, and learn their strategies independently. Each mobile sensor stores and updates an individual table, Q_i, as follows:

$$Q_i(s_i, a_i) \Leftarrow Q_i(s_i, a_i) + \alpha[r_i + \gamma max_{a_i^{'} \in A_i^{'}} Q_i(s_i^{'}, A_i^{'}) - Q_i(s_i, a_i)] \tag{5.3}$$

here α is a learning rate, and γ is a discounting factor, and $A_i^{'}$ is a next action list of current action list A_i.

Cooperative Learning

We propose a cooperative learning algorithm which has two phases. The first phase is Q value update, and the second one is action selection. In the first phase we let each mobile sensor calculate its own Q value based on its own action/state and its neighbor's actions/states. In the second phase, in order to make the learning converge faster we develop a majority action following (MAF) algorithm for the final action selection. *Q Value Update:*

In this phase, our goal is to allow each mobile sensor to aggregate the information of its neighbors via the Q value. Therefore each mobile sensor updates its Q value based on the following equation.

$$Q_i(s_i, a_i) \Leftarrow Q_i(s_i, a_i) + \sum_{j=1}^{|N_i^{\alpha}|} Q_j(s_j, a_j) \tag{5.4}$$

here, $Q_i(s_i, a_i)$ is computed based on Equation (5.3), and $|N_i^{\alpha}|$ is the number of neighbors of mobile sensor i. This idea of Q value update is illustrated in Figure 5.5.

Action Selection Strategy:

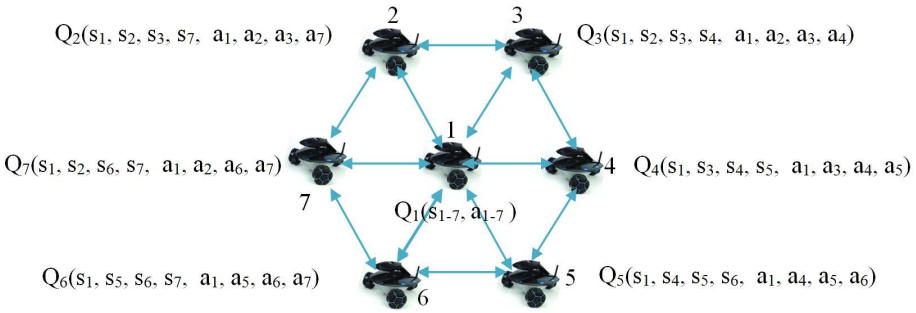


Figure 5.5: Q value update based on the mobile sensor's action/state and its neighboring actions/states.

Usually the next action selection in reinforcement learning is based on the maximum Q value [100, 99], or the Boltzmann action selection strategy [122]. Since each mobile sensor makes its decision only based on the maximum Q value, the convergence of the learning is usually slow. Therefore we need make the preys to agree on the same action as fast as possible. We first let each prey select the next action based on the maximum Q value. The final action for each mobile sensor is decided using the majority action following strategy, which is shown in Algorithm 4. In this algorithm, Step 3 can let each prey select its action as the one that most of its neighbors follow. In this way the cooperative learning can converge faster. Overall, the cooperative learning algorithm is shown in Algorithm 5.

Algorithm 4: Majority Action Following (MAF) Algorithm

for *each mobile sensor i* **do**

Step 1. Selects the next action based on the maximum Q value, $max_{a_i \in A_i} Q_i(s_i, A_i)$, where $Q_i(s_i, a_i)$ is computed via Equation (5.4).

Step 2. Asks/observes its neighbor's decisions.

Step 3. Selects the action that most mobile sensors in the inclusive neighborhood set $\{i \cup N_i^{\alpha}\}$ follow.

if *the number of mobile sensors in the set* $\{N_i^{\alpha}\}$ *selecting the same action are the same* **then**

Robot i will keep its own decision;

else

Goes back to **Step 3.**

5.4 Convergence Analysis of Cooperative Learning Algorithm

In this section, we show the convergence of our proposed cooperative learning Algorithm 5. First, we can rewrite Equation (5.3) iteratively:

$$Q_i^{k+1}(s_i, a_i) = Q_i^k(s_i, a_i) + \alpha[r_j + \gamma Q_i^k(s_i^{,}, a_i^{,}) - Q_i^k(s_i, a_i)] \tag{5.5}$$

here k is the time step, and $Q_i^k(s_i^{,}, a_i^{,}) = max_{a_i^{,} \in A_i^{'}} Q_i^k(s_i^{,}, A_i^{'})$. Let $|N_i^{\alpha} \cup \{i\}|$ be the number of mobile sensors in mobile sensor i's local neighborhood including mobile sensor i itself. Then from (5.4) and (5.5) we have the sum of Q values in each local set $\{N_i^{\alpha} \cup \{i\}\}$ as

$$\begin{aligned} \sum_{j=1}^{|N_i^{\alpha} \cup \{i\}|} Q_j^{k+1}(s_j, a_j) &= \sum_{j=1}^{|N_i^{\alpha} \cup \{i\}|} \alpha[r_i + \gamma Q_j^k(s_j^{,}, a_j^{,}) - Q_j^k(s_j, a_j)] \\ &\quad + \sum_{j=1}^{|N_i^{\alpha} \cup \{i\}|} Q_j^k(s_j, a_j). \end{aligned} \tag{5.6}$$

Since each mobile sensor i updates its final Q_i at time step k based on $Q_i^k(s_i, a_i) = \sum_{j=1}^{|N_i^{\alpha} \cup \{i\}|} Q_j^k(s_j, a_j)$ we have the difference $\Delta Q_i(s_i, a_i)$ for each mobile sensor between the

Algorithm 5: Cooperative Learning in a Distributed Fashion

Set parameters α and γ.

Build the state list (S_i) and action list (A_i) for each mobile sensor

for *each episode* **do**

for *each mobile sensor i* **do**

Initialization phase:

- Initializes the matrix Q_i

- Finds initial state (s_i) that corresponds to the one in the state list (S_i) as defined in Equation (5.2).

- Randomly selects one action (a_i) in the action list (A_i).

Update phase (after mobile sensor *i* does the selected action):

- Updates the next state $(s_i^{'})$.

- Selects the next action $(a_i^{'})$ based on the maximum Q_i value.

- Computes the reward r_i.

- Computes Q_i value:

$$Q_i(s_i,a_i) \quad \Leftarrow \quad Q_i(s_i,a_i)+\alpha[r_i+\gamma max_{a_i^{'}\in A_i^{'}}Q_i(s_i^{'},A_i^{'})-Q_i(s_i,a_i)]$$

- Updates Q_i based on its neighbors:

$$Q_i(s_i,a_i) \Leftarrow Q_i(s_i,a_i)+\sum_{j=1}^{|N_i^{\alpha}|}Q_j(s_j,a_j)$$

here, $|N_i^{\alpha}|$ is the number of neighbors of mobile sensor i.

- Sets the next state as the current state.

Action implementation phase:

- Final action is selected based on Majority Action Following Algorithm (Algorithm 1).

end

end

time steps $(k+1)$ and k as

$$\begin{aligned}\Delta Q_i(s_i,a_i) &= Q_i^{k+1}(s_i,a_i) - Q_i^k(s_i,a_i) \\ &= \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} Q_j^{k+1}(s_j,a_j) - \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} Q_j^k(s_j,a_j) \\ &= \alpha \sum_{j=1}^{|N_i^\alpha \cup \{i\}|} [r_j + \gamma Q_j^k(s_j^{'},a_j^{'}) - Q_j^k(s_i,a_j)]. \end{aligned} \tag{5.7}$$

We can expand Equation (5.7) to n mobile sensors into space representation as follow:

$$\Delta \mathbf{Q}(\mathbf{s},\mathbf{a}) = \alpha \mathbf{R}(\mathbf{s},\mathbf{a}) + \mathbf{H}\mathbf{Q}(\mathbf{s}^{'},\mathbf{a}^{'}) - \alpha \mathbf{Q}(\mathbf{s},\mathbf{a}) \tag{5.8}$$

here $\Delta \mathbf{Q}(\mathbf{s},\mathbf{a}) = [\Delta Q_1(s_1,a_1), \Delta Q_2(s_2,a_2), ..., \Delta Q_n(s_n,a_n)]^T$ with $\mathbf{s} = [s_1, s_2, ..., s_n]^T$ and $\mathbf{a} = [a_1, a_2, ..., a_n]^T$.

$\mathbf{Q}(\mathbf{s},\mathbf{a}) = [\sum_{j=1}^{|N_1^\alpha|+1} Q_j(s_j,a_j), \sum_{j=1}^{|N_2^\alpha|+1} Q_j(s_j,a_j)$
$, ..., \sum_{j=1}^{|N_n^\alpha|+1} Q_j(s_j,a_j)]^T$.

$\mathbf{Q}(\mathbf{s}^{'},\mathbf{a}^{'}) = [\sum_{j=1}^{|N_1^\alpha|+1} Q_j(s_j^{'},a_j^{'}), \sum_{j=1}^{|N_2^\alpha|+1} Q_j(s_j^{'},a_j^{'})$
$, ..., \sum_{j=1}^{|N_n^\alpha|+1} Q_j(s_j^{'},a_j^{'})]^T$,
with $\mathbf{s}^{'} = [s_1^{'}, s_2^{'}, ..., s_n^{'}]^T$ and $\mathbf{a}^{'} = [a_1^{'}, a_2^{'}, ..., a_n^{'}]^T$.

$\mathbf{R} = [\sum_{j=1}^{|N_1^\alpha|+1} r_j(s_j,a_j), \sum_{j=1}^{|N_2^\alpha|+1} r_j(s_j,a_j), ..., \sum_{j=1}^{|N_n^\alpha|+1} r_j(s_j,a_j)]^T$.

$$\mathbf{H} = \begin{bmatrix} \alpha\gamma & \alpha\gamma & ... & \alpha\gamma \\ \alpha\gamma & \alpha\gamma & ... & \alpha\gamma \\ ... & ... & ... & ... \\ \alpha\gamma & \alpha\gamma & ... & \alpha\gamma \end{bmatrix}_{nxn} .$$

Since the next action $a_i^{'}$ is selected according to $max_{a_i \in A_i} Q_i(s_i,A_i)$, and the final action is selected based on the MAF, we can rewrite Equation (5.8) as:

$$\Delta \mathbf{Q}(\mathbf{s},\mathbf{a}) = \alpha \mathbf{R}(\mathbf{s},\mathbf{a}) + \mathbf{H}\mathbf{F}(\mathbf{s}) - \alpha \mathbf{Q}(\mathbf{s},\mathbf{a}) \tag{5.9}$$

here $\mathbf{F}(\mathbf{s}) = [\sum_{j=1}^{|N_1^\alpha|+1} F_j(s), \sum_{j=1}^{|N_2^\alpha|+1} F_j(s), ..., \sum_{j=1}^{|N_n^\alpha|+1} F_j(s)]^T$ with $F_j(s) = max_{a_j \in A_j} Q_j(s_j, A_j)$.

Theorem 4. Consider a system of *n* mobile sensors, that have dynamics (2.18) and are controlled by the control law (2.38). Based on Algorithm 5 and a state differential equation (5.9) the vector $\Delta\mathbf{Q}(\mathbf{s},\mathbf{a})$ will converge to a zero vector.

Proof:

Since after sufficient iterations, each learning mobile sensor will select the action that holds the biggest *Q* value we have $\mathbf{F}(\mathbf{s}) = \mathbf{Q}(\mathbf{s},\mathbf{a})$. From this point we can rewrite Equation (5.9) as

$$\Delta\mathbf{Q}(\mathbf{s},\mathbf{a}) = (\mathbf{H} - \alpha\mathbf{I})\mathbf{Q}(\mathbf{s},\mathbf{a}) + \alpha\mathbf{R}(\mathbf{s},\mathbf{a}) = \mathbf{B}\mathbf{Q}(\mathbf{s},\mathbf{a}) + \alpha\mathbf{R}(\mathbf{s},\mathbf{a}) \tag{5.10}$$

here $\mathbf{I}_{n\times n}$ is an identity matrix, and $\mathbf{B} = (\mathbf{H} - \alpha\mathbf{I})$.

If the time step is small enough we can write Equation (5.10) into a continuous fashion as:

$$\dot{\mathbf{Q}}(\mathbf{s},\mathbf{a}) = \mathbf{B}\mathbf{Q}(\mathbf{s},\mathbf{a}) + \alpha\mathbf{R}(\mathbf{s},\mathbf{a}) \tag{5.11}$$

Now, we can consider Equation (5.11) as a standard feedback control system ($\dot{X} = AX + Bu$) [123, 124]. Namely, the model of the system is $\mathbf{Q}(\mathbf{s},\mathbf{a})$, and the control input is α, and $\mathbf{R}$ is the output signal of the controller. Therefore we can easily see that if all of the roots of the characteristic equation of the differential equation (5.11) have negative real parts then the proposed system (5.11) is stable, or the vector $\Delta\mathbf{Q}(\mathbf{s},\mathbf{a})$ will converge to a zero vector.

We have the characteristic equation:

$$det(\lambda\mathbf{I} - \mathbf{B}) = 0 \tag{5.12}$$

here

$$\lambda\mathbf{I} - \mathbf{B} = \begin{bmatrix} \lambda - \alpha(\gamma-1) & -\alpha\gamma & ... & -\alpha\gamma \\ -\alpha\gamma & \lambda - \alpha(\gamma-1) & ... & -\alpha\gamma \\ ... & ... & ... & ... \\ -\alpha\gamma & -\alpha\gamma & ... & \lambda - \alpha(\gamma-1) \end{bmatrix}_{nxn}$$

From Equation (5.12) we can obtain:

$$(\lambda+\alpha)^n - n(\lambda+\alpha)^{n-1}\alpha\gamma = 0 \tag{5.13}$$

or,

$$(\lambda+\alpha)^{n-1}[(\lambda+\alpha) - n\alpha\gamma] = 0 \tag{5.14}$$

Solve Equation (5.14) we obtain the roots as: $\lambda_1 = \lambda_2 = ... = \lambda_{n-1} = -\alpha$. Since $0 < \alpha < 1$ we have $-1 < \lambda_1 = \lambda_2 = ... = \lambda_{n-1} < 0$, and $\lambda_n = \alpha(n\gamma - 1)$. We can easily see $\lambda_n < 0$ if we select $0 < \gamma < \frac{1}{n}$.

We can ensure that all the roots of the characteristic equation of the differential equation (5.11) have negative real parts if we select $0 < \alpha < 1$ and $0 < \gamma < \frac{1}{n}$. Hence we can conclude that the proposed system (5.11) is stable, or the vector $\Delta\mathbf{Q}(\mathbf{s},\mathbf{a})$ will converge to a zero vector. This finishes our proof.

5.5 Simulation and Experiment Results

In this section we test our cooperative learning algorithm (Algorithm 5) combined with the distributed flocking control algorithm (2.38) in both simulation and experiment. We compare the proposed cooperative learning algorithm with the independent learning algorithm (5.3) in term of connectivity, topology, convergence, action and reward.

5.5.1 Simulation Results

In this simulation we use 15 mobile sensors (prey), and 4 actions (4 safe places to escape predator). This results in $4^{15} = 1073741824$ ($\approx$1 billion) possible joint actions.

In each learning episode we randomly setup initial deployments of the prey; locations of obstacles; as well as trajectories and initial locations of the predator. The learning task is to find out one of four optimal joint actions. The parameters of flocking control are as follows: the desired distance among the prey $d = 16$; the scaling factor $k_c = 1.2$; the active range $r = k_c \times d = 19.2$; the constant $\varepsilon = 0.1$ for the σ-norm.

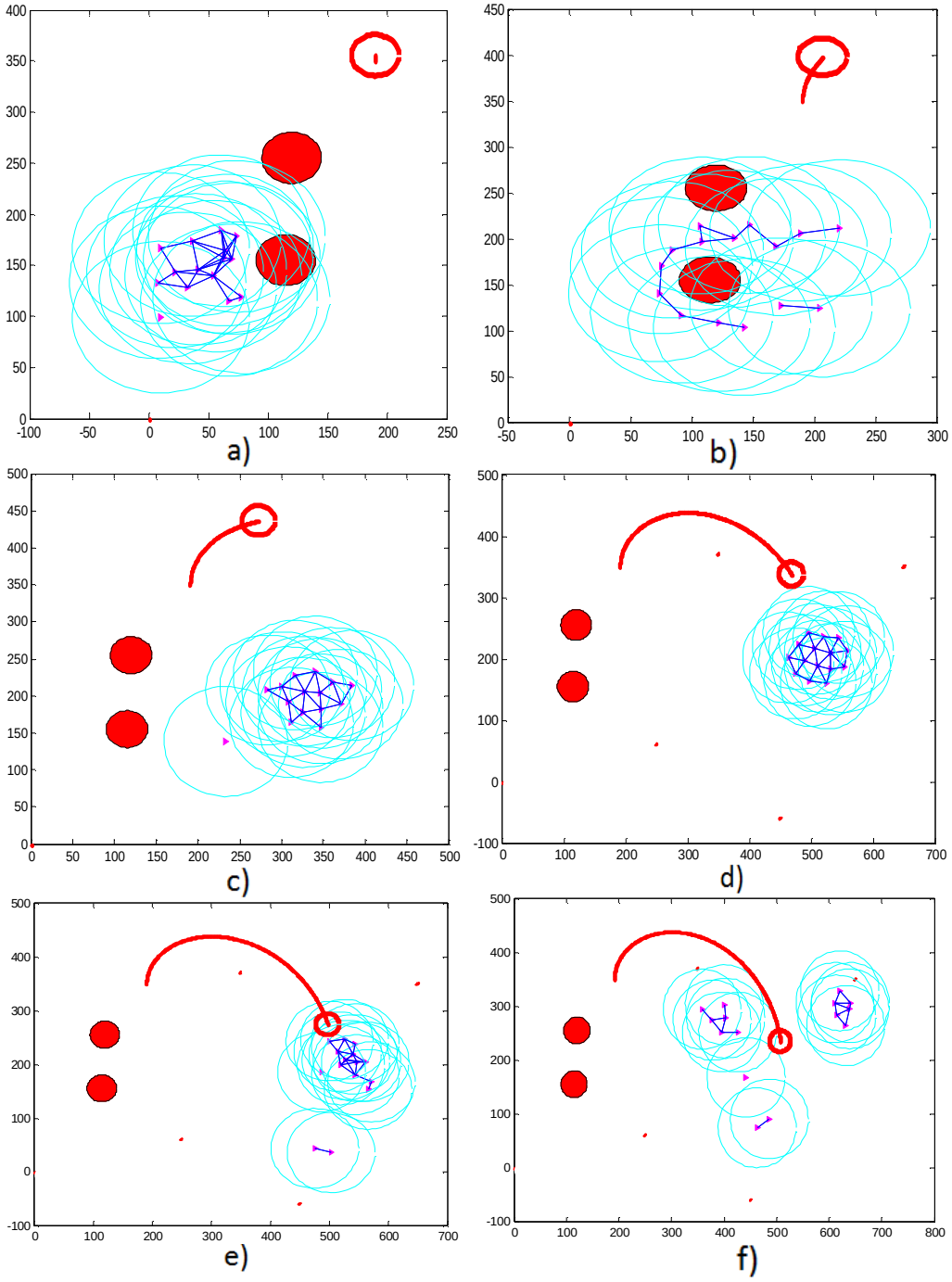


Figure 5.6: Snapshots of our proposed cooperative learning algorithm in the first episode. Four red/darker dots as shown in snapshots (d, e, f) are four safe places. The empty red circle is the predator. The filled red circles are the obstacles.

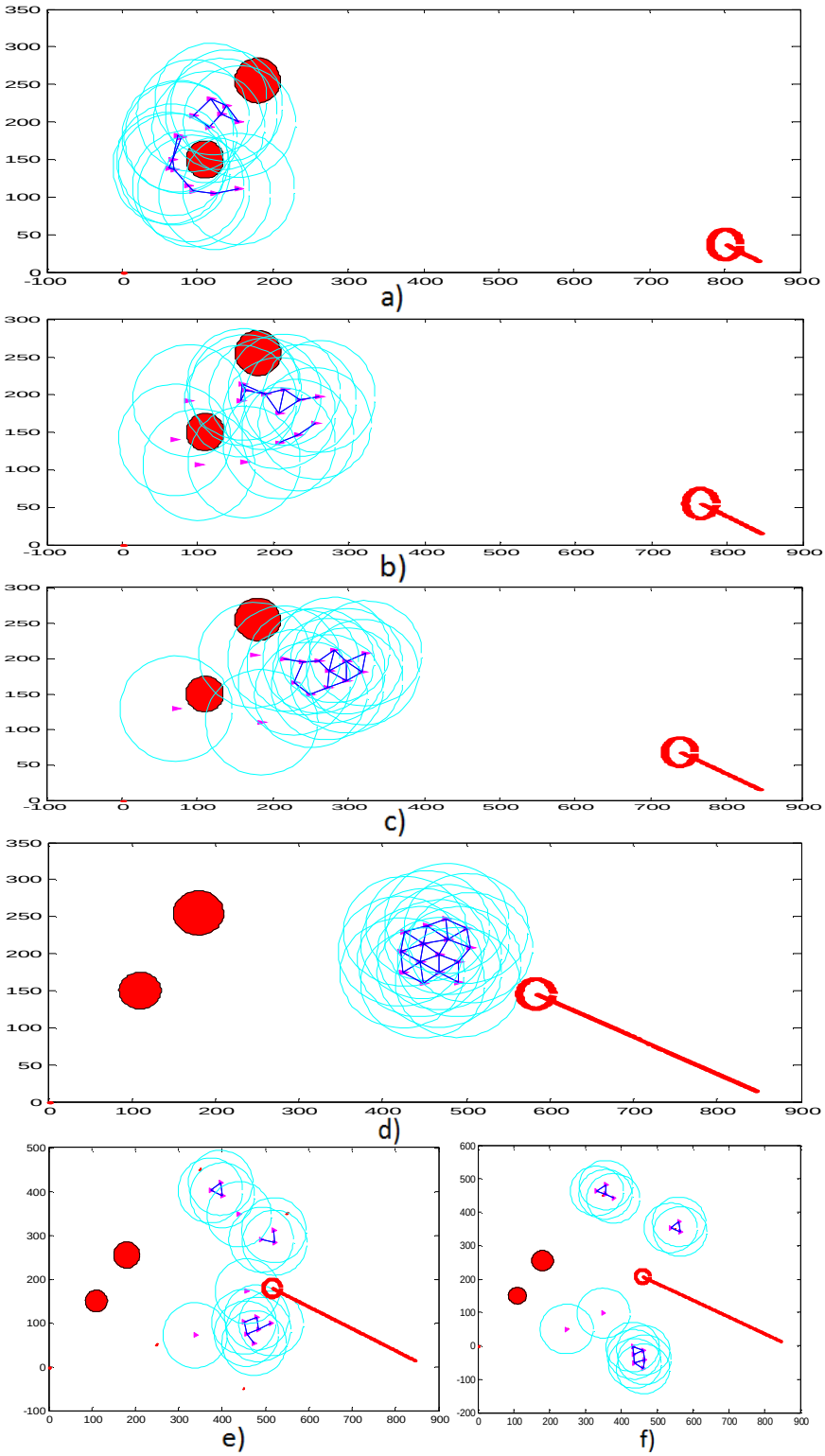


Figure 5.7: Snapshots of our proposed cooperative learning algorithm in the second episode. Four red/darker dots as shown in snapshots (e, f) are four safe places. The empty red circle is the predator. The filled red circles are the obstacles.

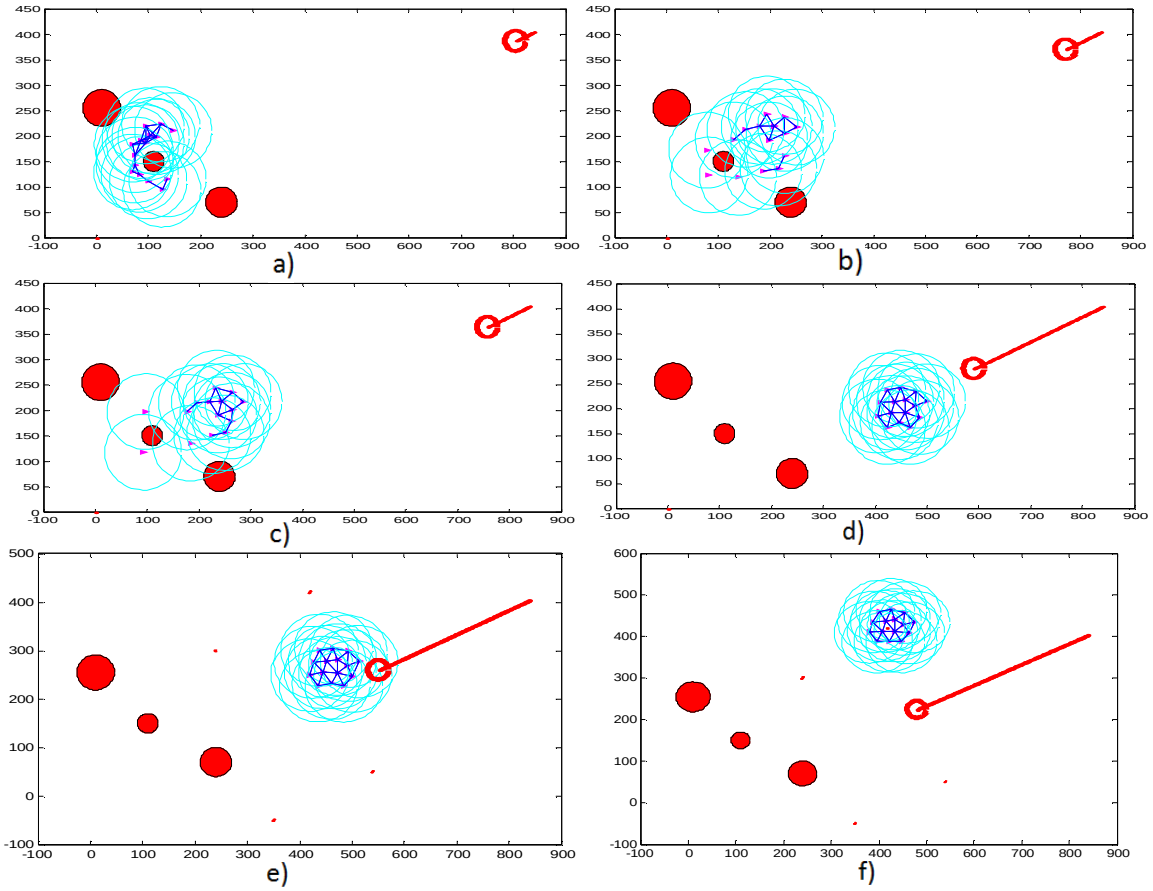


Figure 5.8: Snapshots of the proposed cooperative learning algorithm in the third episode. Four red/darker dots as shown in snapshots (e, f) are the four safe places. The empty red circle is the predator. The filled red circles are the obstacles.

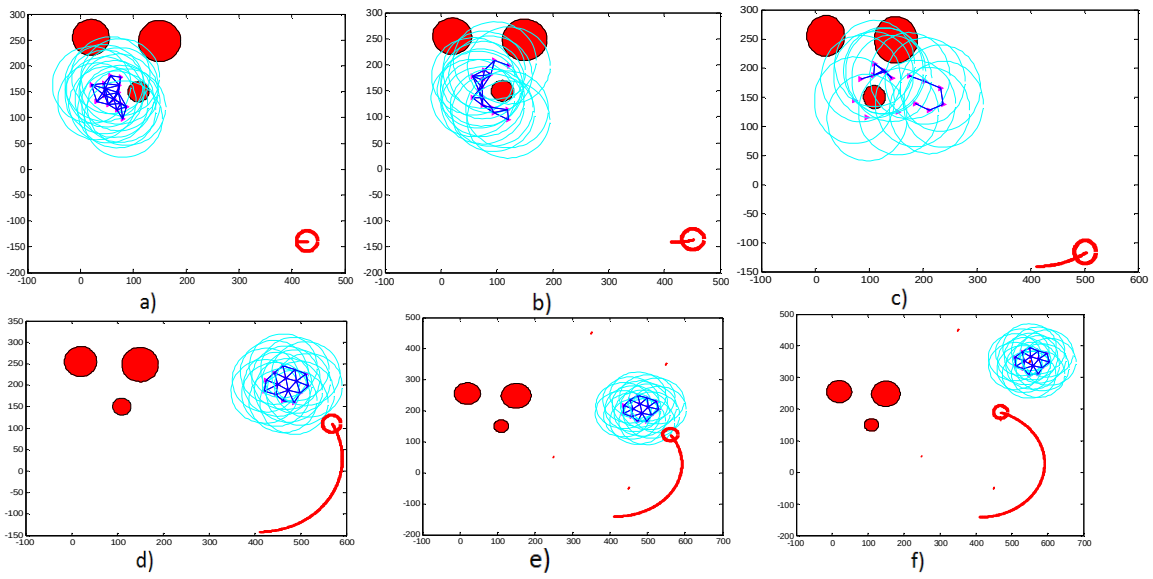


Figure 5.9: Snapshots of the proposed cooperative learning algorithm in the fourth episode. Four red/darker dots as shown in snapshots (e, f) are the four safe places. The empty red circle is the predator. The filled red circles are the obstacles.

Figure 5.6 shows the result of the first training episode. Since at the first time the mobile sensors do not have any experience, they failed to agree on the same action. Hence, the network is broken.

In the second episode shown in Figure 5.7, the learning result is better since more than 50 percent of the mobile sensors agree on the same safe place to go. In the third episode the learning converges and all the mobile sensors choose the same action (see Figure 5.8). Therefore the connectivity is maintained. In the fourth (see Figure 5.9) and fifth episodes (see Figure 5.10), even when we change the trajectory of the predator and the location of the obstacles, the learning results still hold.

5.5.2 Experiment Results

In real experiments we use eight Rovio robots [111] that have omni-directional motion capability as shown in Figure 5.11. In this figure, seven Rovio robots are used as preys, and one Rovio robot is used as a predator. To distinguish with other prey robots the predator

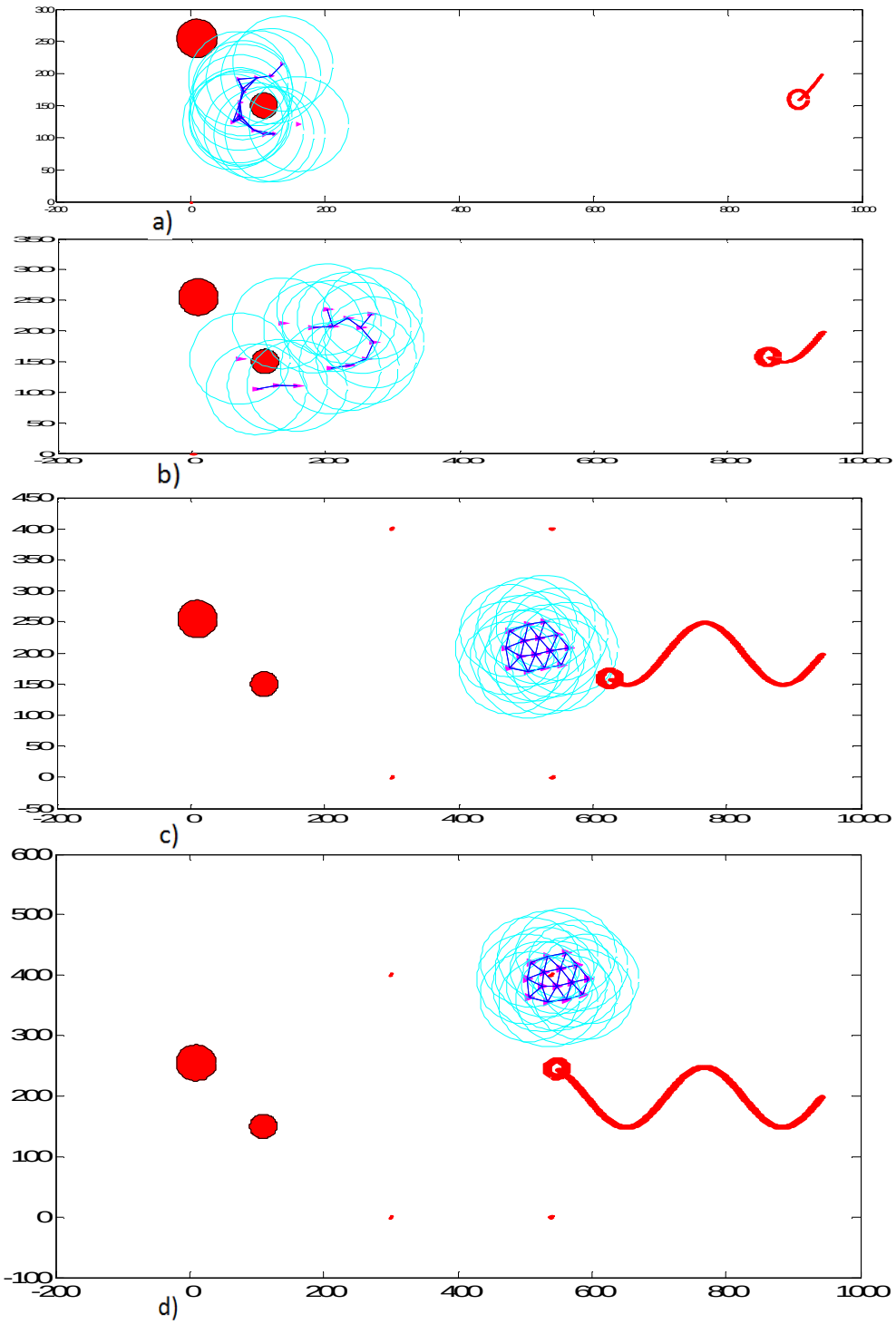


Figure 5.10: Snapshots of the proposed cooperative learning algorithm in the fifth episode. Four red/darker dots as shown in snapshots (c, d) are the four safe places. The empty red circle is the predator. The filled red circles are the obstacles.

Figure 5.11: Seven Rovio prey mobile sensors and one Rovio predator robot (marked with a yellow cup) are used in the experiment.

robot is marked by a yellow cup mounted on it. Basically, these robots can freely move in six directions. The dynamic model of the Rovio robot can be approximated by Equation (2.18). However, the accuracy of the localization of the Rovio robot is low, and the robot does not have any sensing device to sense the pose (position and velocity) of its neighbors, predator and obstacles. Hence we use a VICON motion capture system [1] in our lab (Figure 5.12) that has 12 infrared cameras to track moving objects. This tracking system can give the location and velocity of each moving object with high accuracy.

Figure 5.13 shows the experimental result of the first training episode. Similar to the simulation results, since in the first episode the robots do not have any experience of the behavior of the predator, they failed to agree on the same action. Hence, the network is broken. In the third episode as shown in Figure 5.14 the learning converges and all the robots choose the same action (same safe place). Therefore the topology and the connectivity are maintained.

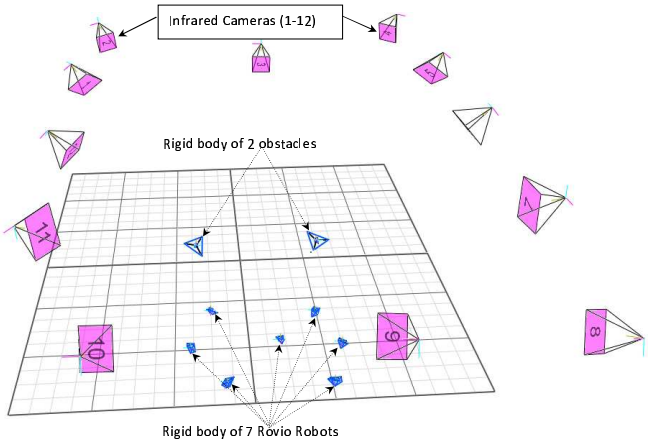


Figure 5.12: Infrared cameras tracking system for experimental setup of multi-robot cooperative learning.

5.5.3 Performance Evaluation

In this subsection we evaluate the connectivity, topology, convergence, and reward performance of our proposed cooperative learning algorithm, then compare with those of the independent learning algorithm.

Connectivity Evaluation

From the result in Figure 5.15 we can see that for the cooperative learning algorithm the connectivity of the network is maintained after 3 training episodes while for the independent learning algorithm the connectivity is not maintained even after 100 training episodes. This means that the robots do not agree on the same action. Note that in Figure 5.15*d* (zoom in at 4th episode of Figure 5.15*b*) the connectivity is only lost from iteration 1 to 320 because the prey have to avoid the obstacles. After about 320 iterations the predator appears, and the preys can avoid the predator and maintain the connectivity based on the proposed cooperative learning algorithm. In contrast, the connectivity is lost using the independent learning algorithm as shown in Figure 5.15*a* and *c*. Here Figure 5.15*a* is zoom in at 100th episode of Figure 5.15*c*.

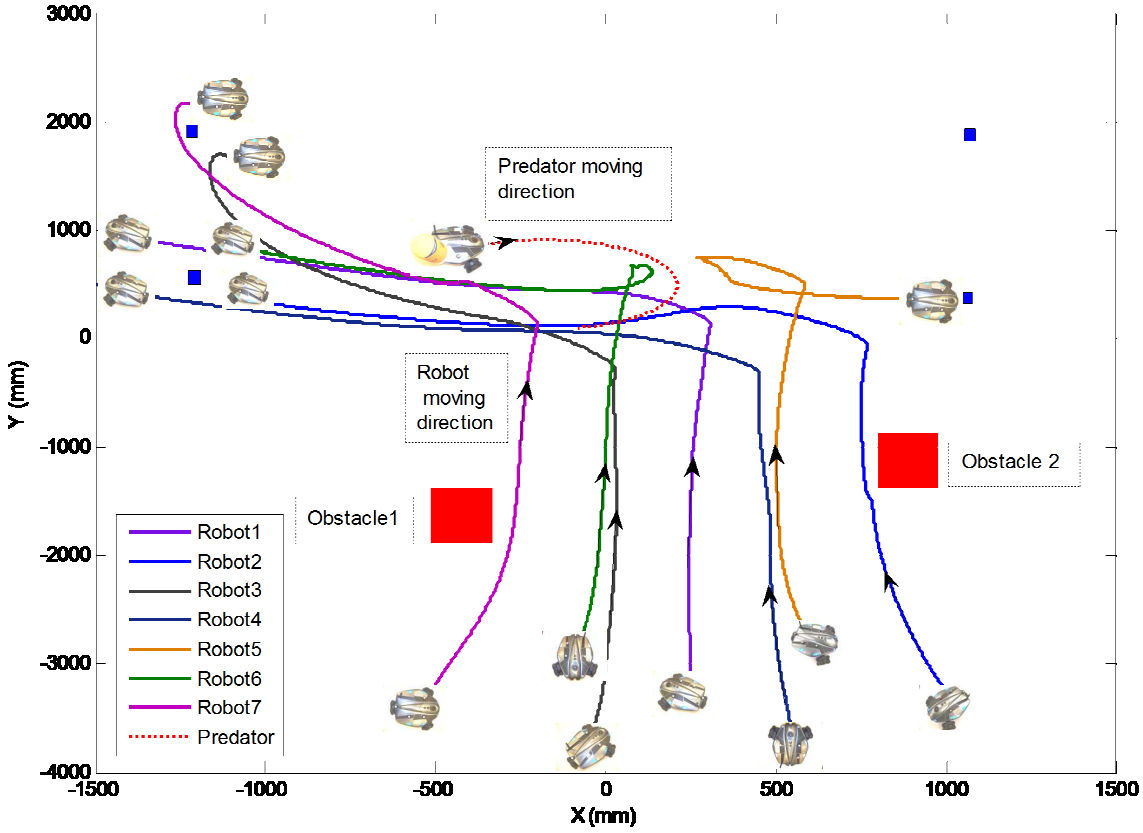


Figure 5.13: The trajectories of 7 Rovio robots and one predator in the first learning episode. The green small squares are the safe places, and the filled red squares are the obstacles.

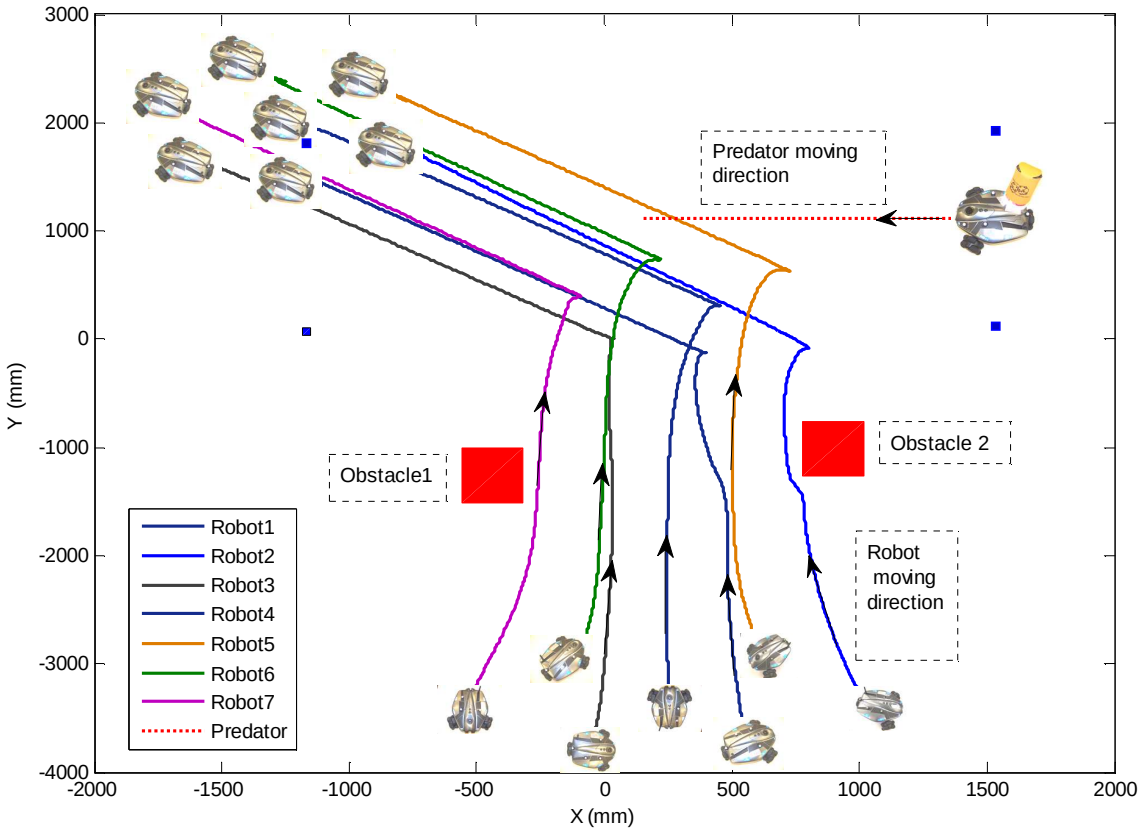


Figure 5.14: The trajectories of 7 Rovio robots and one predator in the third learning episode. The green small squares are the safe places, and the filled red squares are the obstacles.

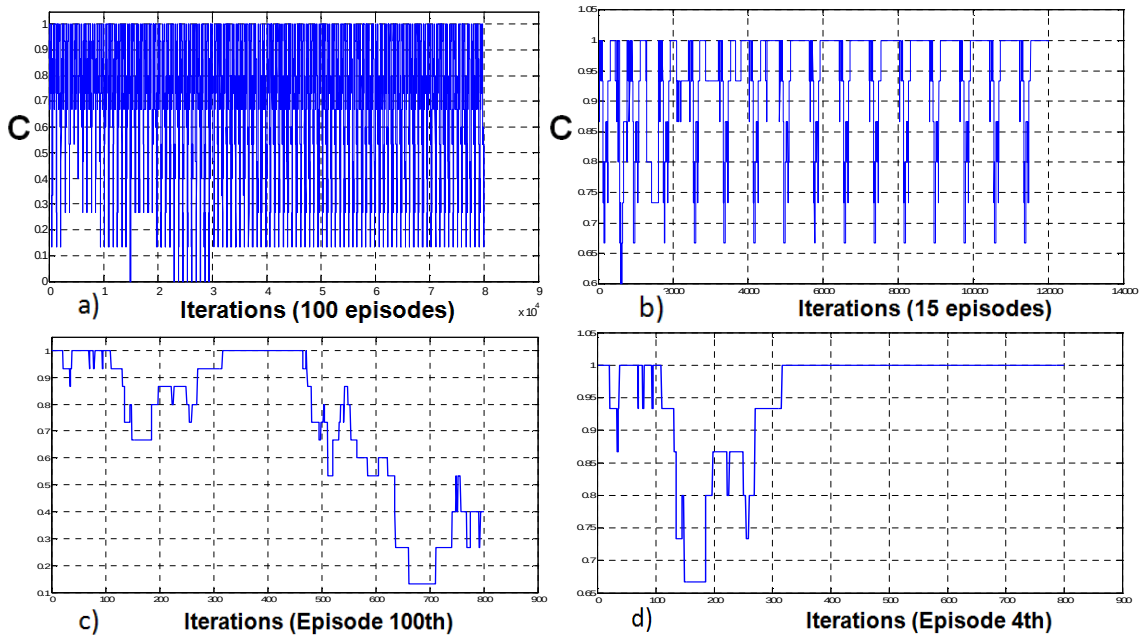


Figure 5.15: Connectivity evaluation for the independent learning algorithm (a, c) and our proposed cooperative learning algorithm (b, d), here (c) is a zoom-in at 100th episode of (a), and (d) is a zoom-in at 4th episode of (b)

Topology Evaluation

To evaluate the topology maintenance, we define a measure T to monitor the change of the number of neighbors for each robot. The topology of the network is evaluated based on the following algorithm.

We see that if $T = 0$ the topology of the network does not change, and if $T > 0$ the topology of the network changes. From the result in Figure 5.16 we can see that for our proposed cooperative learning algorithm the topology of the network does not change after 3 training episodes while for the independent learning algorithm the topology changes in all training episodes. Note that in Figure 5.16 (*right*) the topology is only changed when the prey have to avoid the obstacles, and it is maintained when they are avoiding the predator.

for *each mobile sensor i* **do**

if $|N_i^{\alpha}|$ *changes* **then**
Topology: $T_i = abs(|N_i^{\alpha}(k)| - |N_i^{\alpha}(k-1)|)$ (k is time step or iteration)

else if $|N_i^{\alpha}|$ *does not change, but indices of* $|N_i^{\alpha}|$ *change* **then**
Topology: T_i = number of index changes

else
Topology: $T_i = 0$

For the whole network :Topology: $T = \sum_{i=1}^{n} T_i$

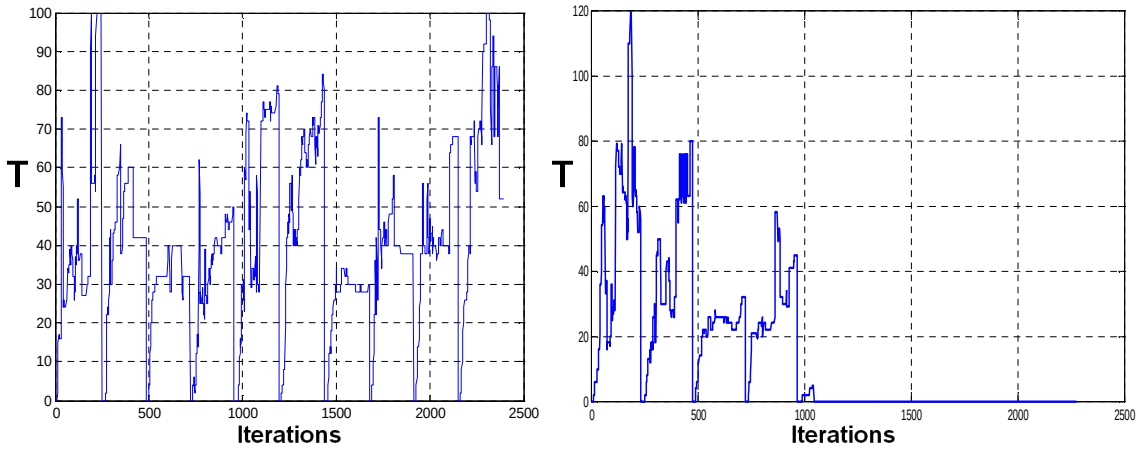


Figure 5.16: Topology evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

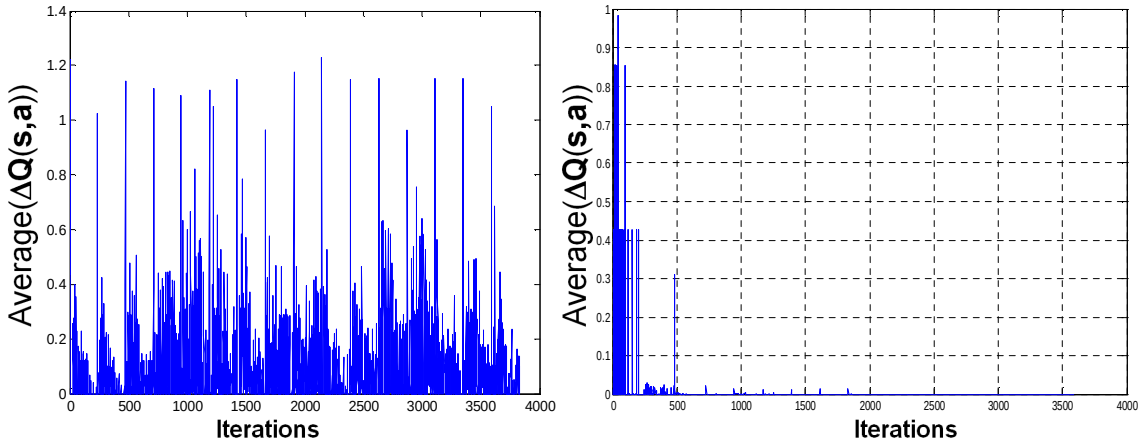


Figure 5.17: Convergence of Q values for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

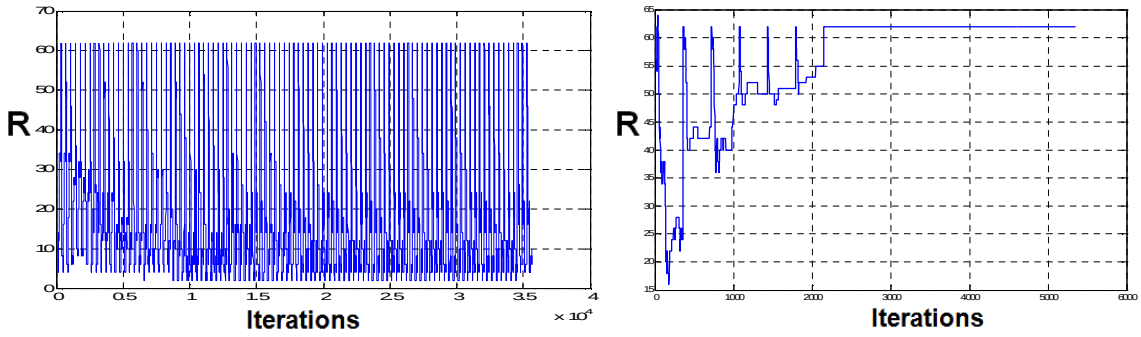


Figure 5.18: Global reward evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

Convergence of Q values Evaluation

The convergence of the proposed system is evaluated based on the average values of the $\Delta\mathbf{Q}(\mathbf{s},\mathbf{a})$. According Theorem 4 if the average of $\Delta\mathbf{Q}(\mathbf{s},\mathbf{a})$ goes to zero the proposed system is stable, otherwise it is not stable. From the result in Figure 5.17 we can see that for our proposed cooperative learning algorithm the average of $\Delta\mathbf{Q}(\mathbf{s},\mathbf{a})$ goes to zero after 2000 iterations while for the independent learning algorithm it does not converge to zero.

Reward Evaluation

The global reward is computed as $R = \sum_{i=1}^{n} r_i$. From the result in Figure 5.18 with our proposed cooperative learning algorithm we can obtain a maximum global reward with a value of 62 after about 2000 iterations, but with the independent learning algorithm the global reward does not converge to a stable value.

5.6 Summary

We proposed a hybrid system that integrates flocking control and reinforcement learning to allow mobile sensors to behave intelligently in continuous space. Reinforcement learning is developed based on cooperative Q learning and Majority Action Following algorithm (MAF). We evaluated the proposed hybrid system in the case of multiple mobile sensors learning to avoid predator. We showed that the proposed cooperative Q learning allows the network to find out the effective joint action more quickly than the independent Q learning. This also allows the network to maintain its topology and connectivity while avoiding the predator. Both simulation and experiment results are collected to demonstrate the effectiveness of our proposed system.

CHAPTER 6

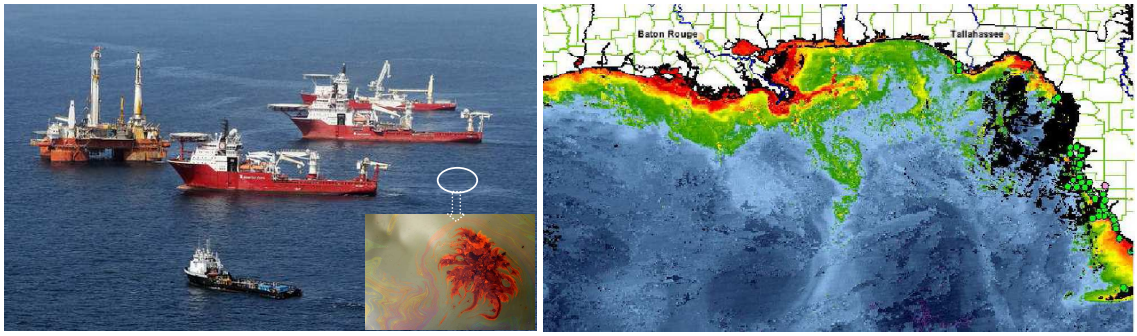
COOPERATIVE AND ACTIVE SENSING FOR MSNs BASED ON DISTRIBUTED CONSENSUS

In this chapter, autonomous mobile sensor networks are deployed to measure a scalar field and build its map. We develop a novel method for multiple mobile sensor nodes to build this map using noisy measurements. Our method consists of three parts. First, we develop a distributed sensor fusion algorithm by integrating two different distributed consensus filters to achieve cooperative sensing among sensor nodes. Second, we use the distributed flocking control algorithm to drive the center of the mobile sensor formation to track the desired paths. Third, we build a path planning strategy to obtain a complete coverage of the field. Simulation results are conducted to demonstrate our proposed method.

This chapter is organized as follows. Section 6.1 presents the introduction of this chapter. Section 6.2 presents the models of the scalar field and the measurement of each sensor node, as well as the problem formulation. Section 6.3 presents the distributed consensus filters and the distributed sensor fusion algorithm for building a map of the unknown scalar field. Section 6.4 describes the path planning strategy for complete coverage of the scalar field. Section 6.5 presents cooperative and active sensing algorithms for improving the confidence performance. Section 6.6 shows simulation results. Finally, Section 6.7 concludes this chapter.

6.1 Introduction

Measuring and exploring an unknown field of interest have attracted much attention of environmental scientists and control engineers [11, 59, 60, 13, 125, 12]. They have numer-



(a) (b)

Figure 6.1: (a) Evacuation: Ships and rig workers evacuate the oil spill area as Tropical Storm Bonnie approaches the region (Photo by Mario Tama/Getty Images). (b) The estimated field of chlorophyll generated by the harmful algal blooms observation system [2] by the National Oceanic and Atmospheric Administration (NOAA), (Photo courtesy of NOAA).

ous applications including environmental monitoring [8], and oil spill and toxic-chemical plume tracing [9, 10] (see Figure 6.1). Because the scalar field is often distributed across a large area, we need many sensors to cover the field if the sensors are mounted at fixed locations. MSNs in which sensors can move together and take measurements along their motion trajectories are ideal candidates for such missions.

In order to create the map of the scalar field, one of the important research problems in MSNs is to achieve cooperative sensing among sensors in a distributed fashion. Development of a novel cooperative sensing algorithm based on distributed estimation and control algorithms for MSNs is very challenging. The estimation and control have to be performed in each sensor node using only local information, while as a whole they exhibit collective intelligence and achieve a global goal. In a resource-constrained multi-agent system, the communication range and sensing range of each agent are small compared to the size of a surveillance region. Hence, agents cannot accomplish the mission without an effective flocking control and path planning strategy.

In this chapter, the problems of cooperative sensing and cooperative motion control

are addressed. Our work has three parts. First, we develop a distributed sensor fusion algorithm by integrating two different distributed consensus filters to achieve cooperative sensing among sensor nodes. In this algorithm, each sensor node obtains measurements from itself and its neighboring sensor nodes within its communication range. Each mobile sensor node will then iteratively update the estimate of the scalar field. Second, we use a distributed flocking control algorithm to drive the center of the mobile sensor formation to track the desired paths. Third, we build a path planning strategy to obtain a complete coverage of the field. From this cooperative sensing framework we extend to active sensing in order to achieve better sensing performance.

6.2 Scalar Field and Measurement Modeling and Problem Statement

6.2.1 Model of the Scalar Field

We model the scalar field of interest as

$$F = \Theta\Phi^T, \tag{6.1}$$

here $\Theta = [\theta_1, \theta_2, ..., \theta_K]$, and $\Phi = [\phi_1, \phi_2, ..., \phi_K]$. We can rewrite Equation (6.1) as

$$F = \sum_{j=1}^{K} \theta_j \phi_j, \tag{6.2}$$

here ϕ_j is a function representing a density distribution, and θ_j is the weight of the density distribution of the function ϕ_j.

We can model the function ϕ_j as a multiple variate Gaussian distribution (other distribution functions such as Poisson, Student, Cauchy distributions, ..., can also be used):

$$\phi_j = \frac{1}{\sqrt{det(C_j)(2\pi)^2}} e^{\frac{1}{2}(x-\mu_x^j)C_j^{-1}(y-\mu_y^j)^T}, j \in [1, 2, ..., K].$$

here $[\mu_x^j \ \mu_y^j]$ is the mean of the distribution of function ϕ_j, and C_j is covariance matrix (positive definite) and it is represented by:

$$C_j = \begin{bmatrix} (\sigma_x^j)^2 & c_j^o \sigma_x^j \sigma_y^j \\ c_j^0 \sigma_x^j \sigma_y^j & (\sigma_y^j)^2 \end{bmatrix},$$

where c_j^0 is a correlation factor.

6.2.2 Measurement Model

We partition the scalar field F into a grid of C cells. Each sensor i makes an observation (measurement) of the scalar field at cell k ($k \in \{1,2,...,C\}$) at time step t based on the following equation

$$m_i^k(t) = O_i^k(t)[F^k(t) + n_i^k(t)], \tag{6.3}$$

here $n_i^k(t)$ is the Gaussian noise with zero mean and variance $V_i^k(t)$ at time step t. We assume that n_i^k is uncorrelated noise which satisfies

$$Cov(n_i^k(s), n_i^k(t)) = \begin{cases} V_i^k, & if \quad s = t \\ 0, & otherwise, \end{cases}$$

here Cov is the covariance. $O_i^k(t)$ is the observability of sensor node i at cell k at time step t, and it is defined as

$$O_i^k(t) = \begin{cases} 1, & if \quad \|q_i(t) - q_c^k\| \leq r_i^s \\ 0, & otherwise, \end{cases} \tag{6.4}$$

here $q_i \in R^2$ is the position of sensor node i; $q_c^k \in R^2$ is the location of cell k at its center. This definition tells us that if cell k is inside the sensing range, r_i^s, of sensor node i then cell k can be measured or observed. Otherwise the observability is zero. Note that r_i^s can be the same for all sensors ($r_1^s = r_2^s = ... = r_n^s = r^s$) or different.

Each mobile sensor node makes an measurement at cell k corresponding to its position. We assume that the variance $V_i^k(t)$ is related to the distance between the sensor node i and the location of the measurement according to:

$$V_i^k(t) = \begin{cases} \frac{\|q_i(t) - q_c^k\|^2 + c_v}{(r_i^s)^2}, & if \quad \|q_i(t) - q_c^k\| \leq r_i^s \\ 0, & otherwise, \end{cases} \tag{6.5}$$

here c_v is the small positive constant between 0 and 1. The reason of introducing c_v is to avoid the variance $V_i^k(t)$ being zero when the distance $\|q_i(t) - q_c^k\|$ equals to zero.

6.2.3 Problem Formulation

Given the measurements of sensor node i and its neighbors at each cell of the scalar field F as modeled in Equation (6.3) (see Figure 6.2), our goal is to build the map for the scalar field F modeled by Equation (6.1).

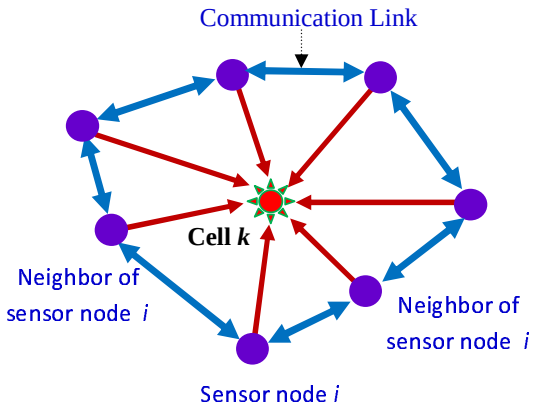


Figure 6.2: Illustration of the measurement model using multiple mobile sensor nodes.

6.3 Distributed Sensor Fusion Algorithm

6.3.1 Overall Approach

In this section we present a distributed sensor fusion algorithm to allow each sensor node to find out an estimate of the value at each cell of the scalar field based on its own measurement and its neighbor's measurements. Our algorithm has two phases. First, each sensor node finds an estimate of the value of the scalar field F at each cell at time step t. Second, each sensor node finds a final estimate of the value of the scalar field F at each cell during its movement. To achieve it, we develop two consensus filters. The consensus filter 1 is to find out an estimate of the value of the field F at each cell at time step t. Since each mobile sensor node makes its own measurement at each cell at time step t with its own weight (confidence), the consensus filter 2 is used to find out an agreement among these confidences.

At each time step t each mobile sensor node needs to find an estimate of the value of each cell based on consensus filter 1, and find an overall confidence of this estimate based on consensus filter 2. This process can be called the *spatial estimate phase*. Then, during the movement of each sensor node, it will have multiple spatial estimates of each cell associated with their own confidences. Hence, these spatial estimates are fused iteratively through the weighted average protocol, and this process can be called the *temporal estimate phase*. To summarize:

(1) *Spatial estimate phase:*

- Building a weighted average consensus filter (consensus filter 1) to find out an agreement of the estimates among the sensor nodes at each time step t,
- Building an average consensus filter (consensus filter 2) to find out an agreement of the weights (confidences) of the measurements among the sensor nodes at each time step t,

(2) *Temporal estimate phase:*

- Building a weighted average protocol to iteratively update the spatial estimates for sensor node i during its movement.

6.3.2 Distributed Consensus Filters

Consensus Filter 1

Distributed consensus [52, 53, 50, 51, 56, 55] is an important computational tool to achieve cooperative sensing. We consider distributed linear iterations of the following form

$$x_i^k(l+1) = w_{ii}^k(l)x_i^k(l) + \sum_{j\in N_i(t)} w_{ij}^k(l)x_j^k(l), \tag{6.6}$$

here l is iteration index. The initial condition for the state is given as $x_i^k(l=0) = m_i^k(t)$. The weight, $w_{ii}^k(l)$, is the self weight or vertex weight of each sensor to cell k, and $w_{ij}^k(l)$ is the edge weight between sensor i and sensor j. These weights will be discussed more later.

Our problem here is to estimate the value of the field F at each cell k at each time step t. Since each sensor node makes the observation at cell k at time step t based on its own confidence (weight), the consensus should converge to the weighted average of all observations (measurements) at cell k from all sensor nodes in the network. This weighted average is the estimate of the value at cell k at time step t, and it is computed as:

$$E^k(t) = \frac{\sum_{i=1}^{n} w_{ii}(t) m_i(t)}{\sum_{i=1}^{n} w_{ii}(t)}. \tag{6.7}$$

If Equation (6.6) converges we have $E_1^k = E_2^k = ... = E_n^k = E^k$. Therefore, our goal is to let

$$lim_{l\to\infty}(x_i^k(l) - E^k(t)) \to 0 \tag{6.8}$$

We can write Equation (6.8) in the matrix form

$$lim_{l\to\infty}\mathbf{x}^k(l) = E^k(t)\mathbf{1} \tag{6.9}$$

here $\mathbf{x}^k(l) = [x_1^k(l),\ x_2^k(l),\ ...,\ x_n^k(l)]_{n\times 1}^T$, and $\mathbf{1} = [1,\ 1, ...,\ 1]_{n\times 1}^T$.

We can also write Equation (6.6) in the matrix form

$$\mathbf{x}^k(l+1) = \mathbf{w}^k(l)\mathbf{x}^k(l) \tag{6.10}$$

with initial condition $\mathbf{x}^k(0) = \mathbf{m}^k(t)$, and $\mathbf{m}^k(t) = [m_1^k(t),\ m_2^k(t),\ ...,\ m_n^k(t)]_{n\times 1}^T$.

To make Equation (6.6) converge to $E^k(t)$ we need

$$lim_{l\to\infty}\mathbf{w}^k(l+1) = \frac{1}{n}\mathbf{1}\mathbf{1}^T \tag{6.11}$$

In order to achieve this we need to ensure that the sum of all weights including the vertex and edge weights at each node equals to 1, or

$$w_{ii}^k(l) + \sum_{j\in N_i(t)} w_{ij}^k(l) = 1. \tag{6.12}$$

To satisfy this, we have the following designs of weight.

Weight Design 1:

From Equation (6.12) the vertex weight at node i is obtained as

$$w_{ii}^k(l) = 1 - \sum_{j \in N_i(t)} w_{ij}^k(l). \tag{6.13}$$

here, $w_{ij}^k(l)$ is defined as

$$w_{ij}^k(l) = \frac{c_1^w}{V_i^k(t) + V_j^k(t)}, i \neq j, j \in N_i(t), \tag{6.14}$$

here, c_1^w is a constant. If both sensor nodes i and j do not observe cell k ($O_i^k(t) = O_j^k(t) = 0$) then to avoid dividing by zero the edge weight $w_{ij}^k(l)$ is set to zero.

Therefore we have the following form of weight design

$$w_{ij}^k(l) = \begin{cases} \frac{c_1^w}{V_i^k(t)+V_j^k(t)}, & if \quad i \neq j, j \in N_i(t), \\ 1 - \sum_{j \in N_i(t)} w_{ij}^k(l), & if \quad i = j, \\ 0, & otherwise. \end{cases} \tag{6.15}$$

Now we need to find c_1^w to satisfy Equation (6.12). We know that

$min(V_i^k(t)) = min(\frac{\|q_i(t) - q_c^k\|^2 + c_v}{(r_i^s)^2}) = \frac{c_v}{(r_i^s)^2}$ if $\|q_i(t) - q_c^k\| = 0$. Hence we have

$$min(V_i^k(t)) + min(V_j^k(t)) = \begin{cases} \frac{2c_v}{(r^s)^2}, & if \quad r_i^s = r_j^s = r^s, \\ \frac{c_v}{(r_i^s)^2} + \frac{c_v}{(r_j^s)^2}, & otherwise. \end{cases} \tag{6.16}$$

To satisfy Equation (6.12) we need

$$0 < \sum_{j \in N_i(t)} w_{ij}^k(t) < 1 \Rightarrow 0 < \sum_{j \in N_i(t)} \frac{c_1^w}{V_i^k(t) + V_j^k(t)} < 1$$

or,

$$0 < c_1^w < \frac{V_i^k(t) + V_j^k(t)}{|N_i(t)|}, \tag{6.17}$$

here $|N_i(t)|$ is the number of neighbors of sensor node i at time t, and from (6.16) and (6.17) we can select c_1^w as

$$\begin{cases} 0 < c_1^w < \frac{2c_v}{(r_i^s)^2 |N_i(t)|}, & if \quad r_i^s = r_j^s = r^s, \\ 0 < c_1^w < \frac{1}{|N_i(t)|}(\frac{c_v}{(r_i^s)^2} + \frac{c_v}{(r_j^s)^2}), & otherwise. \end{cases} \tag{6.18}$$

Weight Design 2:

From Equation (6.12) by assigning the same value to all edge weights we obtain:

$$w_{ij}^k(l) = \frac{1 - w_{ii}^k(l)}{|N_i(t)|}. \tag{6.19}$$

here, $w_{ii}^k(l)$ is defined as

$$w_{ii}^k(l) = \frac{c_2^w}{V_i^k(t)}, \tag{6.20}$$

where c_2^w is a constant. If sensor node i does not observe cell k $(O_i^k(t) = 0)$ then the vertex weight $w_{ii}^k(l)$ is set to zero.

Therefore we have the following weight design

$$w_{ij}^k(l) = \begin{cases} \frac{c_2^w}{V_i^k(t)}, & if \quad i = j, \\ \frac{1 - w_{ii}^k(l)}{|N_i(t)|}, & if \quad i \neq j, j \in N_i(t), \\ 0, & otherwise. \end{cases} \tag{6.21}$$

Now we discuss how to select the constant c_2^w. In order to satisfy Equation (6.12) we need the following condition:

$$0 < \frac{c_2^w}{V_i^k(t)} < 1. \tag{6.22}$$

Since $min(V_i^k(t)) = \frac{c_v}{(r_i^s)^2}$ when $\|q_i(t) - q_c^k\| = 0$, we have:

$$0 < \frac{c_2^w}{\frac{c_v}{(r_i^s)^2}} < 1 \Rightarrow 0 < c_2^w < \frac{c_v}{(r_i^s)^2}. \tag{6.23}$$

Finally, the consensus filter 1 (CF1) is summarized as

$$CF1 \quad : \quad x_i^k(l+1) = w_{ii}^k(l) x_i^k(l) + \sum_{j \in N_i(t)} w_{ij}^k(l) x_j^k(l)$$

$$w_{ij}^k(l) = \begin{cases} \frac{c_1^w}{V_i^k(t) + V_j^k(t)}, & if \quad i \neq j, j \in N_i(t), \\ 1 - \sum_{j \in N_i(t)} w_{ij}^k(l), & if \quad i = j, \\ 0, & otherwise. \end{cases}$$

or,

$$w_{ij}^k(l) = \begin{cases} \frac{c_2^w}{V_i^k(t)}, & if \quad i = j, \\ \frac{1 - w_{ii}^k(l)}{|N_i(t)|}, & if \quad i \neq j, j \in N_i(t), \\ 0, & otherwise. \end{cases} \tag{6.24}$$

Consensus Filter 2

Since each sensor node has its own confidence of the measurement of the value of the scalar field at each cell at each time step t we need to find an agreement among the confidences of sensor nodes. The consensus algorithm 2 is introduced to find the overall confidence from each time step t. This overall confidence is the estimated weight, $W_i^k(t)$, of the weighted average protocol as shown in Equation (6.39).

Let $y_i^k(l=0)$ be the confidence of the measurement of the value of the scalar field at cell k at each time step t for sensor node i, or $y_i^k(l=0) = w_{ii}^k(t)$. Let $y_j^k(l=0)$ be the confidence of the measurement of the value of the scalar field at cell k at each time step t for sensor node j with $j \in N_i(t)$, or $y_j^k(l=0) = w_{jj}^k(t)$. Then, we have the following consensus filter

$$y_i^k(l+1) = w_{ii}^k(l) y_i^k(l) + \sum_{j \in N_i(t)} w_{ij}^k(l) y_j^k(l), \tag{6.25}$$

In this consensus filter, we use the Metropolis weight [53] as

$$w_{ij}^k(l) = \begin{cases} \frac{1}{1+max(|N_i(t)|,|N_j(t)|)}, & if \quad i \neq j, j \in N_i(t), \\ 1 - \sum_{j \in N_i(t)} w_{ij}^k(l), & if \quad i = j, \\ 0, & otherwise. \end{cases} \tag{6.26}$$

6.3.3 Convergence Analysis

In this subsection we analyze the convergence of the Consensus Filter 1.

First, let us define the weight matrix for the whole network as

$$\mathbf{w}^k = \begin{bmatrix} w_{11}^k & w_{12}^k & \dots & w_{1n}^k \\ w_{21}^k & w_{22}^k & \dots & w_{2n}^k \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ w_{n1}^k & w_{n2}^k & \dots & w_{nn}^k \end{bmatrix}_{n \times n}. \tag{6.27}$$

Based on our **Weight Design 1 and 2**, the matrix $\mathbf{w}^k$ has the following properties:

(P1) $w_{ij}^k = 0$ *if* $j \notin N_i$.

(P2) All elements, w_{ij}^k, $i = 1,...,n$ *and* $j \in N_i$, *of the matrix* $\boldsymbol{w}^k$ *satisfy* $0 < w_{ij}^k < 1$.

(P3) Sum of all elements in each row of the matrix $\boldsymbol{w}^k$ *equals to 1.*

With the definition of the weight matrix $\mathbf{w}^k$, we can expand Equation (6.6) to n mobile sensors into space representation as follows

$$\mathbf{x}^k(l+1) = \mathbf{w}^k\mathbf{x}^k(l), \tag{6.28}$$

or we have

$$\mathbf{x}^k(l+1) - \mathbf{x}^k(l) = [\mathbf{w}^k - \mathbf{I}]\mathbf{x}^k(l), \tag{6.29}$$

here $\mathbf{I}$ is the identity matrix.

$$\Delta\mathbf{x}^k = [\mathbf{w}^k - \mathbf{I}]\mathbf{x}^k(l), \tag{6.30}$$

here $\Delta\mathbf{x}^k = \mathbf{x}^k(l+1) - \mathbf{x}^k(l)$. We can also rewrite Equation (6.30) into a continuous fashion

$$\dot{\mathbf{x}}^k = \mathbf{A}^k\mathbf{x}^k, \tag{6.31}$$

here $\mathbf{A}^k = \mathbf{w}^k - \mathbf{I}$.

Theorem 5. *Given any connected network, and by applying the Consensus Filter 1 as defined in Equation (6.6) associated with the* ***Weight Design 1*** *or* ***2*** *as defined in Equations (6.15) and (6.21), respectively, the system (6.31) is stable, or* $\Delta\boldsymbol{x}^k$ *converges to zero.*

Proof: The system (6.31) is a linear time-invariant system or autonomous system. Therefore to show this system stable we need to show that matrix $\mathbf{A}^k$ is a Hurwitz matrix, or all of the roots of the characteristic equation have negative real parts [123, 124, 126].

Given a matrix $B = [b_{ij}]_{n\times n}$ of the autonomous system $\dot{x} = Bx$ we have the following theorem:

Theorem 6 (*Liao et. al* [126]): *If the following conditions*:

(C1) $b_{ii} < 0$ $(i = 1,2,...,n)$ *and* $det(B) \neq 0$; *and*

(C2) there exist constants $c_i > 0$ $(i = 1,2,...,n)$ *such that*

$c_j b_{jj} + \sum_{i=1,i\neq j}^{n} |c_i||b_{ij}| < 0$, $(j = 1,2,...,n)$

are satisfied, then B is a Hurwitz matrix.

First let us write matrix $\mathbf{A}^k$ in details as

$$\mathbf{A}^k = [a_{ij}] = \begin{bmatrix} w_{11}^k - 1 & w_{12}^k & \ldots & w_{1n}^k \\ w_{21}^k & w_{22}^k - 1 & \ldots & w_{2n}^k \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ w_{n1}^k & w_{n2}^k & \ldots & w_{nn}^k - 1 \end{bmatrix}_{n \times n} . \tag{6.32}$$

Based on Theorem 6 we can check our matrix $\mathbf{A}^k$. We clear see that the condition $C1$ is easily satisfied because all diagonal elements w_{ii}^k of the matrix $\mathbf{w}^k$ satisfy $0 < w_{ii}^k < 1$ (see property $P2$ of the matrix $\mathbf{w}^k$). Therefore we obtain that all diagonal elements of the matrix $\mathbf{A}^k$ satisfy $a_{ii} = w_{ii}^k - 1 < 0$.

For the condition $C2$, since the sum of all elements in each row of the matrix $\mathbf{w}^k$ equals to one (see property $P3$ of the matrix $\mathbf{w}^k$), we can easily find the constants c_i to let $c_j a_{jj} + \sum_{i=1, i \neq j}^{n} |c_i||a_{ij}| < 0$. Therefore we can conclude that $\mathbf{A}^k$ is a Hurwitz matrix, or the proposed system (6.31) is stable.

As one example we can show that $\mathbf{A}^k$ is a Hurwitz matrix by showing the roots of the characteristic equation (6.33) in the case of 2×2 dimension of the matrix $(\lambda + 1)\mathbf{I} - \mathbf{w}^k$.

We have the following characteristic equation for the system (6.31) as:

$$det(\lambda \mathbf{I} - \mathbf{A}) = det(\lambda \mathbf{I} - \mathbf{w}^k + \mathbf{I}) = det((\lambda + 1)\mathbf{I} - \mathbf{w}^k) = 0 \tag{6.33}$$

here

$$(\lambda + 1)\mathbf{I} - \mathbf{w}^k = \begin{bmatrix} \lambda + 1 - w_{11}^k, & -w_{12}^k & \ldots & -w_{1n}^k \\ -w_{21}^k, & \lambda + 1 - w_{22}^k & \ldots & -w_{2n}^k \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ -w_{n1}^k, & -w_{n2}^k & \ldots & \lambda + 1 - w_{nn}^k \end{bmatrix} . \tag{6.34}$$

For the case of 2×2 dimension of the matrix $(\lambda+1)\mathbf{I} - \mathbf{w}^k$ we have

$$det((\lambda+1)\mathbf{I} - \mathbf{w}^k) = \lambda^2 + \lambda(2 - w_{11}^k - w_{22}^k) + 1 - w_{11}^k - w_{22}^k + w_{11}^k w_{22}^k - w_{21}^k w_{12}^k \quad (6.35)$$

From the the property $P3$ of the matrix $\mathbf{w}^k$ we have $w_{12}^k = 1 - w_{11}^k$ and $w_{21}^k = 1 - w_{22}^k$. Plug these w_{12}^k and w_{21}^k into Equation (6.35) we obtain

$$det((\lambda+1)\mathbf{I} - \mathbf{w}^k) = \lambda[\lambda + (2 - w_{11}^k - w_{22}^k)] = 0. \quad (6.36)$$

Equation (6.36) has two roots $\lambda_1 = 0$, and $\lambda_2 = -2 + w_{11}^k + w_{22}^k < 0$ since $0 < w_{11}^k < 1$ and $0 < w_{22}^k < 1$. This finishes the proof for Theorem 5.

6.3.4 Distributed Fusion Algorithm

From the consensus filters 1 and 2 we would like to design a distributed sensor fusion algorithm to allow each sensor node to on-line estimate the value of the scalar field at each cell based on its own measurement and its neighbor's measurements. The overall design of such a distributed sensor fusion algorithm is shown in Figure 6.3. In this algorithm, we have two phases running at the same time. In the spatial estimate phase, the measurements of each sensor node and its neighbors at cell k at time step t are inputs of the consensus filter 1. Then, the output of this consensus is the estimate of the value of the scalar field F at cell k at time step t. In the temporal estimate phase, the confidences (weights) of the measurements of each sensor node and its neighbors at cell k at time step t are inputs of the consensus filter 2. Then, the output of this consensus is the estimate of the confidence of the measurement of the scalar field at cell k at time step t. During the movement of sensor nodes , each sensor obtain several spatial estimates of the value at cell k associated with its own confidence, hence the final estimate is iteratively updated based on these spatial estimates via the weighted average protocol. The detail to implement this algorithm is shown in Algorithm 6.

Algorithm 6: Distributed Sensor Fusion Algorithm

Input: the weight $w_{ii}^k(t)$, and the measurement of sensor node i and its neighbors to cell k, $m_i^k(t)$ and $m_j^k(t)$.

Output: the final estimate of the cell k, $\overline{E}_i^k(1:t_l)$

for *each time step t* **do**

for *each sensor node i* **do**

Step1: Make a measurement (observation) $m_i^k(t)$ to cell k if $\|q_i(t) - q_c^k\| \leq r_i^s$

Sensor node i obtains the measurements of cell k from its neighbors and itself

for *each iteration l* **do**

Sensor node i runs the consensus (6.6) with $w_{ij}^k(l)$ is defined in (6.15) or (6.21)

$x_i^k(l=0) = m_i^k(t); x_i^k(l+1) = w_{ii}^k(l)x_i^k(l) + \sum_{j\in N_i(t)} w_{ij}^k(l)x_j^k(l)$

Sensor node i runs the consensus (6.25) with $w_{ij}^k(l)$ is defined in (6.26)

$y_i^k(l=0) = w_{ii}^k(t); y_i^k(l+1) = w_{ii}^k(l)y_i^k(l) + \sum_{j\in N_i(t)} w_{ij}^k(l)y_j^k(l)$

end

Step2: Obtain the estimate of cell k after running the consensus

Let l_c be a time step that both consensus filters converge, then we have:

$E_i^k(t) = x_i^k(l_c); W_i^k(t) = y_i^k(l_c)$

Step3: Update process to find the final estimate of the value of the scalar field at cell k: - Update weight (confidence):

$$\overline{W}_i^k(t) = W_i^k(t-1) + W_i^k(t-2) + \ldots + W_i^k(0) \tag{6.37}$$

- Update the final estimate based on the weighted average protocol:

$$\overline{E}_i^k(t=0) = E_i^k(t=0) = x_i^k(l_c) \tag{6.38}$$

$$\overline{E}_i^k(t) = \frac{\overline{W}_i^k(t-1)\overline{E}_i^k(t-1) + W_i^k(t)E_i^k(t)}{\overline{W}_i^k(t-1) + W_i^k(t)} \tag{6.39}$$

end

end

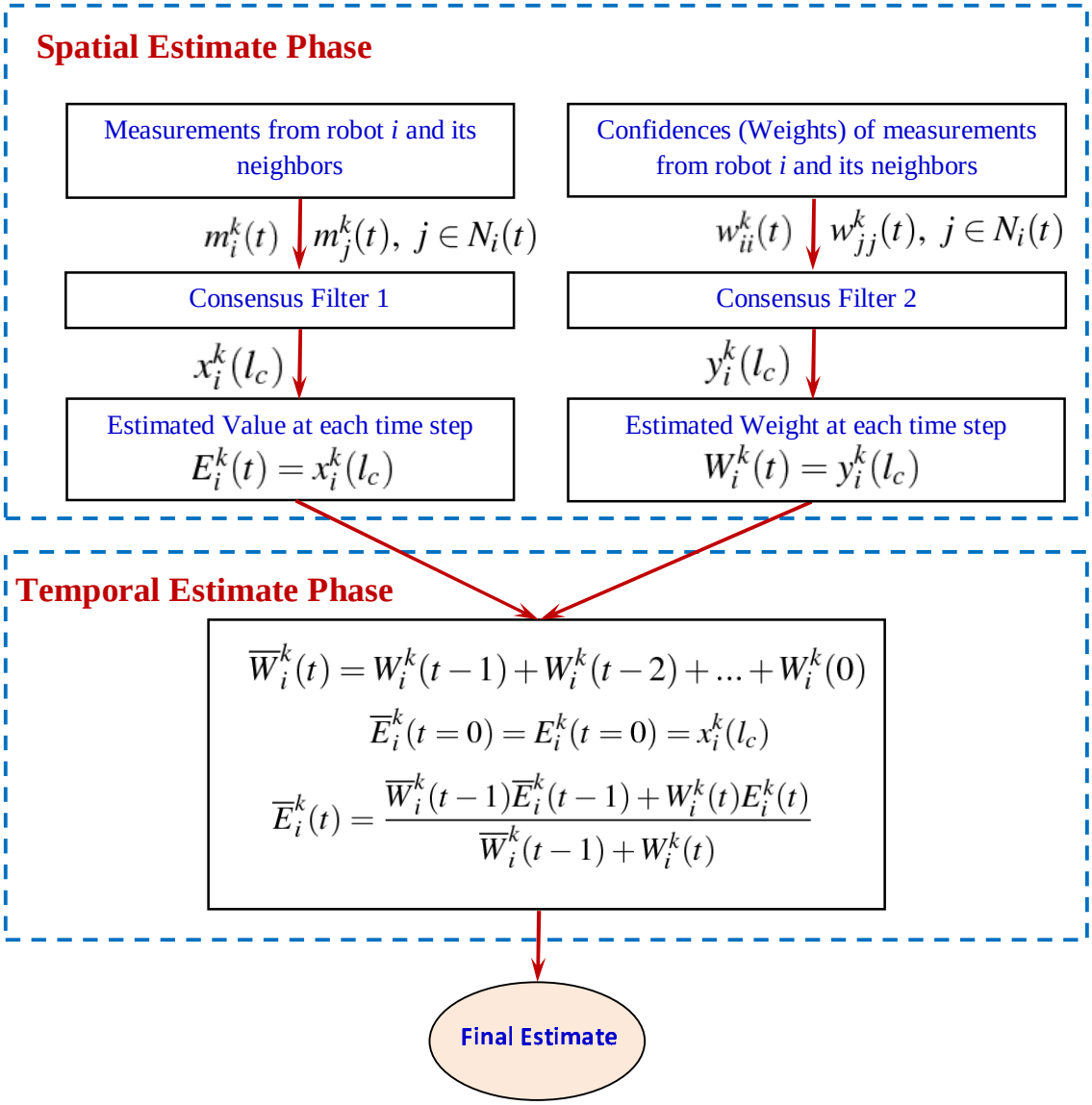


Figure 6.3: Framework of distributed sensor fusion algorithm based two different consensus filters.

6.4 Path Planning Strategy

In this section we present the path planning strategy to ensure that the MSN can cover the entire scalar field.

The flocking control algorithm used to control the center of the mobile sensor node formation to track the desired paths is presented as

$$\begin{aligned} u_i &= f_i^{\alpha} + f_i^{t} \\ &= c_1^{\alpha} \sum_{j \in N_i(t)} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i(t)} a_{ij}(q)(p_j - p_i) \\ &\quad -c_1^{t}(q_i - q_t) - c_2^{t}(p_i - p_t) \\ &\quad -c_1^{mc}(\overline{q}_{(N_i(t)\cup\{i\})} - q_t) - c_2^{mc}(\overline{p}_{(N_i(t)\cup\{i\})} - p_t). \end{aligned} \tag{6.40}$$

Based on the flocking control algorithm (6.40), all mobile sensor nodes can form a lattice formation as shown in Figure 2.6 in Chapter 2, and the center of mass (CoM) of the network as defined in Equation (2.41) can track the leader (q_t, p_t) successfully as shown in Figure 2.7 in Chapter 2. Since the network can track the leader, to allow the network to cover the entire scalar field we only need to design the path of the leader so that the field is fully covered. We assume that the leader knows the total number of sensor nodes in the network. Then based on the distance between sensor nodes (d^{α}), the leader can compute the size of the network. Then, our multi-robot path planning problem becomes a single robot path planning problem. There are some typical types of motion planning for a mobile robot to have complete coverage of the field of interest such as *boustrophedron* motion or wall-following motion [127]. In this chapter we plan the leader motion using a typical *boustrophedron* motion [127]. The result of the path planning is shown in Figure 6.4.

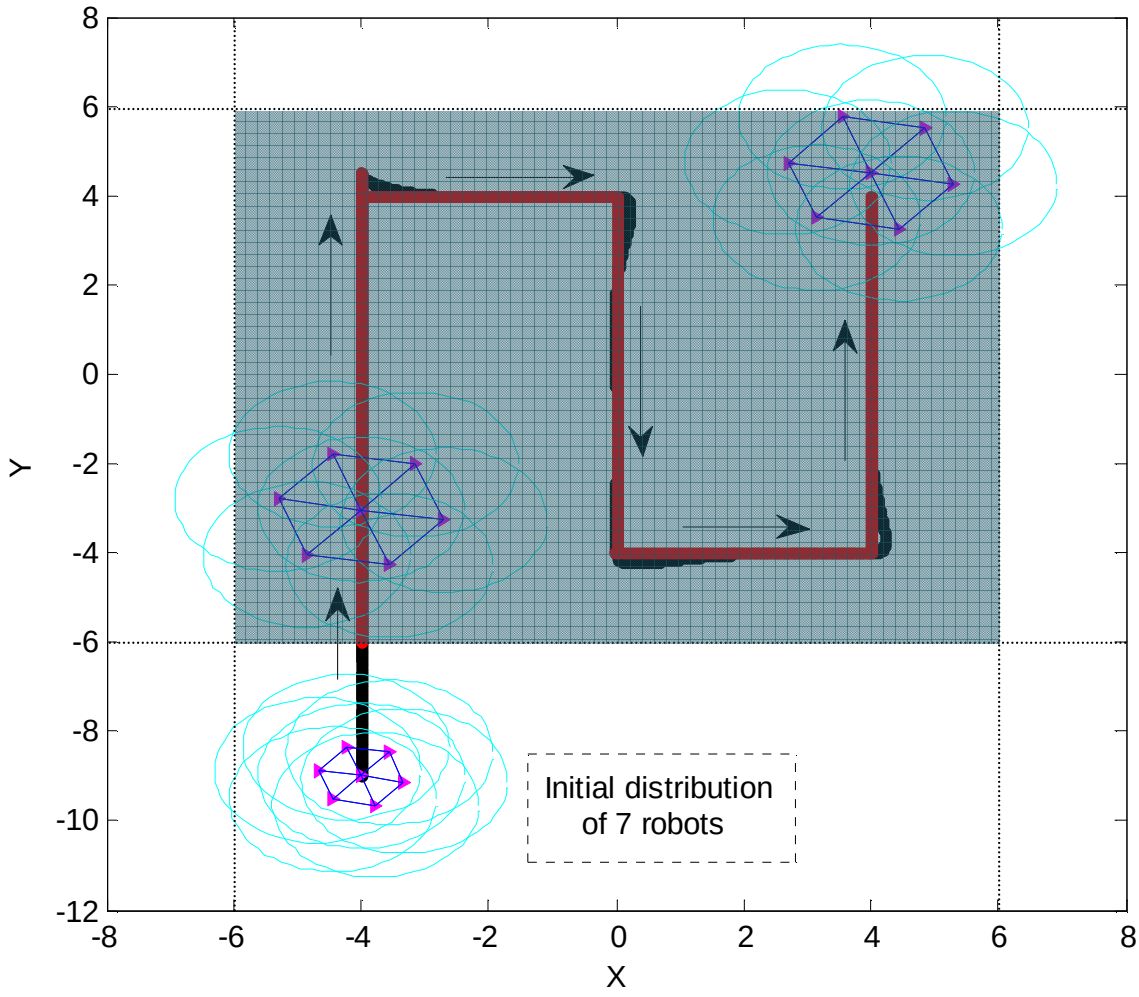


Figure 6.4: Seven mobile sensor nodes flock together and cover the scalar field (the filled square: 12×12). The motion path (red color) is generated by the leader, and the CoM (black/darker color) of the network tracks the leader with small overshoots at sharp change points of the path.

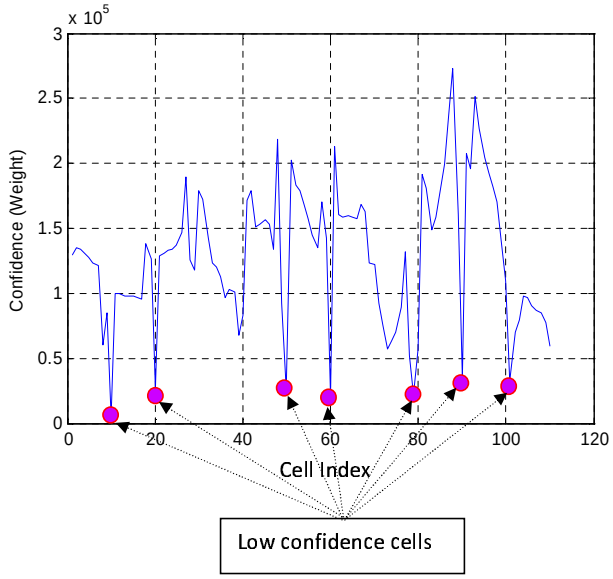


Figure 6.5: The confidence at each cell of the scalar field F.

6.5 Cooperative and Active Sensing

6.5.1 Introduction

In this section we aim to extend our cooperative sensing framework to cooperative and active sensing in which the mobile sensors have the ability to adjust their movements to adapt to the environments so that they can improve the sensing performance in a distributed fashion.

The mission of measuring and exploring an unknown field for building its map requires cooperative and active sensing among mobile sensor nodes. In many scalar field mapping applications such as temperature field mapping, search and rescue, there is a need to achieve a certain level of confidence regarding the estimates at each location. As we can see from Figure 6.5, using the normal cooperative sensing algorithm, some cells have very low confidence. This means that we may miss important information at these locations (cells). For example, in search and rescue operation the MSN may miss the objects at the locations where the confidence of the estimate is not sufficient. This motivates us to develop novel active sensing algorithms which can integrate both sensing ability and

motion control to adapt to the environments in order to improve the sensing performance. Each mobile sensor node needs to cooperate with its neighbors to adjust its configurations such as its relative location to the neighbors. By this way each agent can actively build the confidence map of the environments. In addition, the estimation and control have to be performed on-line in order to adapt to the changes of the environments.

Active sensing in MSNs has recently attracted many researchers in control engineering [128, 129, 130, 57, 58, 131, 61, 132, 133]. The early work on this technique can be found in [57, 58] where the active sensing algorithm for MSNs to estimate the state of dynamic targets is proposed. The localization and tracking tasks of dynamic targets are addressed. To achieve active sensing, the mobility of sensing agents is utilized to improve the sensing performance. However, the gradient controller for active sensing is designed in a centralized way. To relax this limitation, a distributed gradient controller is proposed in [61]. This controller is designed by constructing a dynamic average consensus estimator and using a one-hop neighbor for communication so that both formation control and cooperative sensing are integrated in order to improve the sensing performance.

Besides the developed active sensing algorithms for target estimation, the active sensing algorithms for source seeking and radiation mapping have been developed [134, 135, 136, 137, 132, 133]. The problem of source seeking is first addressed in [134], and then it is thoroughly studied in [135, 136, 137] for the case when direct gradient information of the measured quantity is unavailable. Specifically, Pang and Farrell [135] address chemical plume source localization by constructing a source likelihood map based on Bayesian inference methods. Mesquita *et. al* [136] introduce source seeking behavior without direct gradient information by mimicking E. Coli bacteria. Mayhew *et. al* [137] propose a hybrid control strategy to locate a radiation source utilizing only radiation intensity measurements. Additionally, active sensing for radiation mapping is developed in [132, 133]. The control algorithm takes into account sensing performance as well as dynamics of the observed process therefore it can steer mobile sensors to locations where they maximize the information

content of the measurement data. However, in their work the confidence of estimates is not addressed.

Overall, to our best knowledge the existing works in the area of active sensing using MSNs mostly focus on target(s) tracking [128, 130, 57, 58, 61], sensor placement [131], source seeking [134, 135, 137, 136] and radiation mapping [132, 133]. The problem of scalar field estimation and mapping based on multi-agent distributed active sensing has not been investigated yet.

Our goal is to develop a cooperative and active sensing algorithm for MSNs so that each sensor only interacts with its neighbors and uses the local observation to automatically adjust the configuration of the MSNs such as relative location among sensors, orientation and focal length of the sensors (camera), etc. to adapt to the environments and improve the sensing performance. To achieve this goal the controller should be designed via the real-time feedback of the sensing performance. By this way the controller can steer the mobile sensor to move to the expected locations of the field in order to improve the sensing quality. For simplicity, in this work we only focus on adjusting the relative location among sensors. Specifically, our problem focuses on how to control the movement of the mobile sensors to increase the confidence level on the estimates.

The cooperative and active sensing framework is depicted in Figure 6.6. In this figure, the controller designed via the real-time feedback of the confidence of estimates controls the mobile sensors to first form a quasi lattice network and then move the MSN to expected locations in order to achieve better sensing performance.

To realize the controller, we have two approaches: Distance Controller and Potential Controller.

6.5.2 Distance Controller

In this section, we consider to increase the confidence level of estimates over the lower bound. We design the distance controller to control the size of the network. The main idea

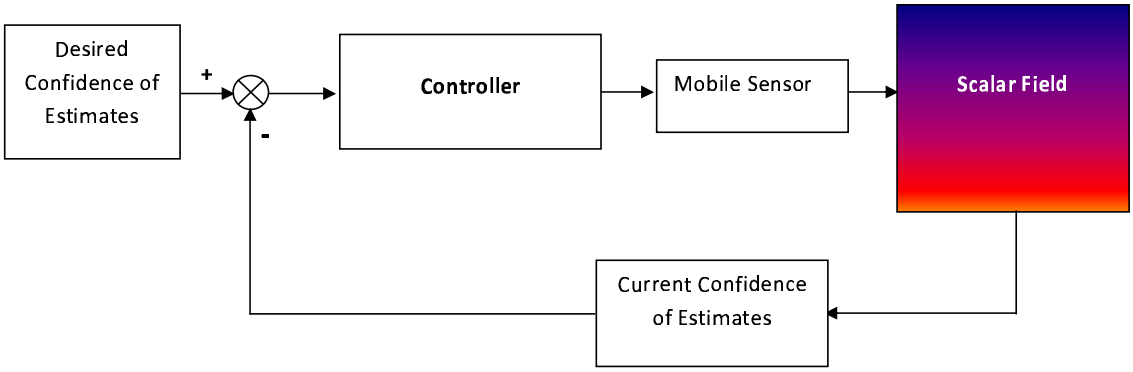


Figure 6.6: Framework of active sensing via confidence feedback

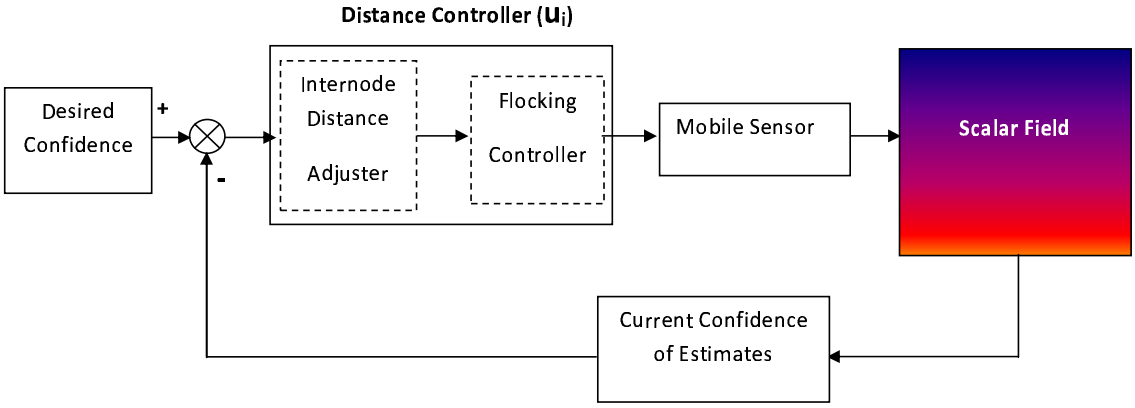


Figure 6.7: Diagram of active sensing based on the distance controller via confidence feedback

of designing this controller (see Figure 6.7) is to shrink the network's size if any covered cell of the scalar field has its confidence lower than the desired one, and recover to the original size of the network if all covered cells have sufficient confidence. This approach is quite straight forward since shrinking the size of the network brings the mobile sensors closer to the low confidence cells, hence it can increase the confidence level of these cells. The distance controller is designed based on the flocking controller and the inter-node distance adjuster. Here the flocking controller was presented in previous chapters, therefore we only present the design of the inter-node distance adjuster.

Let $\overline{W}_d$ be a desired confidence of the estimates of all cells in the scalar field, so $\overline{W}_d$ is a vector of $1 \times C$ dimension. Here again C is a total number of cells in the field. Then, we

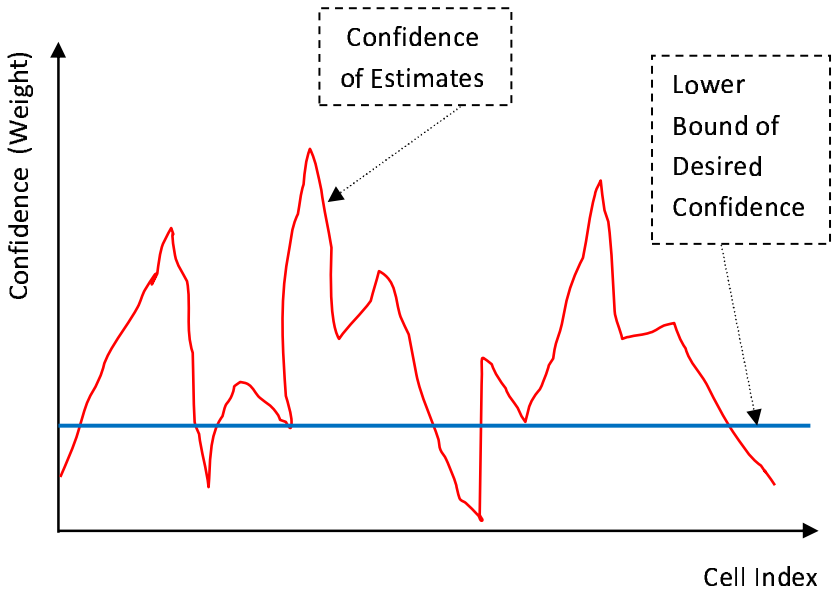


Figure 6.8: Illustration of confidence feedback for lower bound only.

can write $\overline{W}_d = [\overline{W}_d^1, \overline{W}_d^2, ..., \overline{W}_d^C]$.

In previous section we defined $\overline{W}_i^k(t)$ being a current accumulated confidence of the estimates of mobile sensor i at cell k. Therefore we have the accumulated confidence of all cells in the field as $\overline{W}(t) = [\overline{W}^1, \overline{W}^2, ..., \overline{W}^C]$. Note that if cell k is not covered by any sensor, the confidence $\overline{W}^k = 0$.

Let $\Delta_W(t) = \overline{W}_d - \overline{W}(t)$ be a difference between the current confidence and the desired one (see Figure 6.8). We can write $\Delta_W(t)$ as a vector form: $\Delta_W(t) = [\Delta_W^1(t), \Delta_W^2(t), ..., \Delta_W^C(t)]$. Based on this feedback, $\Delta_W(t)$, we can design a distance controller in order to control the size of the network to obtain a better performance of the confidence as shown in Algorithm 7.

In Algorithm (7), K^c is designed so that $d^{new} > 0$. In order to do this we let

$$d - \frac{K^c}{M} \sum_{k=1}^{M} \Delta_W^k(t) > 0 \rightarrow K^c < \frac{d}{\frac{1}{M}\sum_{k=1}^{M} \Delta_W^k(t)}.$$

Therefore we can select $K^c = \frac{d}{\frac{1}{M}\sum_{k=1}^{M} \Delta_W^k(t) + c}$, here c is a positive constant. We can see that Algorithm 7 can generate the appropriate distance for the input of the flocking controller.

Algorithm 7: Design of the Internode Distance Adjuster

if $\Delta_W^k(t) > 0$ **then**

$$d^{new} = d - \frac{K^c}{M} \sum_{k=1}^{M} \Delta_W^k(t)$$

M is the number of covered cells at time t which have a confidence less than the desired one.

else

$$d^{new} = d$$

end

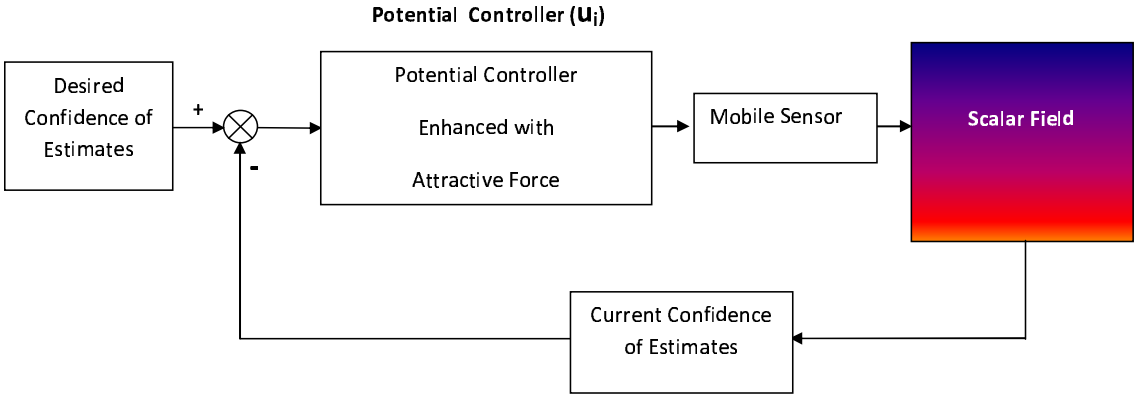


Figure 6.9: Diagram of cooperative and active sensing based on the potential controller enhanced with attractive force via confidence feedback.

6.5.3 Potential Controller

In this section we design another controller called potential controller to control the movement of mobile sensors in oder to increase the confidence level of the estimates. The structure of cooperative and active sensing scheme is shown in Figure 6.9. The main idea of this design is to create a virtual attractive force at the cells that have lower confidence than the lower bound (see Figure 6.8). In order to implement this idea we design the potential controller consisting of flocking controller with additional attractive force, so that it can drive a mobile sensor to move closer to the cells that have low confidence.

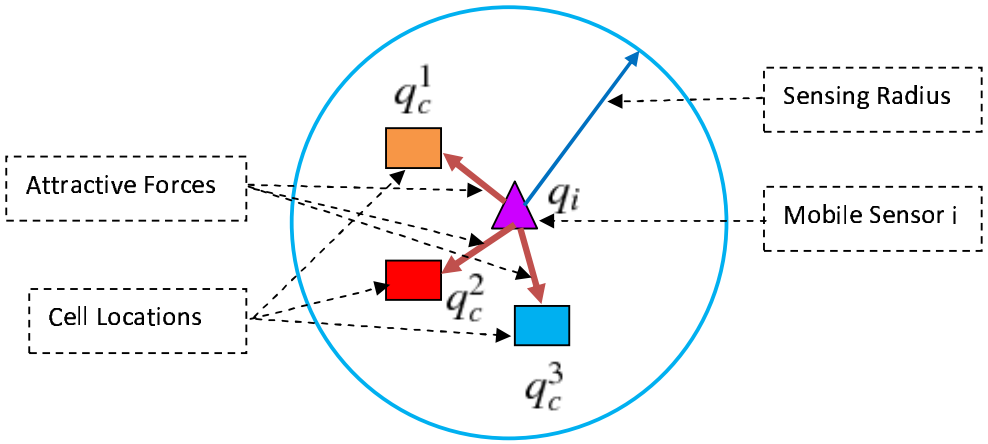


Figure 6.10: Illustration of creating virtual attractive forces in the cells which have the confidence level lower than the lower bound.

Design of Attractive Force

In this subsection, we introduce the attractive force term to the Potential Controller to achieve similar goal as the Distance Controller. The attractive force will steer the mobile sensors to the cells which have low confidence. In order to do this, first let q_c^k be the location of the cell that has confidence lower than the lower bound, or $k \in O_i^L(t)$, here $O_i^L(t)$ is the subset of cells covered by mobile sensor i at time t, which have confidence lower than the lower bound. $O_i^L(t) \subset O_i^c(t)$, here $O_i^c(t)$ is the set of cells covered by mobile sensor i at time t, and it is defined as

$$O_i^c(t) = \left\{ k \in \vartheta_O : \|q_c^k - q_i\| \leq r_i^s, \vartheta_O = \{1,2,...,k\} \right\}. \tag{6.41}$$

For these cells we will create the virtual attractive force to attract the mobile sensor to move closer to them in oder to get higher confidence of the estimates at these cells. This idea is illustrated in Figure 6.10.

At each time t, the mobile sensor i may have several cells which have confidence lower than the desired one. In order to steer the mobile sensor to go to these low confidence cells, the virtual attractive force are generated at these cells. If the cell has lower confidence the bigger attractive force is generated. To express the details of the attractive force design,

first let $\overline{W}_d^L$ be a lower bound of the desired confidence of the estimates of all cells in the scalar field, and $\overline{W}_d^L$ is a vector of $1 \times C$ dimension. Let $\Delta_W^L(t) = \overline{W}_d^L - \overline{W}(t)$ be the difference between the current confidence and the lower bound (see Figure 6.11), $\Delta_W^L(t) = [\Delta_W^1(t), \Delta_W^2(t), ..., \Delta_W^C(t)]$. Based on this feedback, $\Delta_W^L(t)$, we can design an attractive force as shown in Algorithm 8.

Algorithm 8: Design of Attractive Force

if $O_i^L(t) \neq \emptyset$ *or* $\Delta_W^k(t) > 0$ **then**

$$f_i^{att} = -\sum_{k \in O_i^L(t)} C_k^{att} \phi_{att}(\|q_c^k - q_i\|_\sigma) n_{i,k}^{att}$$

$C_k^{att} = c_a \frac{\Delta_W^k(t)}{\sqrt{1+(\Delta_W^k(t))^2}}, \Delta_W^k(t) \in \Delta_W^L(t)$, here c_a is a positive constant.

else

$$f_i^{att} = 0$$

end

In Algorithm 8, $C_k^{att} = c_a \frac{\Delta_W^k(t)}{\sqrt{1+(\Delta_W^k(t))^2}}$ controls the amplitude of the attractive force. Namely, if cell k has low confidence or $\Delta_W^k(t)$ is large, the the amplitude of the attractive force is big in order to attract the mobile sensor to go to closer this cell.

The attractive force function $\phi_{att}(\|q_c^k - q_i\|_\sigma)$ is designed as:

$$\phi_{att}(\|q_c^k - q_i\|_\sigma) = \rho_h(\frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s}) \frac{\|q_c^k - q_i\|_\sigma}{\sqrt{1+\|q_c^k - q_i\|_\sigma^2}}, k \in O_i^L(t).$$

here, $r_\alpha^s = \|r^s\|_\sigma$ (r^s is sensing range as defined before), and the bump function $\rho_h(\frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s})$ with $h \in (0,1)$ is defined as [23]

$$\rho_h(\frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s}) = \begin{cases} 1, & \frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s} \in [0,h) \\ 0.5[1+cos(\pi(\frac{\frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s} - h}{1-h}))], & \frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s} \in [h,1] \\ 0, & otherwise. \end{cases} \quad (6.42)$$

The vector along the line connecting q_c^k $(k \in O_i^L(t))$ and q_i is defined as:

$$n_{ik}^{att} = (q_c^k - q_i)/\sqrt{1+\varepsilon\|q_c^k - q_i\|^2}, k \in O_i^L(t). \tag{6.43}$$

here, ε is small positive constant.

The formation controller is used to control the network to form a quasi lattice formation, and it is designed based on a pairwise attractive/repulsive force as discussed in previous chapter. This formation controller [23] is restated as follows

$$f_i^\alpha = c_1^\alpha \sum_{j\in N_i(t)} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j\in N_i(t)} a_{ij}(q)(p_j - p_i). \tag{6.44}$$

The leader tracking controller is used to control each mobile sensor to track the virtual leader which generates the path for path planning purpose as presented in the Path Planning section (Section 6.4). This controller is presented as

$$f_i^t = -c_1^t(q_i - q_t) - c_2^t(p_i - p_t) \tag{6.45}$$

here c_1^t and c_2^t are positive constant, and q_t and p_t are position and velocity of the virtual leader, respectively.

Finally, we propose the whole control algorithm for the cooperative and active sensing including the attractive force term only as follows:

$$\begin{aligned} u_i &= f_i^{att} + f_i^\alpha + f_i^t \\ &= \sum_{k\in O_i^L(t)} C_k^{att} \phi_{att}(\|q_c^k - q_i\|_\sigma) n_{i,k}^{att} \\ &+ c_1^\alpha \sum_{j\in N_i(t)} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j\in N_i(t)} a_{ij}(q)(p_j - p_i) \\ &- c_1^t(q_i - q_t) - c_2^t(p_i - p_t) \end{aligned} \tag{6.46}$$

6.5.4 Quasi Uniformity of Confidence

Based on the attractive force design in the previous section, the confidence level can be increased, however some cells may have too high confidence. This is unnecessary since

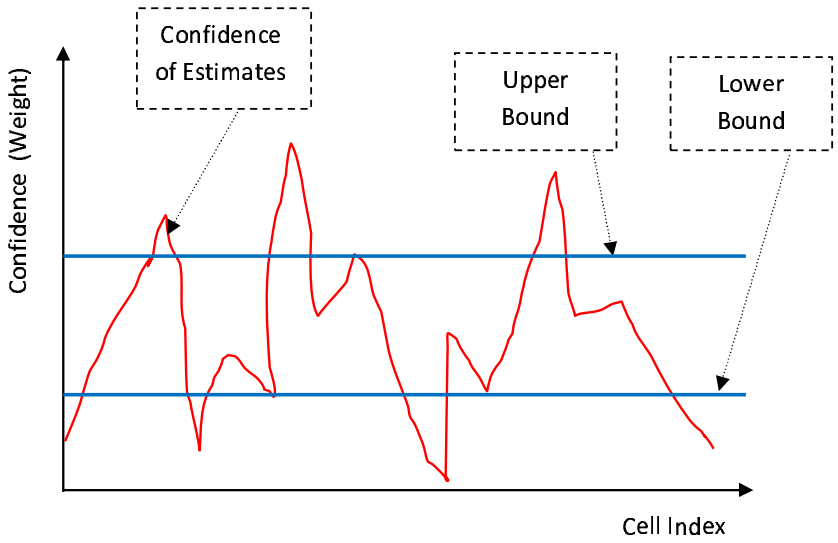


Figure 6.11: Illustration of confidence feedback for quasi uniformity of the confidence. The upper bound and lower bound are used to create a quasi uniform of the confidence.

this needs more measurements, and causes more energy consumption. Therefore, it is desirable if we can maintain a bound of the confidence performance, or we call a quasi uniform confidence (see Figure 6.11). Hence, we introduce a repulsive force term to the Potential Controller in order to steer the mobile sensors to move away from the cells which have too high confidence. The structure of cooperative and active sensing scheme is shown in Figure 6.12. The main idea of this design is to create a virtual attractive force at the cells that have lower confidence than the lower bound as shown in the previous section, and a repulsive force at the cells that have higher confidence than the upper bound (see Figure 6.11). In order to implement this idea we design the potential controller consisting of flocking controller with additional attractive and repulsive forces, so that it can drive a mobile sensor to move closer to the cells that have low confidence and move away from the cells that have high confidence.

Let q_c^k be the location of the cell that has confidence higher than the upper bound (see Figure 6.11). For these cells we will create the virtual repulsive force to steer the mobile sensors to move away. This idea is illustrated in Figure 6.13. The repulsive force is created based on the f_i^{rep} controller as shown in Algorithm 9.

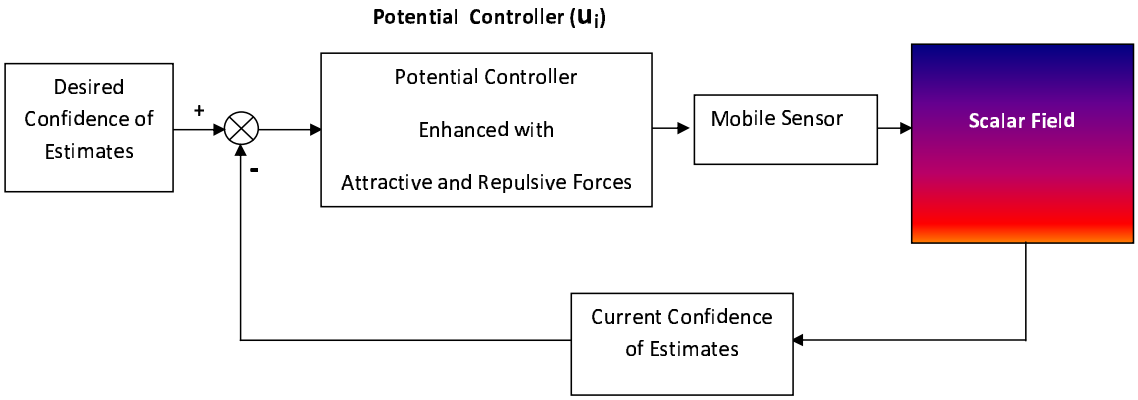


Figure 6.12: Diagram of cooperative and active sensing based on the potential controller enhanced with attractive and repulsive forces via confidence feedback.

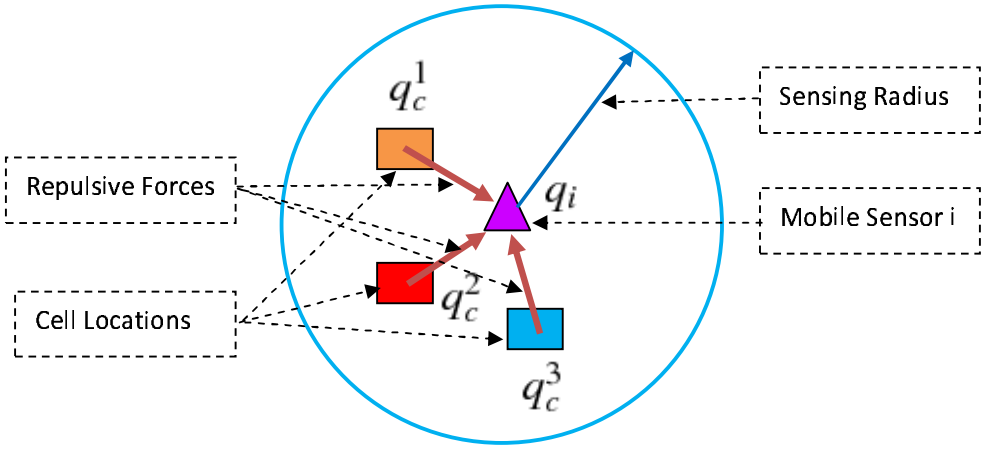


Figure 6.13: Illustration of creating virtual repulsive forces in the cells which have the confidence level higher than the upper bound.

To express the details of the repulsive force design, first let $\overline{W}_d^H$ be a upper bound of the desired confidence of the estimates of all cells in the scalar field, and $\overline{W}_d^H$ is a vector of $1 \times C$ dimension. Let $\Delta_W^H(t) = \overline{W}_d^H - \overline{W}(t)$ be the difference between the current confidence and the upper bound (see Figure 6.11), $\Delta_W^H(t) = [\Delta_W^1(t), \Delta_W^2(t), ..., \Delta_W^C(t)]$. Based on this feedback, $\Delta_W^H(t)$, we can design a repulsive force as shown in Algorithm 9.

Algorithm 9: Design of Repulsive Force

if $O_i^H(t) \neq \emptyset$ *or* $\Delta_W^k(t) < 0$ **then**

$$f_i^{rep} = \sum_{k \in O_i^H(t)} C_k^{rep} \phi_{rep}(\|q_c^k - q_i\|_\sigma) n_{i,k}^{rep}$$

$C_k^{rep} = c_r \frac{|\Delta_W^k(t)|}{\sqrt{1+(\Delta_W^k(t))^2}}, \Delta_W^k(t) \in \Delta_W^H(t)$, here c_r is a positive constance.

else

$$f_i^{rep} = 0$$

end

In Algorithm 9, $O_i^H(t)$ is the subset of cells covered by mobile sensor i at time t, which have confidence higher than the upper bound. Obviously, $O_i^H(t) \subset O_i^c(t)$. C_k^{rep} is used to control the amplitude of the repulsive force. Namely, if cell k has high confidence, or $\Delta_W^k(t)$ is large, the the amplitude of the repulsive force is big in order to push the mobile sensor to move away from this cell further.

The repulsive force function $\phi_{rep}(\|q_c^k - q_i\|_\sigma)$ is designed as [23]:

$$\phi_{rep}(\|q_c^k - q_i\|_\sigma) = \rho_h(\frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s})(\frac{\|q_c^k - q_i\|_\sigma - r_\alpha^s}{\sqrt{1+(\|q_c^k - q_i\|_\sigma - r_\alpha^s)^2}} - 1), k \in O_i^H(t).$$

The bump function $\rho_h(\frac{\|q_c^k - q_i\|_\sigma}{r_\alpha^s})$ is defined as (6.42), but it is now applied for the high confidence cells or $k \in O_i^H(t)$. The vector along the line connecting q_c^k ($k \in O_i^H(t)$) and q_i is defined as:

$$n_{ik}^{rep} = (q_c^k - q_i)/\sqrt{1+\varepsilon\|q_c^k - q_i\|^2}, k \in O_i^H(t). \tag{6.47}$$

Finally, we propose the whole control algorithm for the cooperative and active sensing including both attractive and repulsive force terms as follows:

$$
\begin{aligned}
u_i &= f_i^{rep} + f_i^{att} + f_i^{\alpha} + f_i^{t} \\
&= \sum_{k \in O_i^H(t)} C_k^{rep} \phi_{rep}(\|q_c^k - q_i\|_\sigma) n_{i,k}^{rep} \\
&+ \sum_{k \in O_i^L(t)} C_k^{att} \phi_{att}(\|q_c^k - q_i\|_\sigma) n_{i,k}^{att} \\
&+ c_1^{\alpha} \sum_{j \in N_i(t)} \phi_{\alpha}(\|q_j - q_i\|_\sigma) n_{ij} + c_2^{\alpha} \sum_{j \in N_i(t)} a_{ij}(q)(p_j - p_i) \\
&- c_1^t(q_i - q_t) - c_2^t(p_i - p_t)
\end{aligned} \tag{6.48}
$$

6.6 Simulation Results

In this section we test the Consensus Filters 1 and 2, then use our distributed sensor fusion algorithm to build the map of the scalar field.

6.6.1 Tests of Consensus Filter 1 and 2

In this subsection we test the Consensus Filter 1 and 2.

The Consensus Filter 1 is tested for the case of a single cell $k = 1$. We randomly generate a connected network of 10 nodes as shown in Figure 6.14(a). The cell is located at the center of the network (the read square in Figure 6.14(a)). The ground truth of the measurement at this location is 50. In this case the location of the measurement is inside the sensing range of all nodes, hence all nodes can make its own measurement to this location.

Each node makes a measurement as

$$m_i^1 = F^1 + n_i^1.$$

here $F^1 = 50$, and n_i^1 is the Gaussian noise, $N(0, V_i^1)$, with $V_i^1 = \frac{\|q_i - \overline{q}\|^2 + c_v}{(r_i^s)^2}$, $c_v = 0.01$, $r_1^s = r_2^s = ... = r_{10}^s = 1.6$, and $\overline{q} = \frac{1}{10}\sum_{i=1}^{10} q_i$. The initial condition for running the Consensus Filter 1 is $x_i^1(l = 0) = m_i^1$.

The results of the convergence of the Consensus Filter 1 associated with two different weights, *Weight Design 1* defined in Equation (6.15) and *Weight Design 2* defined in Equation (6.21), respectively, are presented in Figure 6.14. In Figure 6.14(a) to compare the speed of the convergence of $(x_i^1(l) - E^1)$ among nodes we generate a connected network with 10 nodes in which we let the node 4 have only 4 neighbors while other nodes have more than or equal to 7 neighbors. Observing Figure 6.14(b, d) we can see that $(x_i^1(l) - E^1)$, $i = 1, 2, ..., 10$, converge to zero after 300 iterations for *Weight Design 1* and 5 iterations for *Weight Design 2*. Therefore, it is better to use *Weight Design 2* since it can converge faster. We can also see that node 4 converges slower than the other nodes because it has less neighbors. Additionally, to clearly see the convergence, we show the result of

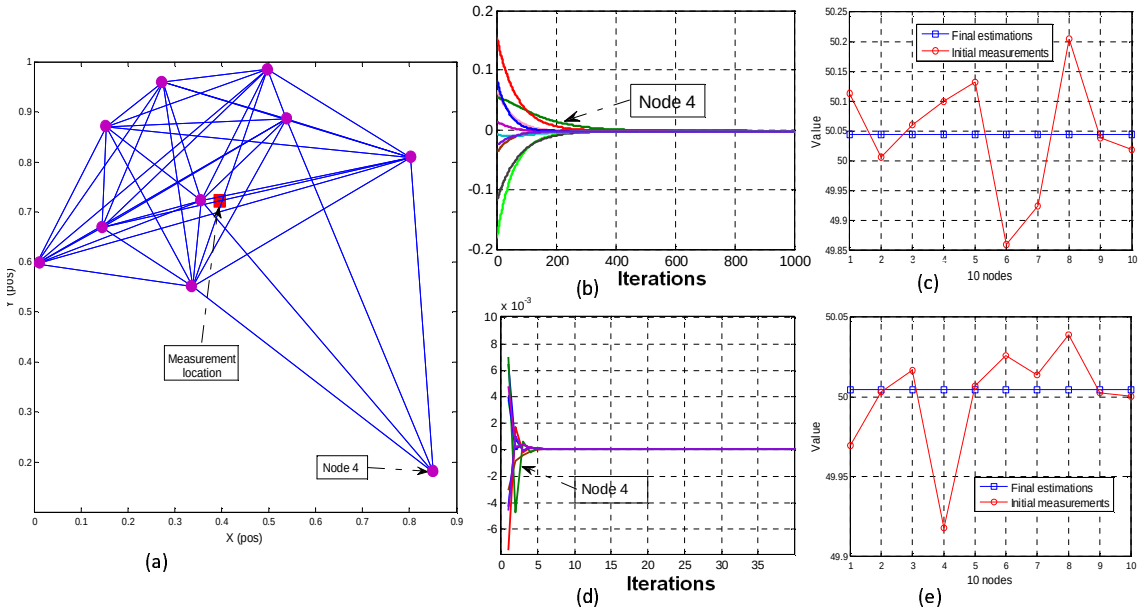


Figure 6.14: (a). 10 nodes estimate the value at cell k (pink square). (b, c) Result of convergence of 10 nodes, and agreement of 10 nodes when applying *Weight Design 1* in (6.15). (d, e) Result of convergence of 10 nodes, and agreement of 10 nodes when applying *Weight Design 2* in (6.21).

the agreement among nodes in Figure 6.14(c, e).

For testing the Consensus Filter 2, we let each sensor make its own measurement as

$$m_i^1 = F^1 + n_i^1.$$

here $F^1 = 50$, and n_i^1 is the Gaussian noise, $N(0,1)$. The initial condition for running the Consensus Filter 2 is $y_i^1(l=0) = m_i^1$.

The results of the convergence of the Consensus Filter 2 in (6.25) with the Metropolis weight, (6.26), are presented in Figure 6.15. Namely, Figure 6.15(b) shows the convergence of $(y_i^1(l) - \frac{1}{10}\sum_{i=1}^{10} y_i^1(0))$, and we can see that they converge to zero after 40 iterations. Figure 6.15(c) shows the agreement among 10 nodes, and we can see that all nodes in the network can agree on the same average value $(\frac{1}{10}\sum_{i=1}^{10} y_i^1(0))$. We also see that node 1 which has less neighbors than others converges slower.

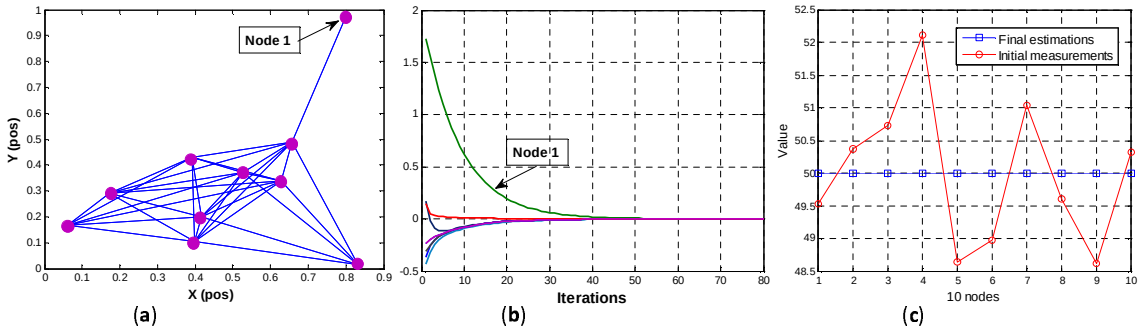


Figure 6.15: (a). Distribution of 10 nodes. (b, c) Result of convergence of 10 nodes, and agreement of 10 nodes when applying the Consensus Filter 2 in (6.25) with Metropolis weight (6.26).

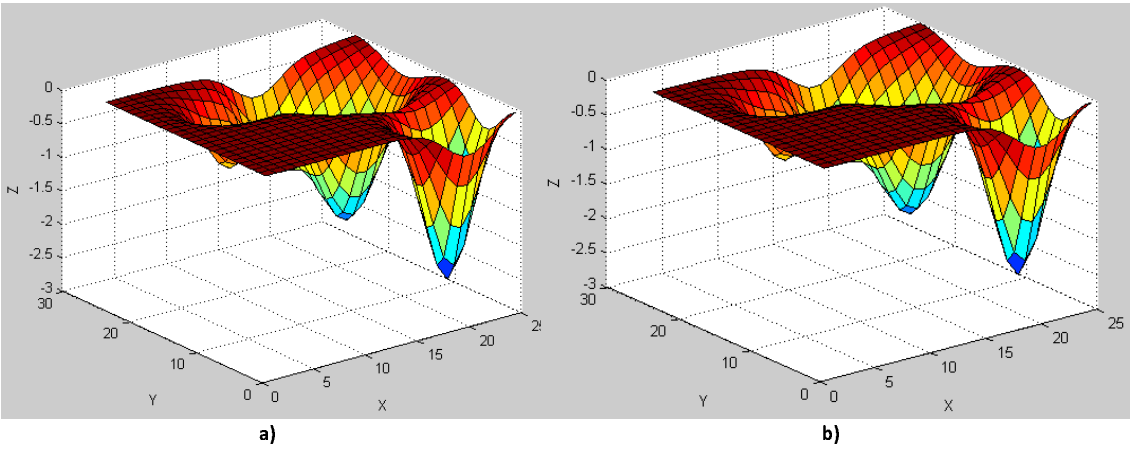


Figure 6.16: (a) the original map of the scalar field F, (b) the built map of the scalar field F using Algorithm 6.

6.6.2 Simulation Results of Cooperative Sensing

We model the environment (scalar field F) as multiple variate Gaussian distributions. We set $\Theta = [30\ 10\ 8\ 20]$, and use four multiple variate Gaussian distributions ($K = 4$), and each one is represented as:

$$\phi_1 = \frac{1}{\sqrt{det(C_1)(2\pi)^2}} e^{\frac{1}{2}(x-3)C_1^{-1}(y-2)^T},$$

here $C_1 = \begin{bmatrix} 2.25 & 0.2999 \\ 0.2999 & 2.25 \end{bmatrix}$, with $c_1^0 = 0.1333$.

$$\phi_2 = \frac{1}{\sqrt{det(C_2)(2\pi)^2}} e^{\frac{1}{2}(x-1)C_2^{-1}(y-4.5)^T},$$

here $C_2 = \begin{bmatrix} 1.25 & 0.1666 \\ 0.1666 & 1.25 \end{bmatrix}$, with $c_2^0 = c_1^0$.

$$\phi_3 = \frac{1}{\sqrt{det(C_3)(2\pi)^2}} e^{\frac{1}{2}(x+2)C_3^{-1}(y-3)^T},$$

here $C_3 = C_2$, and $c_3^0 = c_2^0$.

$$\phi_4 = \frac{1}{\sqrt{det(C_4)(2\pi)^2}} e^{\frac{1}{2}(x-4)C_4^{-1}(y+4)^T},$$

here $C_4 = C_3$, and $c_4^0 = c_3^0$.

The field F has a size $x \times y = 12 \times 12$, and it is partitioned into $25 \times 25 = 625$ cells. The result of the built map of the scalar field is shown in Figure 6.16. The snapshots of multiple sensor nodes forming a flock and building the map of the unknown scalar field are shown in Figure 6.17. The errors between the built and original maps in one and three dimensions are shown in Figure 6.18, 6.19, respectively. Three algorithms, Algorithm 1 with the weighted average update protocol, Algorithm 1 with the normal average update protocol, and the centralized fusion algorithm, are compared. We see that the map error in Algorithm 1 with the weighted average update protocol is similar to the one using the centralized fusion algorithm, but slightly smaller than the one using Algorithm 1 with the normal average update protocol. The confidence map which is built based on the summation of the weights at each cell of the field F is shown in Figure 6.20.

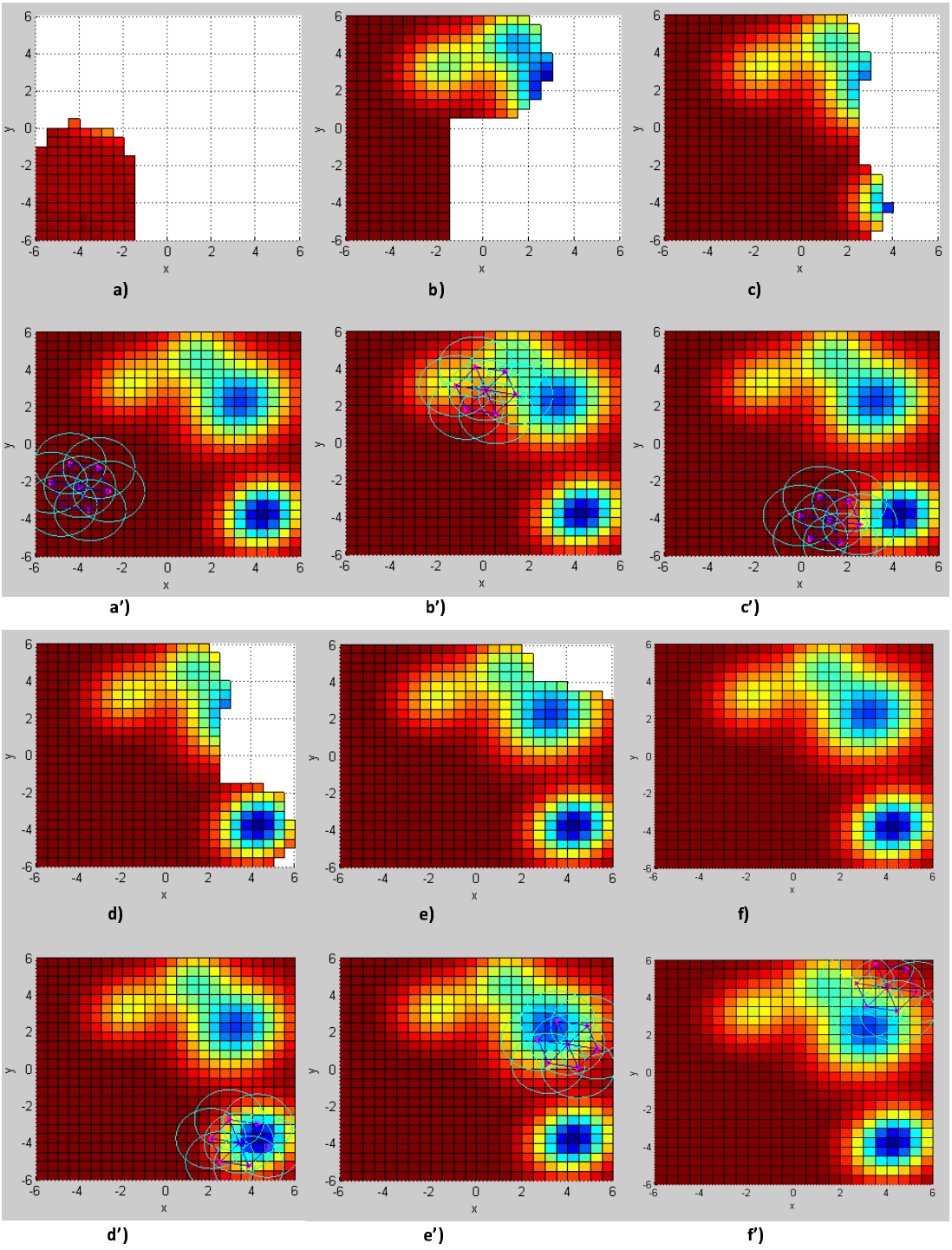


Figure 6.17: The snapshots of building the map of the scalar field F using Algorithm 6 and flocking control algorithm (6.40).

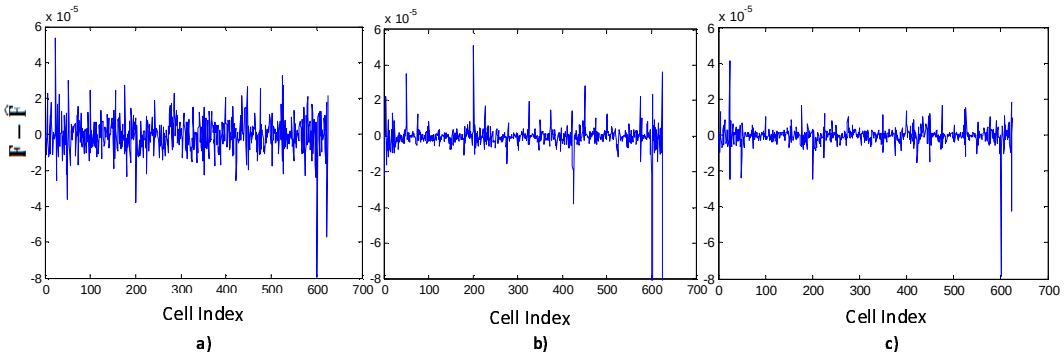


Figure 6.18: The error between the built and original maps for all cells in one dimension. (a) for Algorithm 1 with the normal average update protocol; (b) for centralized fusion algorithm; (c) for Algorithm 1 with the weighted average update protocol.

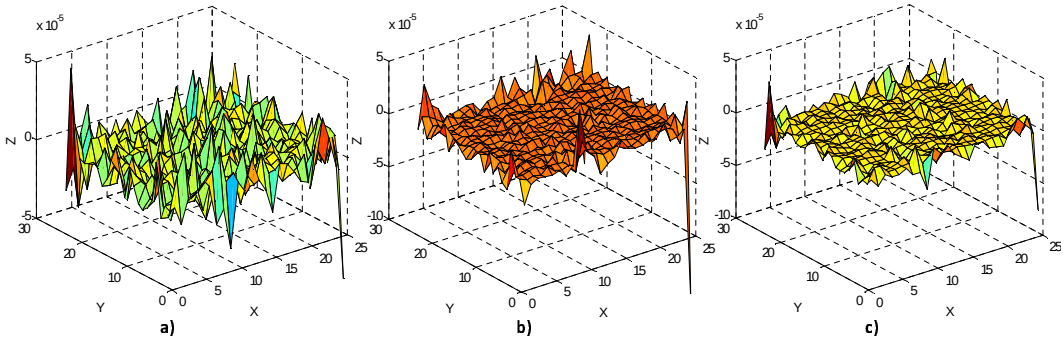


Figure 6.19: The error between the built and original maps for all cells in three dimensions. (a) for Algorithm 6 with the normal average update protocol; (b) for centralized fusion algorithm; (c) for Algorithm 6 with the weighted average update protocol.

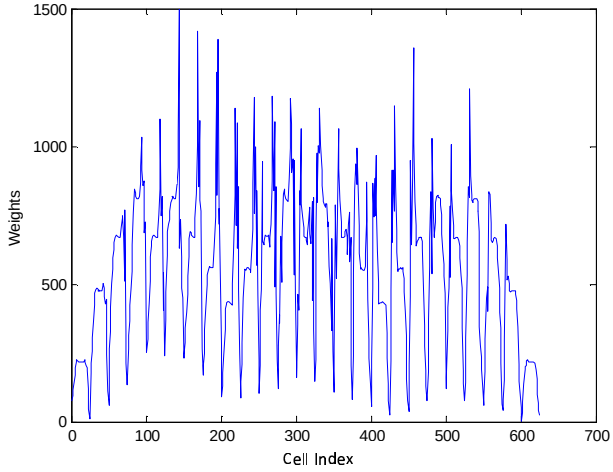


Figure 6.20: The confidence at each cell of the scalar field F.

6.6.3 Simulation Results of Cooperative Sensing and the Flocking Control with a Minority of Informed Agents

In this subsection we use the proposed distributed sensor fusion Algorithm 6, and the proposed flocking control algorithm with a minority of informed agents, Algorithm 3 in Chapter 3, to build the map of a scalar field.

In this flocking control algorithm, only a few sensor nodes closest to the virtual leader know its position and velocity. However, based on our algorithm, all mobile sensor nodes can flock together and form a network of lattice formation. Our flocking control algorithm allows the mobile sensor network to maintain the connectivity and reduce the tracking overshoot.

To evaluate the tracking performance the center of mass (CoM) of informed agents (sensors) is defined as

$$\begin{cases} \overline{q}^{inf} = \frac{1}{n}\sum_{i=1}^{n} q_i^{inf} \\ \overline{p}^{inf} = \frac{1}{n}\sum_{i=1}^{n} p_i^{inf} \end{cases} \tag{6.49}$$

To model the environment (scalar field F) four multiple variate Gaussian distributions ($K = 4$) with $\Theta = [20\ 50\ 35\ 40]$, and each one is represented as:

$$\phi_1 = \frac{1}{\sqrt{det(C_1)(2\pi)^2}} e^{\frac{1}{2}(x-2)C_1^{-1}(y-2)^T},$$

here $C_1 = \begin{bmatrix} 2.25 & 0.2999 \\ 0.2999 & 2.25 \end{bmatrix}$, with the correlation factor $c_1^0 = 0.1333$.

$$\phi_2 = \frac{1}{\sqrt{det(C_2)(2\pi)^2}} e^{\frac{1}{2}(x-1)C_2^{-1}(y-.5)^T},$$

here $C_2 = \begin{bmatrix} 1.25 & 0.1666 \\ 0.1666 & 1.25 \end{bmatrix}$, and the correlation factor $c_2^0 = c_1^0$.

$$\phi_3 = \frac{1}{\sqrt{det(C_3)(2\pi)^2}} e^{\frac{1}{2}(x-4.3)C_3^{-1}(y-3.5)^T},$$

here $C_3 = C_2$, and the correlation factor $c_3^0 = c_2^0$.

$$\phi_4 = \frac{1}{\sqrt{det(C_4)(2\pi)^2}} e^{\frac{1}{2}(x-3)C_4^{-1}(y+3)^T},$$

here $C_4 = C_3$, and with the correlation factor $c_4^0 = c_3^0$.

The field F has a size of 11.1×10, and it is partitioned into into 483 cells. The result of the built map of the scalar field is shown in Figure 6.24. The snapshots of multiple sensor nodes forming a flock and building the map of the unknown scalar field are shown in Figure 6.22. In this figure, we can see that only two mobile sensors (blue squares) have information (position and velocity) of the virtual leader (q_t, p_t), but they can drag the whole network to track the virtual leader while maintaining the network connectivity. The errors between the built and original maps in one and three dimensions are shown in Figure 6.23 (a, b), respectively. The final confidence of the estimate at each cell of the field F is shown in Figure 6.23 (c). The confidence map represents the accuracy of the estimate of the field. The higher confidence, the better accuracy of the estimate. The cells near the border of the field have measurements compared with the ones inside the field. Therefore, these border cells have lower accuracy (see Figure 6.23 (c)) corresponding with more error (see Figure 6.23 (a, b)) than other cells.

Figure 6.25 shows the tracking error between the position of the virtual leader (q_t) and the average of the position of the two informed agents $(mean(q^{inf}))$. We can see that the tracking performance has a small off-set distance between the virtual leader and the informed agents. At the sharp turning points of the path of the virtual leader, the tracking performance has bigger error.

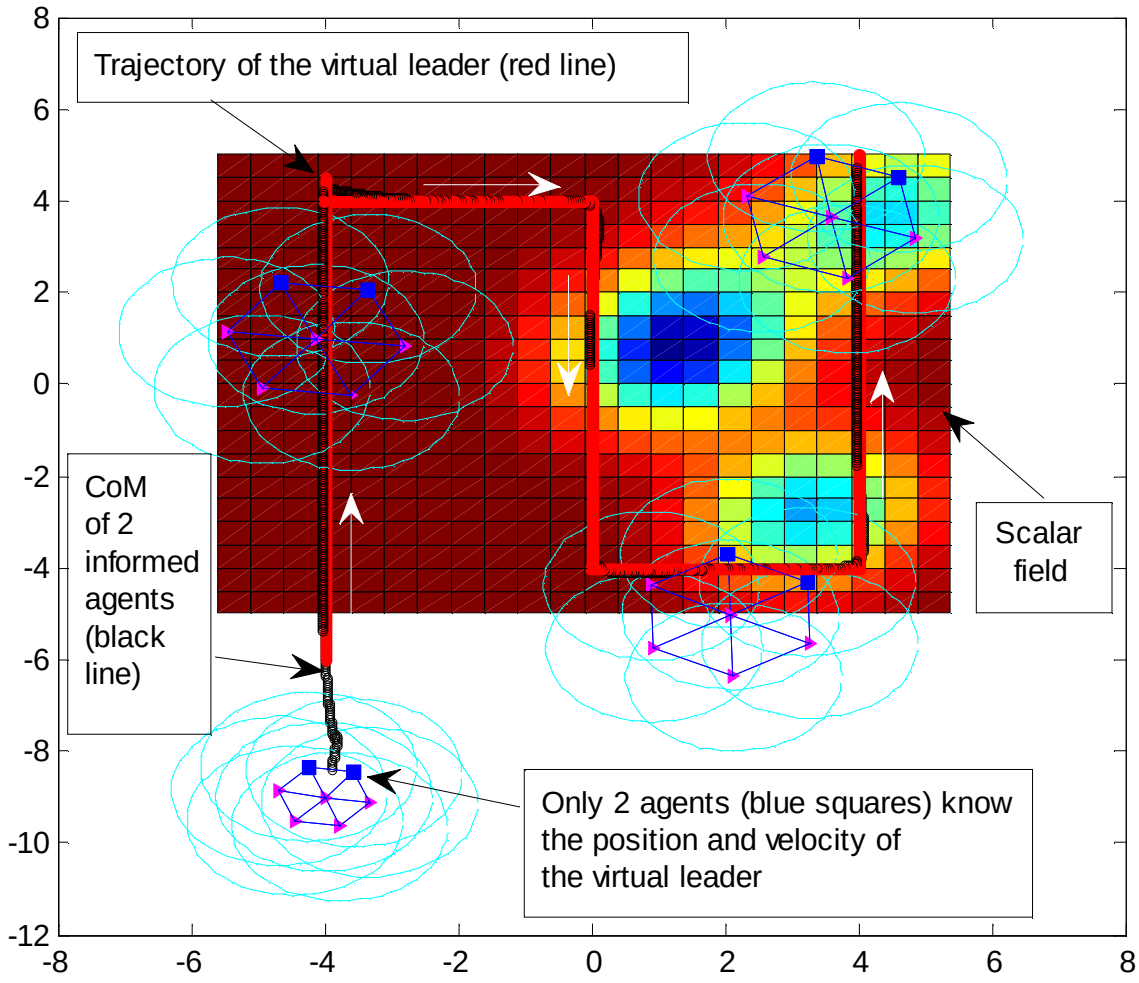


Figure 6.21: Seven mobile sensor nodes flock together and cover the scalar field (the filled square: 12×12). The motion path (red color) is generated by the leader, and the CoM (black/darker color) of the network tracks the leader with small overshoots at sharp turning points of the path.

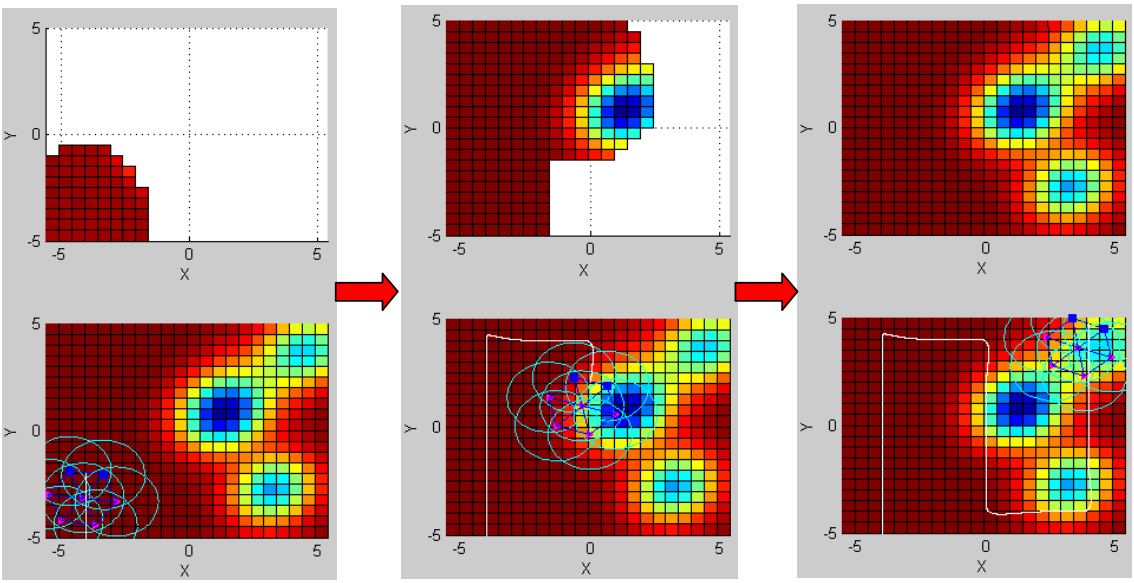


Figure 6.22: Snapshots of multiple mobile sensors flocking together and building the map of the scalar field. In these snapshots, only two mobile sensors (blue squares) have information of the virtual leader. The white line is the trajectory of the center of of position of two informed mobile sensors.

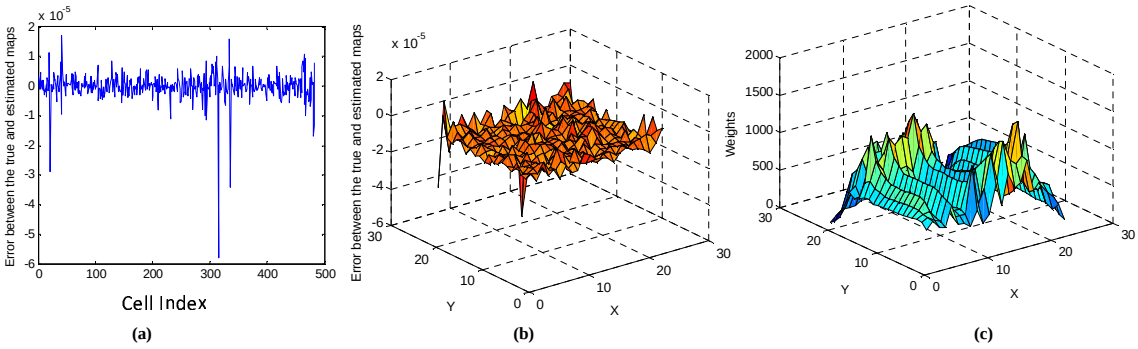


Figure 6.23: (a)- The error between the built and true maps for all cells in one dimension; (b)- The error between the built and true maps for all cells in three dimensions; (c)- The three dimensional confidence map.

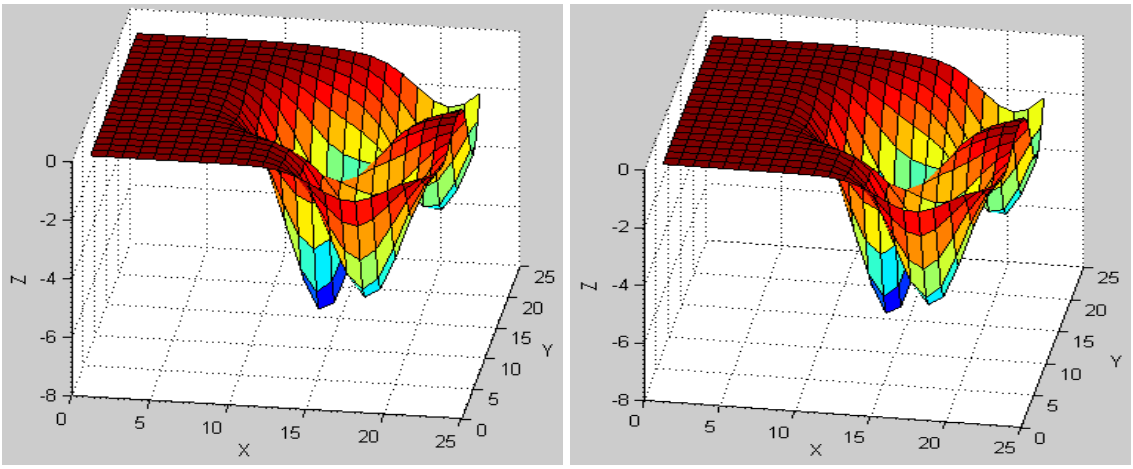


Figure 6.24: (a) The original map of the scalar field F; (b) The built map of the scalar field F using Algorithm 1.

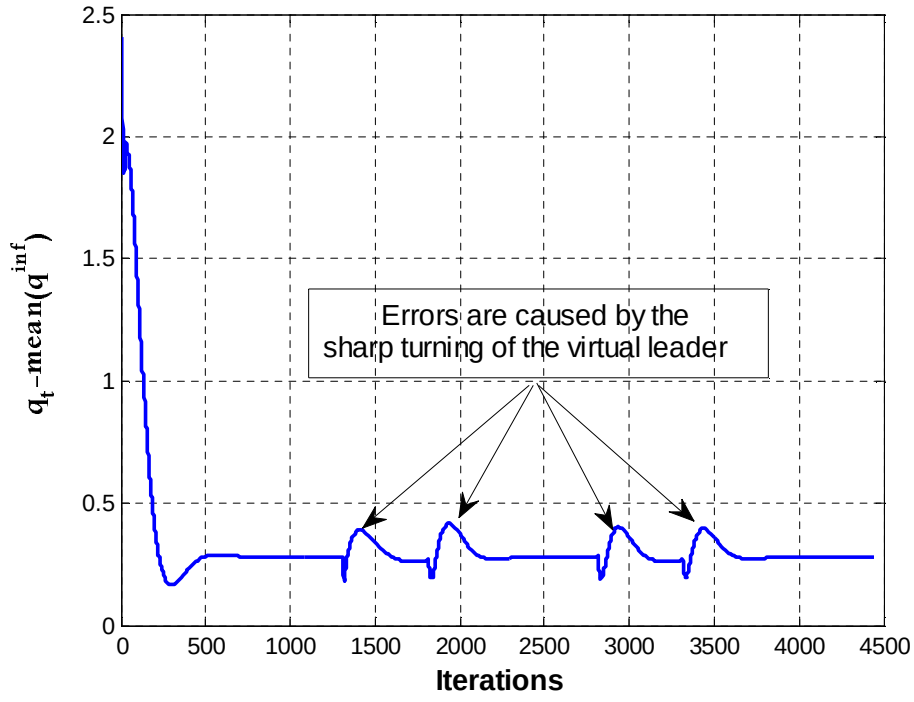


Figure 6.25: Tracking error between the position of the virtual leader (q_t) and the average of the position of the two informed agents ($mean(q^{inf})$).

6.6.4 Simulation Results of Active Sensing

In this subsection, we test our cooperative and active sensing algorithms and compare them with the normal cooperative sensing algorithm in terms of the sensing performance.

As before we model the environment (scalar field F) as multiple variate Gaussian distributions. We set $\Theta = [20\ 50\ 35\ 40]$, and use four multiple variate Gaussian distributions ($K = 4$), and each one is represented as:

$$\phi_1 = \frac{1}{\sqrt{det(C_1)(2\pi)^2}} e^{\frac{1}{2}(x-2)C_1^{-1}(y-2)^T},$$

here $C_1 = \begin{bmatrix} 2.25 & 0.2999 \\ 0.2999 & 2.25 \end{bmatrix}$, with the correlation factor $c_1^0 = 0.1333$.

$$\phi_2 = \frac{1}{\sqrt{det(C_2)(2\pi)^2}} e^{\frac{1}{2}(x-1)C_2^{-1}(y-.5)^T},$$

here $C_2 = \begin{bmatrix} 1.25 & 0.1666 \\ 0.1666 & 1.25 \end{bmatrix}$, and the correlation factor $c_2^0 = c_1^0$.

$$\phi_3 = \frac{1}{\sqrt{det(C_3)(2\pi)^2}} e^{\frac{1}{2}(x-4.3)C_3^{-1}(y-3.5)^T},$$

here $C_3 = C_2$, and the correlation factor $c_3^0 = c_2^0$.

$$\phi_4 = \frac{1}{\sqrt{det(C_4)(2\pi)^2}} e^{\frac{1}{2}(x-3)C_4^{-1}(y+3)^T},$$

here $C_4 = C_3$, and with the correlation factor $c_4^0 = c_3^0$.

We set the lower bound of the confidence level is 0.5×10^5, and the higher bound of the confidence level is 1.9×10^5.

The field F has a size of 10×9, and it is partitioned into 110 cells. The snapshots of multiple sensor nodes forming a flock and building the map of the unknown scalar field are shown in Figure 6.26. The errors between the built and original maps in one and three dimensions are shown in Figure 6.27 (a, b), respectively. Figure 6.27 (a) indicates the map

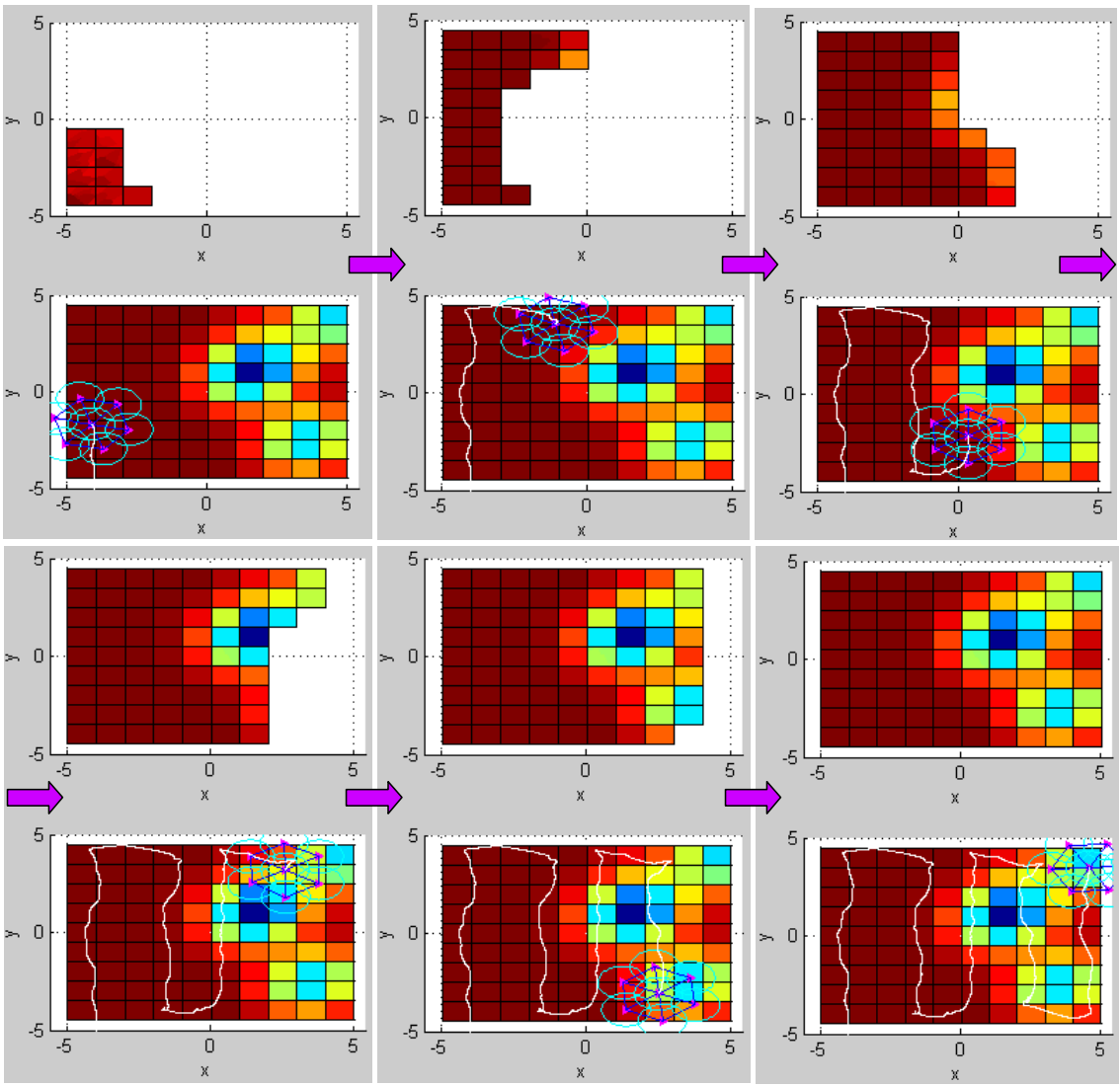


Figure 6.26: Snapshots of building the map of the scalar field F using Algorithm 6 and the cooperative and active sensing algorithm (6.48).

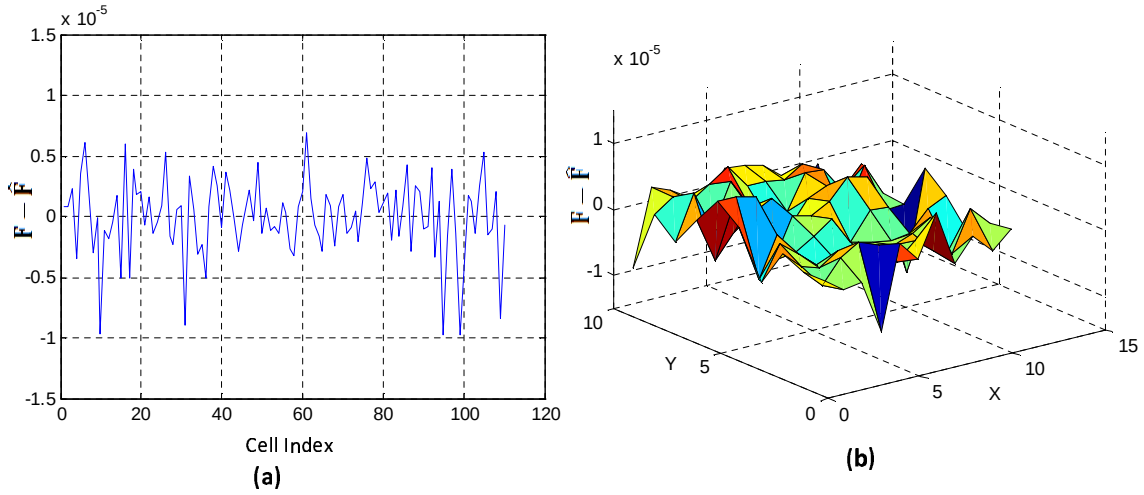


Figure 6.27: (a)- The error between the built and true maps for all cells in one dimension; (b)- The error between the built and true maps for all cells in three dimensions using Algorithm 6 and the cooperative and active sensing algorithm (6.48).

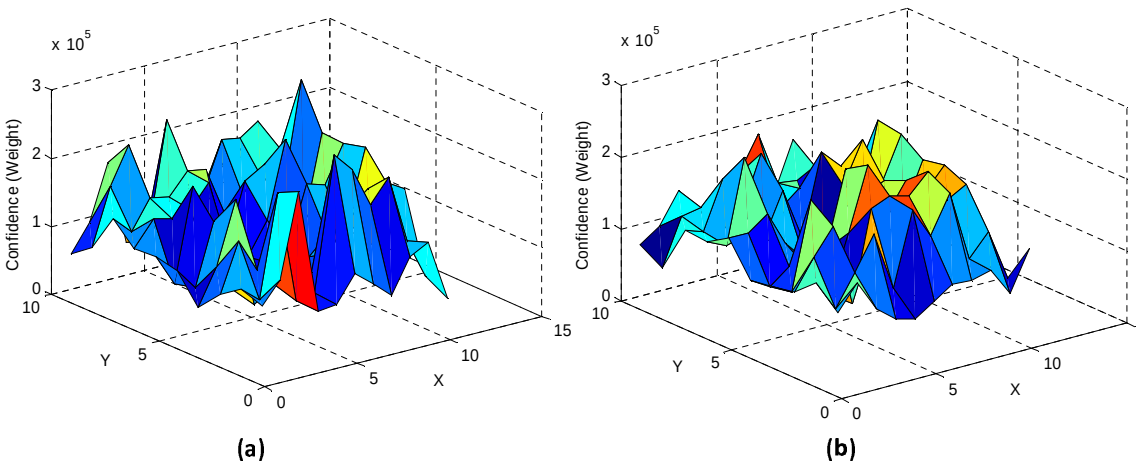


Figure 6.28: Confidence over the cells in 3 dimensions: (a) for active sensing with Potential Controller using attractive force only, Algorithm (6.46) ; (b) for active sensing with Potential Controller using both attractive and repulsive, Algorithm (6.48).

error corresponding to the index of the cell, while Figure 6.27 (b) indicates the map error corresponding to the location of the cell.

The confidence maps in three dimensions of the active sensing algorithm with the Potential Controller are shown in Figure 6.28. This three dimensional confidence map indicates the confidence of the estimate at each cell corresponding with its location in the scalar field. We can see that the Potential Controller using both attractive and repulsive forces (see Figure 6.28 (b)) performs better than that of using only the attractive force since the confidence level is increased, and the quasi uniformity of the confidence performance is achieved.

For more details, the final confidence of the estimate in one dimension at each cell of the field F is also shown in Figure 6.29. In this figure we compared four methods together. Namely, Figure 6.29 (a) shows the confidence of normal cooperative sensing, where Algorithm 6 and the flocking control algorithm (6.40) are used. Figure 6.29 (b) shows the confidence of active sensing with the Distance Controller, where Algorithm 6 and the flocking control algorithm with the distance controller in Algorithm 7 are used. Figure 6.29 (c) shows the confidence of active sensing with the Potential Controller, where Algorithm 6 and the cooperative and active sensing algorithm (6.46) are used. Figure 6.29 (d) shows the confidence of active sensing with Potential Controller, where Algorithm 6 and the cooperative and active sensing algorithm (6.48) are used. From these results, we can see that by using both attractive and repulsive force controllers we have better uniformity of the confidence performance. This indicates that all the cells of the scalar field are observed with a certain level of confidence.

To see how the mobile sensors adjust their movement in order to obtain better confidence performance, we show the distance between the mobile sensor 1 and one of its neighbors in Figure 6.30 (d). We see that this distance is changing over time, or the mobile sensor tries to move closer to the low confidence cells and and stay away from the high confidence cells. This creates the better uniformity of the confidence (see Figure 6.29

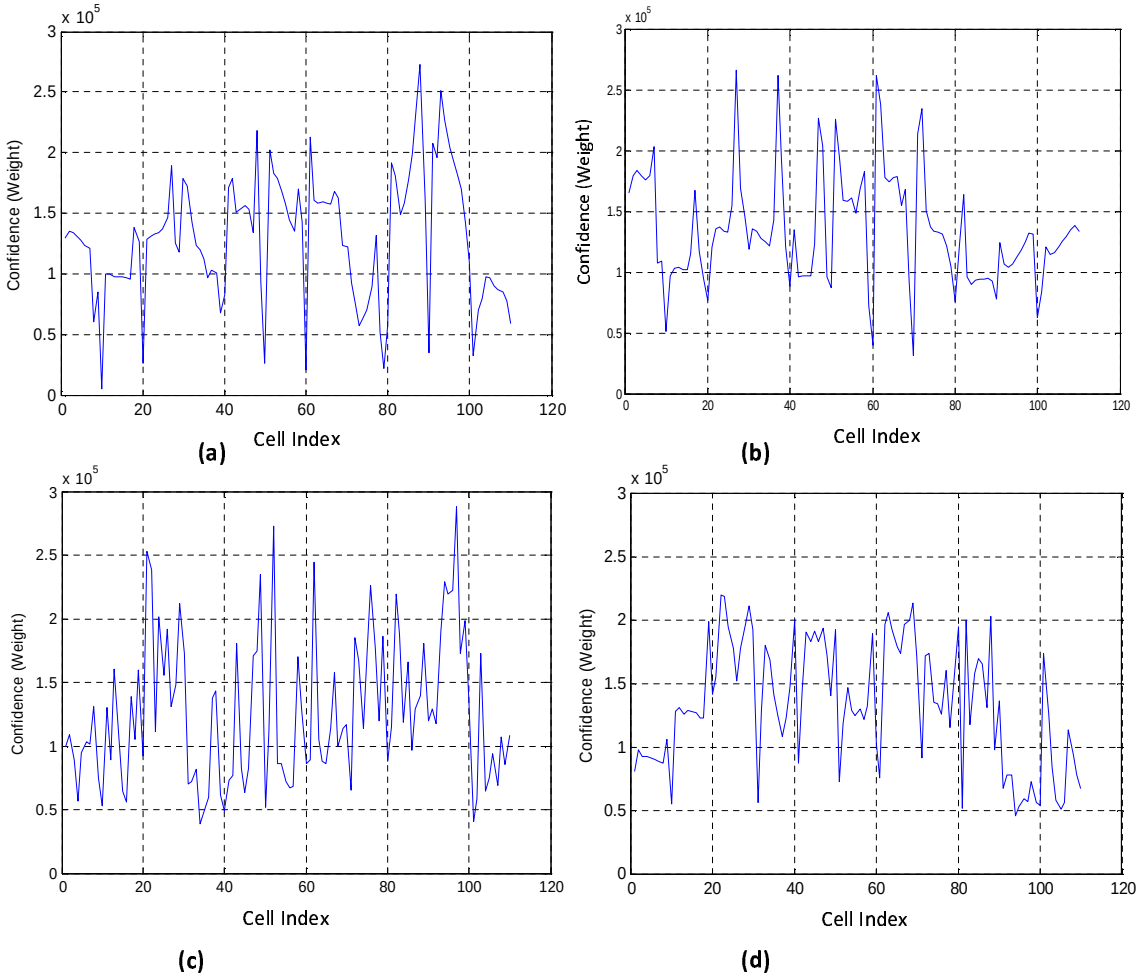


Figure 6.29: Confidence over the cells in one dimension: (a) for normal cooperative sensing; (b) for active sensing with Distance Controller; (c) for active sensing with Potential Controller using only attractive force (6.46); (d) for active sensing with Potential Controller using both attractive and repulsive forces (6.48).

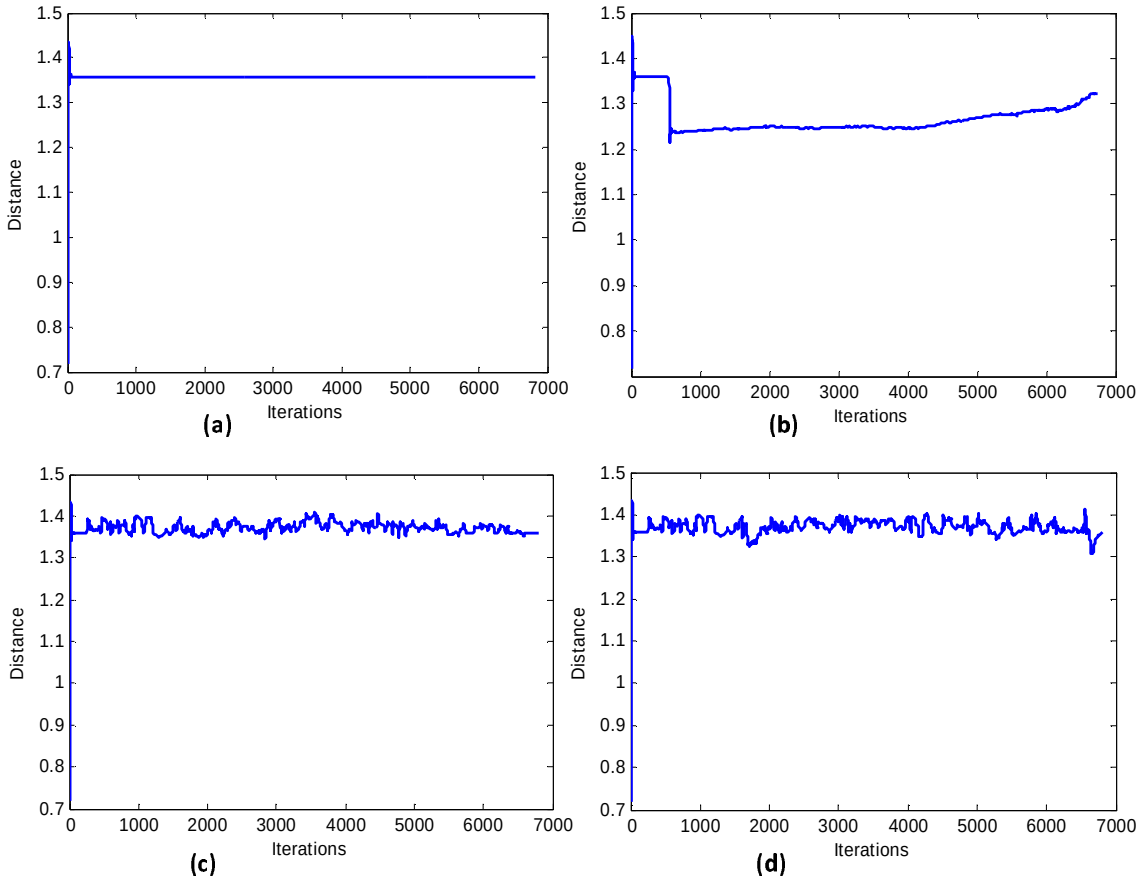


Figure 6.30: Distance between the mobile sensor 1 and its neighbor: (a) for normal cooperative sensing; (b) for active sensing with Distance Controller; (c) for active sensing with Potential Controller using only attractive force (6.46); (d) for active sensing with Potential Controller using both attractive and repulsive forces (6.48).

(d)). To see the advantages of the active sensing algorithm we compare it to three other algorithms, normal cooperative sensing, active sensing with the Distance Controller, and active sensing with the Potential Controller integrating only attractive force. Based on this comparison, we see that for the normal cooperative sensing the inter-nodes distance does not change (see Figure 6.30 (a)) therefore the confidence is not good (Figure 6.29 (a)), and some cells have very low confidence. For the active sensing with the Distance Controller and the Potential Controller using only attractive force (Figure 6.30 (b, c)), the results are better than that of the normal cooperative sensing algorithm, since the mobile sensors try to adjust their movement to achieve maximal confidence at each cell. However, the uniformity is not good as shown in (Figure 6.29 (b, c)).

To see the advantages of the active sensing we compare it with the normal sensing in term of mapping error. As shown in Figure 6.31 we see that the error between the original map and the built map in one dimension over cells is small (see Figure 6.31 (b)) when applying the active sensing and big when applying the normal sensing (see Figure 6.31 (a)).

In Figure 6.32, we can see that the higher confidence corresponds to the smaller error, and the lower confidence may lead to the bigger error. More specifically, at cells 10th, 92th and 100th the confidences are smallest (see Figure 6.32 (a)) therefore at these cells the errors between the original map and the built map are biggest (see Figure 6.32 (b)).

To see the effectiveness of the quasi uniformity of the confidence, we collect the total number of measurements at each cell as shown in Figure 6.33. We can see that for the cooperative and active sensing algorithm using the Potential Controller with attractive force only, some cells have very high number of measurements. This is unnecessary because it may cause more power consumption to estimate the value at these cells. For the cooperative and active sensing algorithm using the Potential Controller with both attractive and repulsive forces, we can reduce the number of measurements at these cells corresponding to the one using attractive force only.

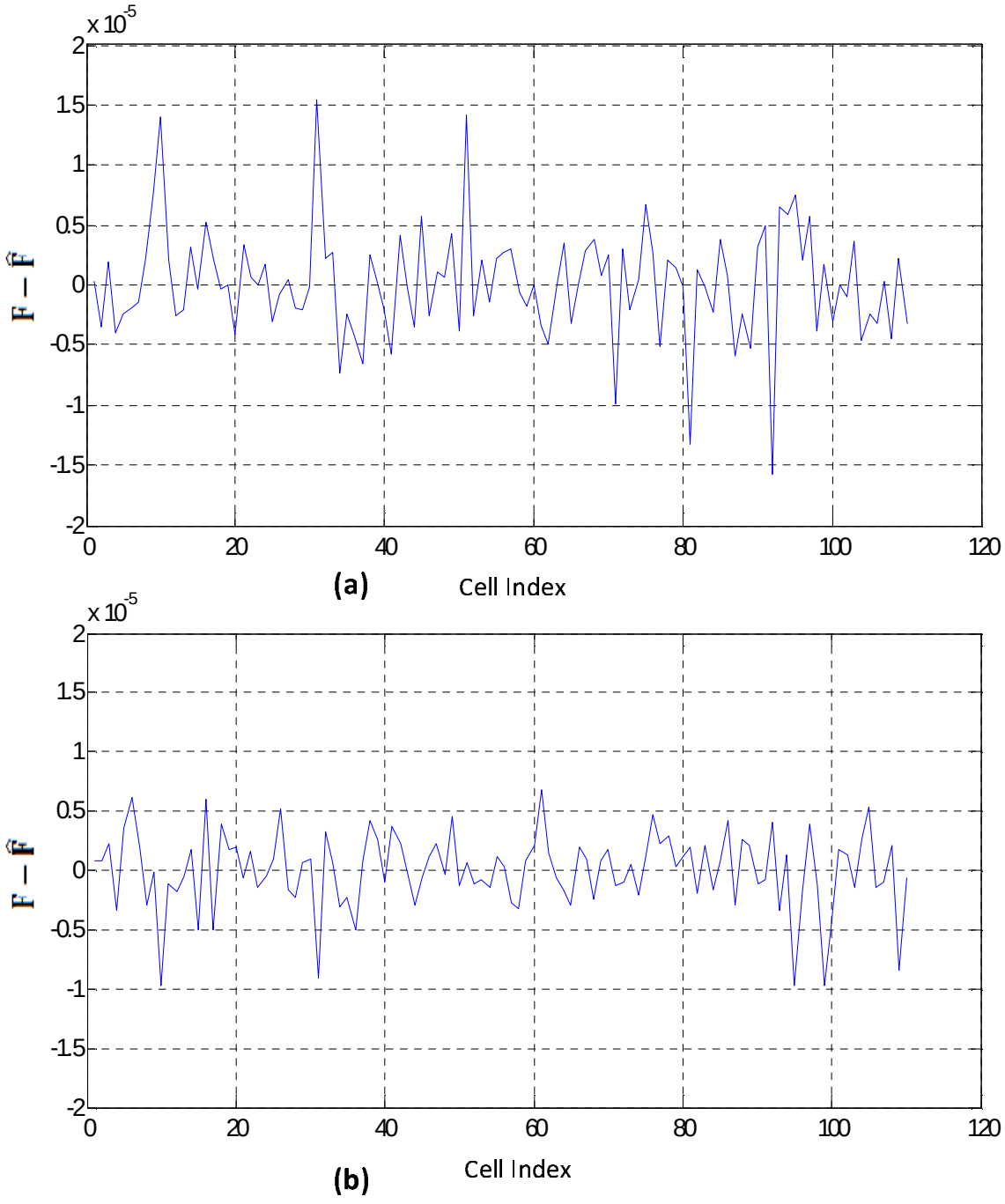


Figure 6.31: Error between the original map and the built map in one dimension over cells: (a) for the normal sensing; (b) for the active sensing.

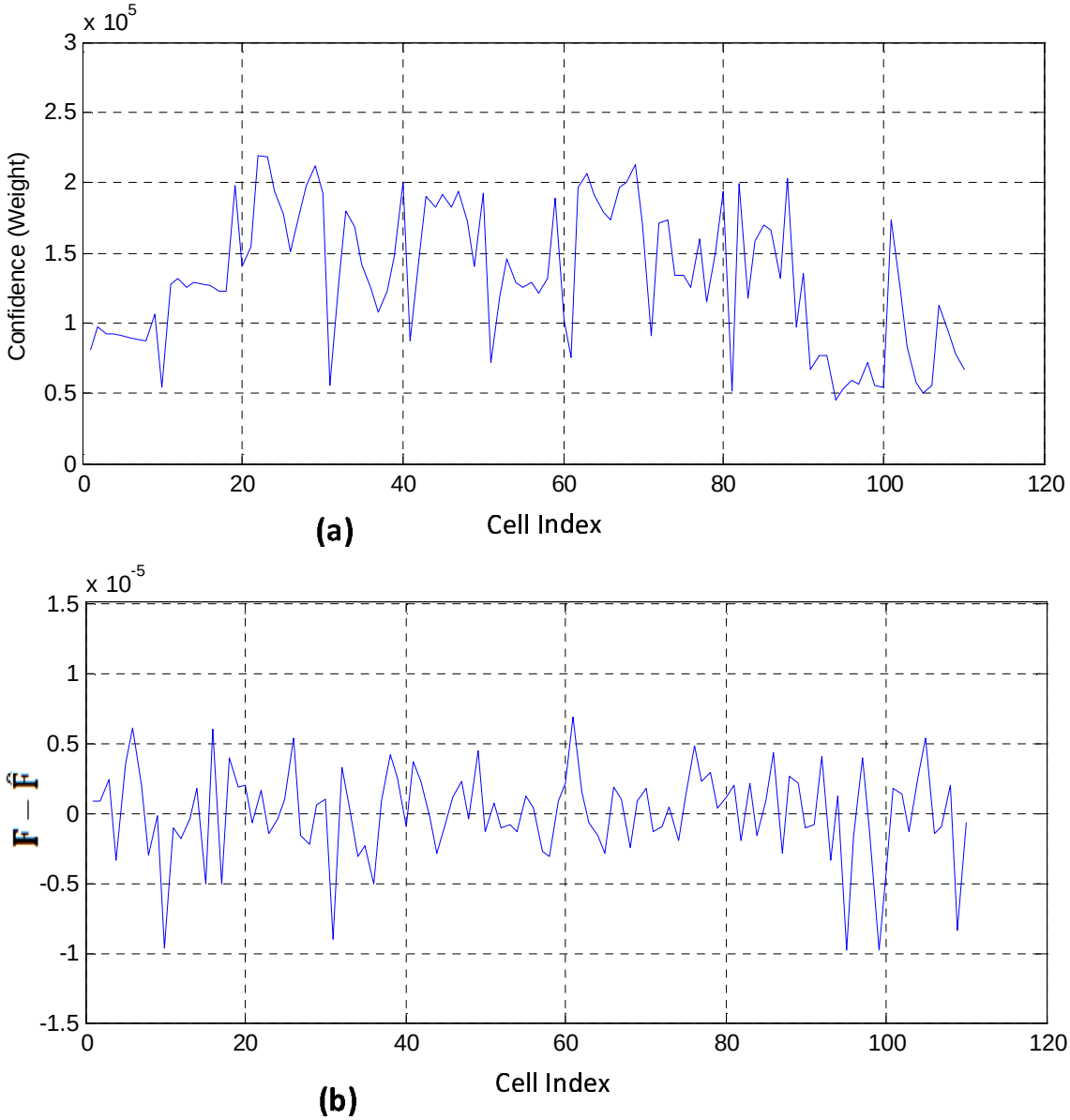


Figure 6.32: (a) Confidence over cells; (b) Error between the original map and the built map in one dimension over cells.

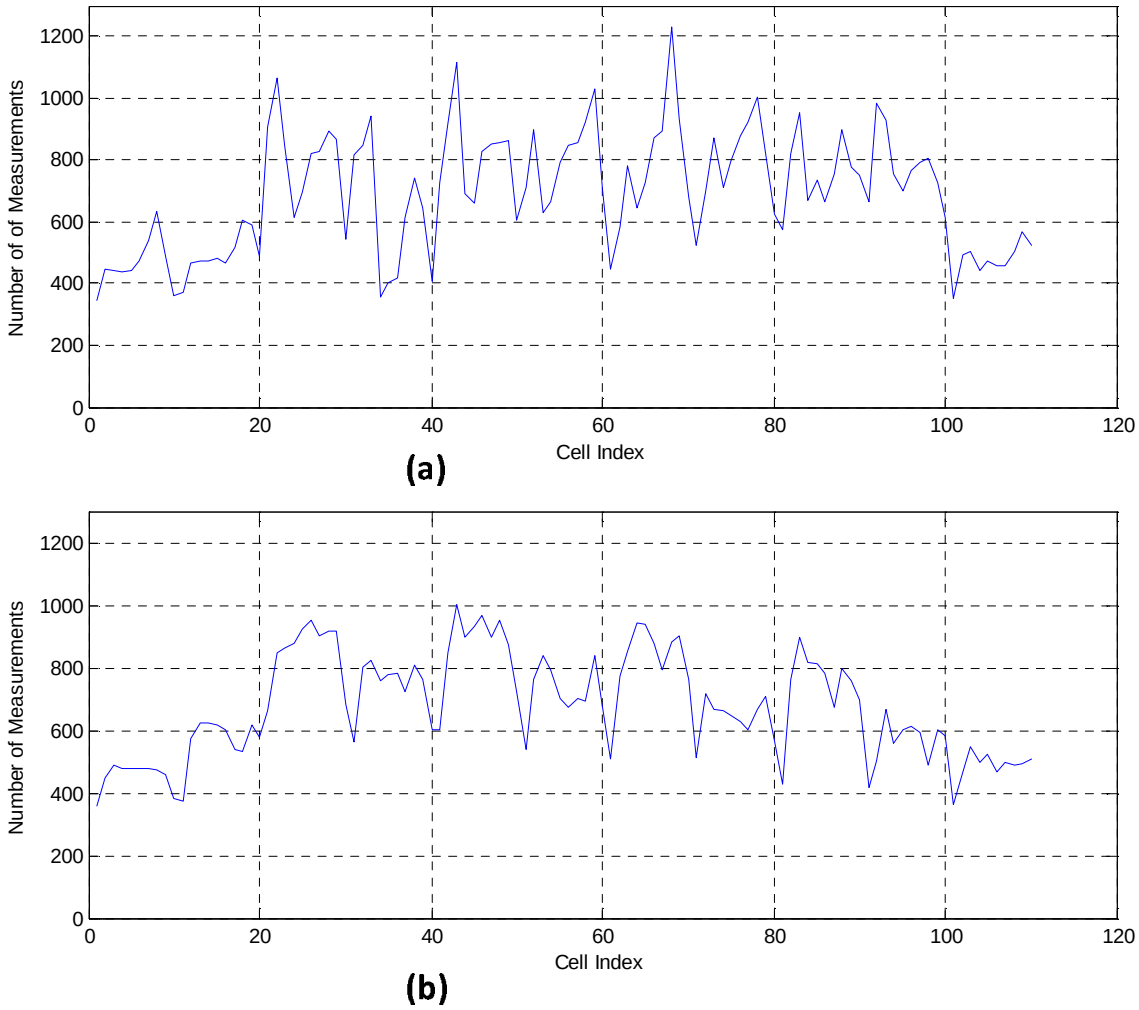


Figure 6.33: (a) For Potential Controller with attractive force only; (b) For Potential Controller with both attractive and repulsive force.

6.7 Summary

This chapter presented cooperative and active sensing algorithms for mobile sensor networks to build the map of an unknown scalar field. The proposed distributed sensor fusion algorithm consists of two different distributed consensus filters which can find an agreement on the estimates and an agreement on the confidences among sensor nodes. Each sensor node cooperates with neighboring sensors to estimate the value of the field at each cell. The final estimates of the values of the scalar field are updated on-line based on the weighted average protocol. For the active sensing, the mobile sensors can automatically adjust their movement to achieve quasi uniform confidence. Experimental results are collected to demonstrate the proposed algorithms.

In our measurement or observation model defined in Equation (6.3) we model the variance of noise based on the normalization of the distance between the location of the sensor and the measurement location (cell location). To avoid the variance $V_i^k(t)$ to be equal to zero when the distance $\|q_i(t) - q_c^k\|$ is equal to zero, we introduced a small constant c_v. However, there are other possibilities to model the uncertainty of observation which should depend on what kind of sensing device is used. Additionally, we can see that the confidence of the estimate as defined in Equation (6.37) is based on the accumulated weight, or $\overline{W}_i^k(t) \in [0\ \infty)$. Therefore, it could be desirable if other measure of confidence can be explored to ensure that the confidence is normalized between zero and one.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This dissertation develops cooperative control, learning and sensing algorithms in a distributed fashion for MSNs to realize coordinated motion control, intelligent learning and sensing situational awareness.

For single target tracking, the *Single-CoM* and *Multi-CoM* flocking control algorithms are proposed to make the CoM of the sensor network converge to the target. This enables the mobile sensors to track and observe the target more effectively while maintaining their formation in the obstacle space. The comparison among three flocking control algorithms (*No-CoM*, *Single-CoM* and *Multi-CoM*) shows that the tracking performance in the flocking control with *Single-CoM* and *Multi-CoM* is better than that in the flocking control with *No-CoM*.

To deal with the situation where only very few agents have the information of the target, the decentralized flocking control algorithm which utilizes a minority of informed agents is proposed to lead the whole network to track the target while maintaining the connectivity.

To deal with changing environments the adaptive flocking control algorithm is proposed in which each agent can cooperatively learn the network's parameters in a decentralized fashion to change the size of network in order to maintain connectivity, tracking performance and similar formation when passing through a narrow space among obstacles. To see the benefit of the adaptive flocking algorithm we compared it with the regular flocking control algorithm, and we found that the connectivity, formation and tracking performance in the adaptive flocking control algorithm are better than those in the regular flocking con-

trol algorithm.

For multiple dynamic target tracking, the SGGP algorithm is proposed to solve the problem of splitting/merging the sensor agents from the network. Also, to demonstrate the benefit of this algorithm we compare it with the RS algorithm, and the results show that the SGGP algorithm outperforms the RS algorithm.

In noisy environments, a flocking control algorithm is proposed to coordinate the activities of multiple agents through noisy measurements. Based on our algorithm, all agents can form a network and maintain connectivity. We show that even with noisy measurements the flocks can achieve cohesion and follow the moving target. The stability and scalability of our algorithm are also investigated.

To create adaptive and intelligent MSNs we propose a hybrid system that integrates reinforcement learning and flocking control. Two problems in multi-robot concurrent learning of cooperative behaviors are studied. The first problem is how to generate efficient combination of high level behaviors (discrete states and actions) and low level behaviors (continuous states and actions) for multi-robot cooperation; and the second one is how to conduct concurrent learning in a distributed fashion. As a result, the proposed hybrid system can allow MSNs to learn avoiding predators while maintaining network topology and connectivity. The stability and scalability of the proposed system are given.

Additionally, we propose a novel method for multiple mobile sensor nodes to build a map of a scalar field through noisy measurements. Our method consists of three parts. First, we develop a distributed sensor fusion algorithm integrating two different distributed consensus filters to achieve cooperative sensing among sensor nodes. Second, we use the distributed flocking control algorithm to drive the center of the mobile sensor formation to track the desired paths generated. Third, we build a path planning strategy to obtain a complete coverage of the field. We also extended our cooperative sensing to active sensing in which the mobile sensors have the ability to adjust their movements to adapt to the environments in order to improve the confidence of the estimates.

Our work lays the foundation for developing intelligent motion control and situational awareness for MSNs which can be used in many applications.

7.2 Future work

There are several potential directions that can extend the work in this dissertation.

First, we can extend our cooperative and active sensing through multi-agent learning. We have realized the cooperative and active sensing where each mobile sensor can automatically adjust its relative location through the confidence feedback. However, it is better if each mobile sensor can learn the full sensor network configuration so that better sensing performance can be achieved. More specifically, each mobile sensor has to learn (i) how to find the optimal configuration of MSNs, and (ii) how to make decisions for next actions in order to maximize information gain and obtain the uniform confidence of estimates. In our cooperative sensing algorithm in Chapter 6, we assume that the field of view (FoV) of each agent is 360 degree. However it may not be valid since many sensors have limited FoV. One example of the coverage for multiple mobile sensors with limited FoV is shown in Figure 7.1. Therefore we can extend our cooperative sensing to the scenario of limited FoV. Through the reinforcement learning, at each moment each sensor can select the right action in order to obtain maximum coverage and the certain confident level of the estimation.

Second, we can implement our cooperative and active sensing algorithms on real mobile sensor networks. We can use our new developed platforms of mobile robots as shown in Figure 7.2. These mobile robots are WiFi enabled, have Fit-PC2 with an CPU: Intel Atom Z530, 1.6Ghz, LAN: Gigabit Ethernet, WLAN: 802.11g, IR receiver. These robots are also equipped with a variety of sensors including laser range finder URG-Hokuyo [138] with 240 degree of scanning range, fish-eyes camera Q24 [139] with 360 degree of viewing range, and webcam. In addition, more sensors such as temperature sensor, sonar, ultrasonic sensor and magnetic sensor can be readily added to the robot. We has already successfully developed a software of fully autonomous robot localization and object tracking based on

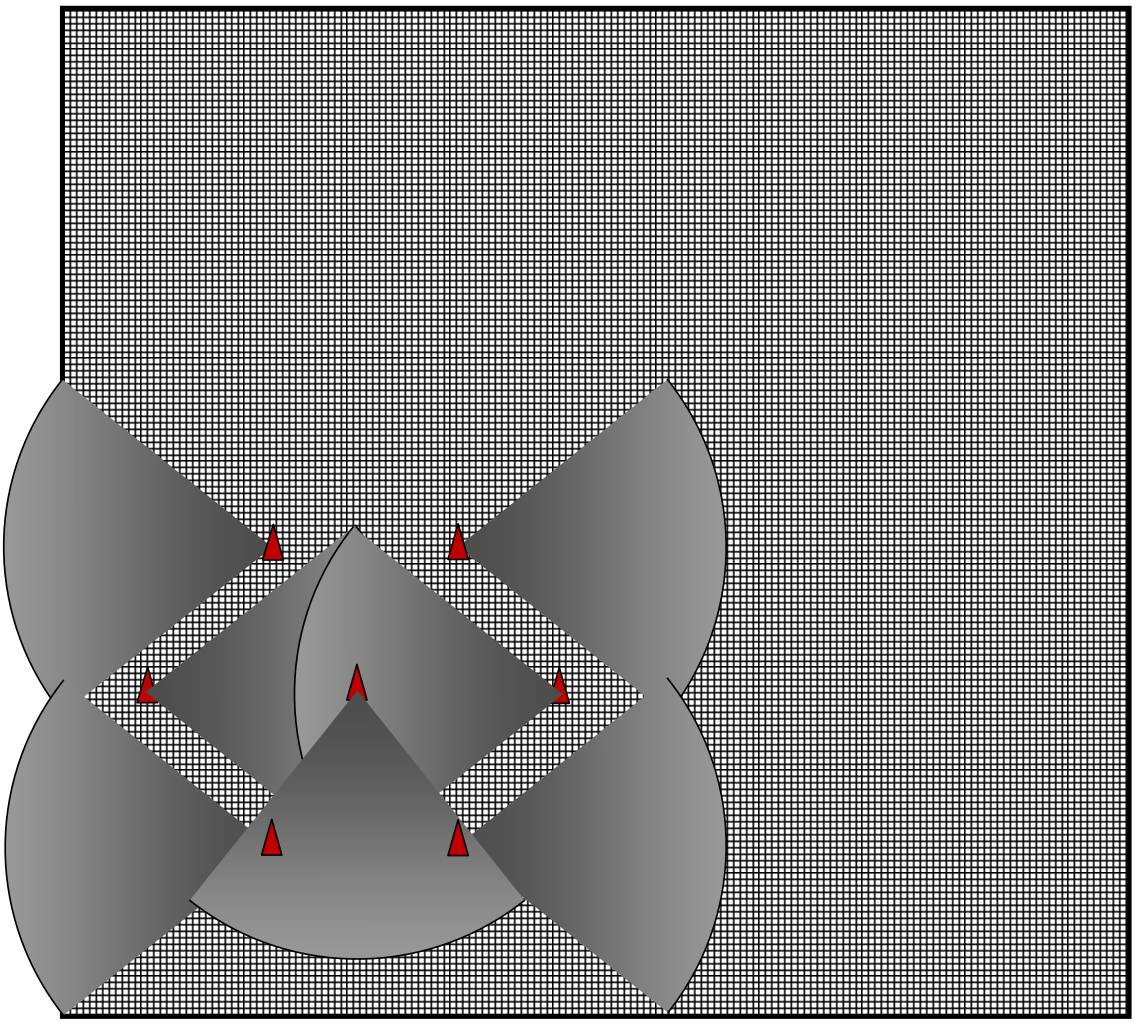


Figure 7.1: Illustration of coverage with limited sensing range.

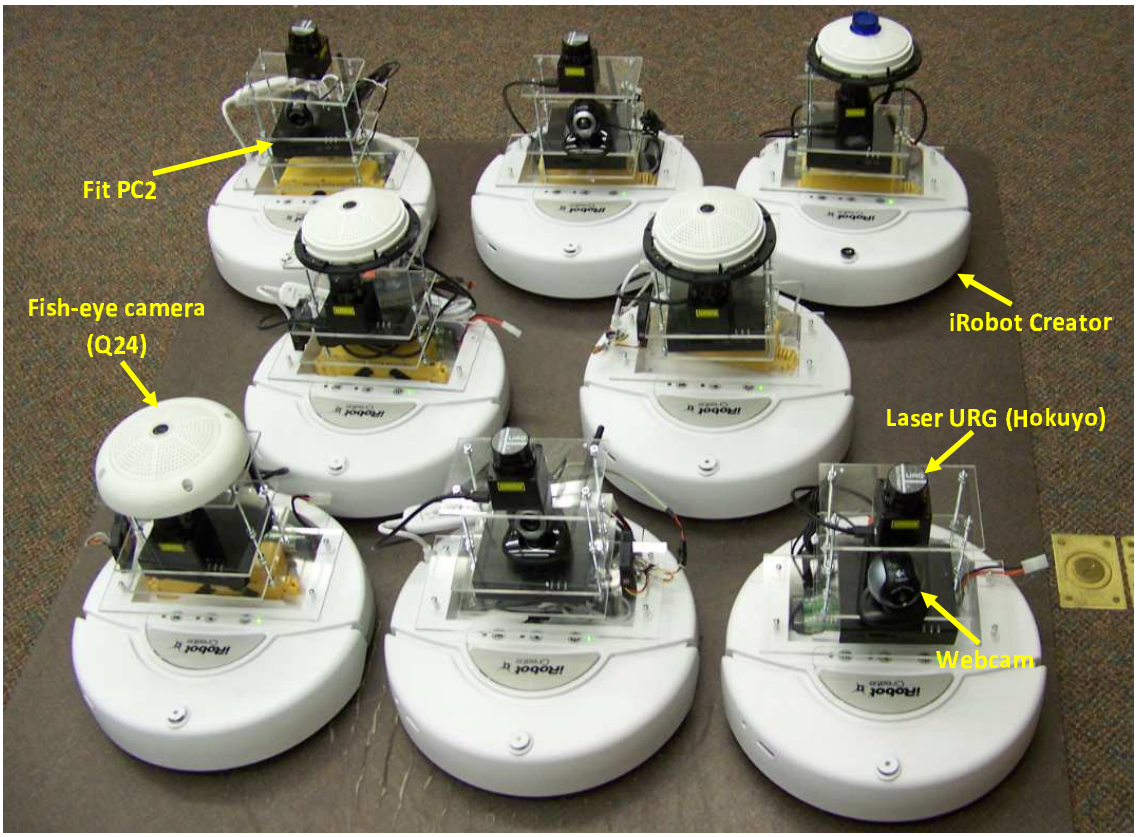


Figure 7.2: A mobile sensor network test bed.

the laser sensor and fish-eye Q24. This mobile sensing platform is an ideal experimental setup for test and evaluation of the cooperative and active sensing associated with distributed learning and coordination control algorithms for MSNs.

BIBLIOGRAPHY

[1] "VICON motion capture system: http://www.vicon.com/," 2011.

[2] "Harmful algal blooms observing system (HABSOS) by national oceanic and atmospheric administration (NOAA),"

[3] S. Kamath, E. Meisner, and V. Isler, "Triangulation based multi target tracking with mobile sensor networks," *the IEEE International Conference on Robotics and Automation,* ICRA 2007, pp. 3283–3288, 2007.

[4] I. Akyildiz, W. Su, Y. Sankarasubramanian, and E. Cayirci, "A survey of sensor networks," *IEEE Communications Magazine*, pp. 102–114, 2002.

[5] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[6] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 24–33, 2004.

[7] M. Tubaishat and S. Madria, "Sensor networks: An overview," *IEEE Potentials*, pp. 20–23, May 2003.

[8] A. Dhariwal, G. S. Sukhatme, and A. A. G. Requicha, "Bacterium inspired robots for environmental monitoring," *in Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1436–1443, 2004.

[9] A. Lilienthal, H. Ulmer, H. Frohlich, A. Stutzle, F. Werner, and A. Zell, "Gas source declaration with a mobile robot," *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pp. 1430–1435, 2004.

[10] D. V. Zarzhitsky, D. F. Spears, and D. R. Thayer, "Experimental studies of swarm robotic chemical plume tracing using computational fluid dynamics simulations," *International Journal of Intelligent Computing and Cybernetics*, pp. 1–44, 2010.

[11] I. I. Hussein, "A kalman filter-based control strategy for dynamic coverage control," *Proceedings of the American Control Conference*, pp. 3271–3276, 2007.

[12] J. Choi, S. Oh, and R. Horowitz, "Distributed learning and cooperative control for multi-agent systems," *Automatica*, vol. 45, no. 12, pp. 2802–2814, 2009.

[13] F. Zhang and N. E. Leonard, "Cooperative filters and control for cooperative exploration," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 650–663, 2010.

[14] J. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," *Proceedings of the 8th International Conference on Advanced Robotics, Monterrey, CA, USA*, 1997.

[15] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, "Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research," *Proceedings of the IEEE Conference on Systems, Man and Cybernetics, Toyko, Japan*, 1999.

[16] A. Ganguli, S. Susca, S. Martinez, F. Bullo, and J. Cortes, "On collective motion in sensor networks: sample problems and distributed algorithms," *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pp. 4239–4244, December 12-15 2005.

[17] R. Olfati-Saber, "Distributed tracking for mobile sensor networks with information driven mobility," *Proceedings of the 2007 American Control Conference,*, pp. 4606–4612, 2007.

[18] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA transactions*, vol. 46, pp. 3–13, 2007.

[19] E. Biagioni and K. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," *Special Issue on Distributed Sensor Networks for the International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 112–121, 2002.

[20] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[21] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry, "Distributed control applications within sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1235–1246, 2003.

[22] Z. Lin, M. E. Broucke, and B. A. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 622–629, 2004.

[23] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[24] R. M. Murray, "Recent research in cooperative control of multi–vehicle systems," *Journal of Dynamic Systems, Measurement, and Control*, pp. 571–583, 2007.

[25] C. Reynolds, "Flocks, birds, and schools: A distributed behavioral model," *Computer Graphics,ACM SIGGRAPH '87 Conference Proceedings, Anaheim, California*, vol. 21, no. 4, pp. 25–34, 1987.

[26] P. K. Stout, "Rhode island sea grant fact sheets," *http://seagrant.gso.uri.edu/factsheets/schooling.html*.

[27] T. Vicsek, A. Czirok, E. Jacob, I. Cohen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," *Phys. Rev. Lett*, vol. 75, pp. 1226–1229, 1995.

[28] H. Levine, W. J. Rappel, and I. Cohen, "Self-organization in systems of self-propelled particles," *Phys. Review. E*, vol. 63, pp. 017101–017104, 2000.

[29] A. Mogilner, L. Edelstein-Keshet, L. Bent, and A. Spiros, "Mutual interactions, potentials, and individual distance in a social aggregation," *J. Math. Biol*, vol. 47, pp. 353–389, 2003.

[30] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *J. Theor.Biol*, vol. 218, pp. 1–11, 2002.

[31] M. Anderson, E. McDaniel, and S. Chenney, "Constrained animation of flocks," *Eurographics/SIGGRAPH Symposium on Computer Animation*, pp. 1–12, 2003.

[32] D. Cruz, J. McClintock, B. Perteet, O. A. orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control–a multivehicle platform for research in networked embedded systems," *IEEE Control Systems Magazine*, pp. 58–78, 2007.

[33] V. Gervasi and G. Prencipe, "Coordination without communication: the case of the flocking problem," *Discrete Applied Mathematics*, pp. 324–344, 2004.

[34] H. Su, X. Wang, and Z. Lin, "Flocking of multi–agents with a virtual leader, part I: with a minority of informed agents," *Proceedings of the 46th IEEE Conference on Decision and Control*, pp. 2937–2942, 2007.

[35] H. Su, X. Wang, and Z. Lin, "Flocking of multi–agents with a virtual leader, part II: with a virtual leader of varying velocity," *Proceedings of the 46th IEEE Conference on Decision and Control*, pp. 1429–1434, 2007.

[36] H. Shi, L. Wang, T. Chu, F. Fu, and M. Xu, "Flocking of multi–agents with a virtual leader," *Proceedings of the 2007 IEEE Symposium on Artificial Life*, pp. 287–29, 2007.

[37] H. G. Tanner, A. Jadbabai, and G. J. Pappas, "Stable flocking of mobile agents, part I: fixed topology," *Proceedings of the 42nd IEEE Conference on Decision and Control*, pp. 2010–2015, 2003.

[38] H. G. Tanner, A. Jadbabai, and G. J. Pappas, "Stable flocking of mobile agents, part II: dynamic topology," *Proceedings of the 42nd IEEE Conference on Decision and Control*, pp. 2016–2021, 2003.

[39] A. Regmi, R. Sandoval, R. Byrne, H. Taner, and C. T. Abdallah, "Experimental implementation of flocking algorithms in wheeled mobile robots," *American Control Conference*, pp. 4917–4922, 2005.

[40] Z. Wang and D. Gu, "A survey on application of consensus protocol to flocking control," *Proceedings of the 12th Chinese Automation and Computing Society Conference in the UK, England*, pp. 1–8, 2006.

[41] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision making in animal groups on the move," *Letters to Nature*, vol. 433, no. 3, pp. 513–516, 2005.

[42] S. G. Reebs, "Can a minority of informed leaders determine the foraging movements of a fish shoal?," *Animal Behavior*, vol. 59, pp. 403–409, 2000.

[43] H. Milinski and R. Heller, "Influence of a predator on the optimal foraging behavior of sticklebacks," *Nature 275*, pp. 642–644, 1978.

[44] G. Roberts, "Why individual vigilance declines as group size increases," *Animal Behavior*, vol. 51, pp. 1077–1806, 1996.

[45] J. Krause, G. Ruxton, and D. Rubenstein, “Is there always an influence of shoal size on predator hunting success?,” *Journal of Fish Biology*, vol. 52, pp. 494–501, 1998.

[46] H. Su, X. Wang, and Z. Lin, “Flocking of multi–agents with a virtual leader,” *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 293–307, 2009.

[47] Y. Yang, N. Xiong, N. Y. Chong, and X. Defago, “A decentralized and adaptive flocking algorithm for autonomous mobile robots,” *The 3rd International Conference on Grid and Pervasive Computing–Workshops, IEEE Computer Society*, pp. 262–268, 2008.

[48] G. Lee and N. Y. Chong, “Adaptive flocking of robot swarms: Algorithms and properties,” *IEICE Transactions on Communications*, no. 9, pp. 2848–2855, 2008.

[49] W. Ren, R. W. Beard, and E. M. Atkins, “A survey of consensus problems in multi-agent coordination,” *Proceedings of the American Control Conference*, pp. 1859–1864, 2005.

[50] R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time–delays,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[51] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperative in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[52] L. Xiao and S. Boy, “Fast linear iterations for distributed averaging,” *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003.

[53] L. Xiao, S. Boy, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” *Proceedings of the International Conference on Information Processing in Sensor Networks*, pp. 63–70, 2005.

[54] L. Xiao, S. Boy, and S. Lall, "Distributed average consensus with time varying metropolis weights," *Automatica, June 2006. submitted.*, 2006.

[55] E. Kokiopoulou and P. Frossard, "Distributed classification of multiple observation sets by consensus," *Proceedings of the IEEE*, 2010.

[56] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 355–369, 2009.

[57] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network," *the 43th IEEE Conference on Decision and Control*, pp. 1914–1919, 2004.

[58] T. H. Chung, J. W. Burdick, and R. M. Murray, "Decentralized motion control of mobile sensing agents in a network," *the 44th IEEE Conference on Decision and Control*, 2005.

[59] 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, 2004.

[60] F. Zhang, D. M. Fratantoni, D. Paley, J. Lund, and N. E. Leonard, "Control of coordinated patterns for ocean sampling," *International Journal on Control*, vol. 80, no. 7, pp. 1186–1199, 2007.

[61] P. Yang, R. A. Freeman, and K. M. Lynch, "Distributed cooperative active sensing using consensus filters," *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pp. 405–410, 2007.

[62] K. M. Lynch, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Transaction on Robotics*, vol. 24, no. 3, pp. 710–724, 2008.

[63] B. Jung and G. S. Sukhatme, "Cooperative multi–robot target tracking," *The 8th International Symposium on Distributed Autonomous Robotic Systems*, pp. 81–90, 2006.

[64] H. M. La and W. Sheng, "Moving targets tracking and observing in a distributed mobile sensor network," *Proceedings of the 2009 American Control Conference (*ACC*), St. Louis, MO, USA*, 2009.

[65] T. Balch and R. C. Arkin, "Behavior–based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[66] J. Lawton, R. W. Beard, and F. Y. Hadaegh, "An adaptive control approach to satellite formation flying with relative distance constraints," *Proceedings of the American Control Conference*, pp. 1545–1549, 1999.

[67] E. Atkins, "Autonomous satellite formation assembly and reconfiguraton with gravity fields," *Proceedings of the Aerospace Conference*, pp. 783–795, 2002.

[68] Z. Lin, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Transactions on Automatic Control*, vol. 50, no. 1, pp. 121–127, 2005.

[69] R. Olfati-Saber, W. B. Dunbar, and R. M. Murray, "Cooperative control of multi–vehicle systems using cost graphs and optimization," *Proceedings of the American Control Conference*, pp. 2217–2222, 2003.

[70] M. Alighanbari and J. P. How, “Decentralized task assignment for unmanned aerial vehicles,” *Proceedings of the 44nd IEEE Conference on Decision and Control and European Control Conference*, pp. 5668–5673, 2005.

[71] D. Hristu and K. Morgansen, “Limited communication control,” *Systems and Control Letters*, vol. 37, pp. 193–205, 1999.

[72] R. Olfati-Saber, “Flocking with obstacle avoidance,” *Technique Report 2003-006, California Institute of Technology,* CA,, 2003.

[73] P. Ogren, E. Fiorelli, and N. E. Leonard, “Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment,” *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, 2006.

[74] R. Olfati-Saber and R. M. Murray, “Graph rigidity and distributed formation stabilization of multi–vehicle systems,” *Proceedings of the 41st IEEE Conference on Decision and Control*, pp. 2965–2971, 2002.

[75] T. Eren, P. N. Belhumeur, B. D. O. Anderson, and A. S. Morse, “Closing ranks in vehicle formations based on rigidity,” *Proceedings of the American Control Conference*, pp. 2959–2964, 2002.

[76] T. Eren, P. N. Belhumeur, B. D. O. Anderson, and A. S. Morse, “A framework for maintaining formations based on rigidity,” *Proceedings of the* IFAC *World Congress*, pp. 2752–2757, 2002.

[77] H. M. La and W. Sheng, “Flocking control of a mobile sensor network to track and observe a moving target,” *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (*ICRA*), Kobe, Japan*, 2009.

[78] H. M. La and W. Sheng, "Adaptive flocking control for dynamic target tracking in a mobile sensor network," *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (*IROS*), St. Louis, MO, USA*, 2009.

[79] H. M. La and W. Sheng, "Dynamic targets tracking and observing in a mobile sensor network," *Submitted to the Elsevier journal on Robotics and Autonomous Systems*, 2009.

[80] H. M. La, R. S. Lim, H. Chen, and W. Sheng, "Decentralized flocking control with minority of informed agents," *Proceedings of the IEEE Conference on Industrial Electronics and Applications (ICIEA), Beijing, China*, 2011.

[81] G. Folino and G. Spezzano, "An adaptive flocking algorithm for spatial clustering," *Parallel Problem Solving from Nature: Springer-Verlag Berlin Heidelberg*, vol. 2439, pp. 924–933, 2002.

[82] B. Jung and G. S. Sukhatme, "Tracking targets using multiple robots: The effect of environment occlusion," *Autonomous Robots Journal*, vol. 13, no. 3, pp. 191–205, 2002.

[83] Z. Tang and U. Ozguner, "Motion planning for mutitarget surveillence with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, 2005.

[84] A. Kolling and S. Carpin, "Cooperative observation of multiple moving targets: an algorithm and its formalization," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 935–953, 2007.

[85] H. Ariyoshi, I. Shirakawa, and H. Ozaki, "Decomposition of graph into compactly connected two terminal subgraphs," *IEEE Transactions on Circuit Theory*, vol. 18, no. 4, pp. 430–435, 1971.

[86] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, “Graph and minimum cut trees,” *Internet Mathematics*, vol. 1, no. 4, pp. 385–408, 2004.

[87] L. Kaufman and P. Rousseeuw, “Finding groups in data. an introduction to cluster analysis,” *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*, W*iley*, N*ew* Y*ork*, 1990.

[88] M. Girvaan and M. Newman, “Community structure in social and biological networks,” *Proceedings of the* N*ational* A*cademy of* S*ciences*, vol. 99, pp. 7821–67826, 2002.

[89] B. Derbel and M. Mosbah, “A fully distributed linear time algorithm for cluster network decomposition,” *Proceedings of Parallel and Distributed Computing and Systems*, 2004.

[90] A. Goebels, H. K. Buning, S. Priesterjahn, and A. Weimer, “Multi target partitioning of sets based on local information,” *Proceedings of the fourth IEEE Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST)*.

[91] A. Goebels, H. K. Buning, S. Priesterjahn, and A. Weimer, “Towards online partitioning of agent sets based on local information,” *Proceedings of Parallel and Distributed Computing and Networks*, 2005.

[92] B. Derbel, M. Mosbah, and A. Zemmari, “Fast distributed graph partition and application,” *IEEE International Parallel and Distributed Processing Symposium*, pp. 10–20, 2006.

[93] C. Bettstetter, “The cluster density of a distributed clustering algorithm in ad hoc networks,” *IEEE Communications Society*, pp. 4336–4340, 2004.

[94] R. Virrankoski and A. Savvides, "Tasc: topology adaptive spatial clustering for sensor networks," *IEEE International Conference on Mobile Adhoc and Sensor Systems*, p. 10 pp, 2005.

[95] A. Durresi and V. Paruchuri, "Adaptive clustering protocol for sensor networks," *IEEE Aerospace Conference*, pp. 1–8, 2005.

[96] A. Meka and A. K. Singh, "Distributed spatial clustering in sensor networks," *The 10th International Conference on Extending Database Technology*, pp. 980–1000, 2006.

[97] A. Belghith, I. Jemili, and M. Mosbah, "A distributed clustering algorithm without an explicit neighborhood knowledge," *International Journal of Computing and Information Science*, vol. 5, no. 1, pp. 24–34, 2007.

[98] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, pp. 1–39, 2005.

[99] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[100] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligent Research*, vol. 4, pp. 237–285, 1996.

[101] F. Fernandez, D. Borrajo, and L. Parker, "A reinforcement learning algorithm in cooperative multi-robot domain," *Journal of Intelligent and Robotic Systems*, vol. 43, pp. 161–174, 2005.

[102] X. Sun, T. Mao, J. D. Kralik, and L. E. Ray, "Cooperative multi-robot reinforcement learning: A framework in hybrid state space," *Proceedings of the 2009 IEEE/RSJ*

*International Conference on Intelligent Robots and Systems (*IROS 2009*), St. Louis, MO, USA*, 2009.

[103] Y. Sanz, J. de Lope, and J. A. Martin, "Applying reinforcement learning to multi-robot team coordination," *Hybrid Artificial Intelligence Systems. Springer Verlag*, 2008.

[104] F. Fernandez and D. Borrajo, "VQQL. applying vector quantization to reinforcement learning," *In RoboCup–99: Robot Soccer World Cup* III. *Springer Verlag*, 2000.

[105] L. E. Parker and C. Touzet, "Multi-robot learning in a cooperative observation task," *In Distributed Autonomous Robotic Systems 4, Springer*, pp. 391–401, 2000.

[106] T. B. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb, "Autonomous oceanographic sampling networks," *Oceanography*, vol. 6, no. 3, pp. 86–94, 1993.

[107] L. Huang, "Velocity planning for a mobile robot to track a moving target – a potential field approach," *Robotics and Autonomous Systems*, pp. 1–9, 2008.

[108] J. C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[109] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planing," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[110] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials, and coordinated control of groups," *Proceedings of the 40th IEEE Conference on Decision and Control*, pp. 2968–2973, 2001.

[111] "Rovio robot: http://www.wowwee.com/en/support/rovio," 2011.

[112] H. G. Tanner, A. Jadbabai, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, pp. 863–868, May 2007.

[113] Y. Liu and K. M. Passino, "Stable social foraging swarms in a noisy environment," *IEEE Transactions on Automatic Control*, vol. 49, no. 1, pp. 30–44, 2004.

[114] M. A. Fields, E. Haas, S. Hill, C. Stachowiak, and L. Barnes, "Effective robot team control methodologies for battlefield applications," *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (*IROS 2009*), St. Louis, MO, USA*, 2009.

[115] G. J. Gordon, "Stable function approximation in dynamic programming," *Proceedings of the 12th International Conference on Machine Learning*, pp. 261–268, 1995.

[116] C. Gaskett, D. Wettergreen, and A. Zelinsky, "Q-learning in continuous state and action spaces," *AI'09, LNAI 1747, Springer-Verlag, Berlin, Heidelberg*, pp. 417–428, 1999.

[117] J. C. Santamaria, R. S. Sutton, and A. Ram, "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adaptive Behaviour*, vol. 6, no. 2, pp. 163–218, 1998.

[118] W. D. Smart and L. P. Kaelbling, "Practical reinforcement learning in continuous spaces," *Proceedings of the International Conference on Machine Learning*, pp. 903–910, 2000.

[119] B. Baddeley, "Reinforcement learning in continuous time and space: Interference and not ill conditioning is the main problem when using distributed function approximators," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 38, no. 4, pp. 950–956, 2008.

[120] S. H. Lee, "Predator's attack-induced phase-like transition in prey flock," *Physical Letters A*, vol. 357, pp. 270–274, 2006.

[121] K. Morihiro, H. Nishimura, T. Isokawa, and N. Matsui, "Learning grouping and anti-predator behaviors for multi-agent systems," *Proceedings of the International conference on Knowledge-Based Intelligent Information and Engineering Systems*, pp. 426–433, 2008.

[122] M. Coggan, "Exploration and exploitation in reinforcement learning," *Fourth International Conference on Computational Intelligence and Multimedia Applications*, 2001.

[123] R. C. Dorf and R. H. Bishop, "Modern control systems," *Ninth Edition, Prentice Hall, Upper Saddle River, NJ 07458*, 2001.

[124] H. K. Khalil, "Nonlinear systems," *Third Edition, Prentice Hall, Upper Saddle River, NJ 07458*, 2002.

[125] "DOD/ONR MURI: adaptive sampling and prediction project. also available at http://www.princeton.edu/ dcsl/asap/,"

[126] X. Liao, L. Wang, and P. Yu, "Stability of dynamical systems," *First Edition, Elsevier, 1000* AE *Amsterdam, The Netherlands*, 2007.

[127] H. Choset, "Coverage of known spaces: the boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, pp. 247–253, 2000.

[128] Z. Feng, S. Jaewon, and R. James, "Information-driven dynamic sensor collaboration for target tracking," *IEEE Signal Processing Magazine*, vol. 19, no. 2, 2002.

[129] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. D. Schutter, "Active sensing for robotics – a survey," *Katholieke Universiteit Leuven, Department of Mechanical Engineering: Heverlee, Belgium*, 2003.

[130] J. Spletzer and C. Taylor, "Dynamic sensor planning and control for optimally tracking targets," *International Journal of Robotics Research*, 2003.

[131] S. Martinez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, 2006.

[132] R. A. Cortez and H. G. Tanner, "Radiation mapping using multiple robots," *In Proceedings of the 2nd Intl. Joint Topical Meeting on Emergency Preparedness and Response and Robotic and Remote Systems*, 2008.

[133] H. G. Tanner, R. A. Cortez, and R. Lumia, "Distributed robotic radiation mapping," *In G. J. Pappas O. Khatib and V. Kumar, editors, Experimental Robotics–The Eleventh International Symposium, volume 54 of Springer tracts in advanced robotics, Springer*, pp. 147–156, 2009.

[134] C. Zhang, D. Arnold, N. Ghods, A. Siranosian, and M. Krstic, "Source seeking with nonholonomic unicycle without position measurment part 1: Tuning of forward velocity," *IEEE Conference on Decision and Control*, pp. 3040–3045, 2006.

[135] S. Pang and J. A. Farrell, "Chemical plume source localization," *IEEE Transactions On Systems, Man, And Cybernetics*, vol. 36, no. 5, pp. 1068–1080, 2006.

[136] A. R. Mesquita, J. P. Hespanha, and K. Astrom, "ptimotaxis: A stochastic multiagent on site optimization procedure," *the 11th International Conference on Hybrid Systems: Computation and Control*, 2008.

[137] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, "Robust source-seeking hybrid controllers for autonomous vehicles," *American Control Conference*, pp. 1185–1190, 2007.

[138] HUKOYO, "Scanning range finder," *http://www.hokuyo-aut.jp*.

[139] Mobotix, "Hemispheric q24, fish-eyes camera," *http://www.mobotix.com*.

[140] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data associa-

tion," *Proceedings of IEEE International Conference on Robotics and Automation*, ICRA2001, vol. 2, pp. 1665–1670, 2001.

[141] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for a mobile robot navigation," *Proceedings of the 46th IEEE Conference on Robotics and Automation*, pp. 1398–1404, 1991.

[142] J. Fredslund and M. J. Matarie, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 837–846, 2002.

[143] K. Plarre and P. R. Kumar, "Tracking objects with networked scattered directional sensors," *EURASIP Journal on Advances in Signal Processing*, pp. 1–10, 2008.

[144] S. Oh, S. Sastry, and L. Schenato, "A hierarchical multiple-target tracking algorithm for sensor networks," *Proceedings of the IEEE International Conference on Robotics and Automation* ICRA 2005, pp. 2197–2202, 2005.

[145] T. Matsuyama and N. Ukita, "Real-time multi-target tracking by a cooperative distributed vision system," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1136–1150, 2002.

[146] J. Liu, M. Chu, and J. E. Reich, "Multi-target tracking in distributed sensor networks," *IEEE Signal Processing Magazine*, pp. 36–46, 2007.

[147] M. Ditzel, C. Lageweg, J. Janssen, and A. Theil, "Multi-target data aggregation and tracking in wireless sensor networks," *Journal of Networks*, vol. 3, no. 1, pp. 1–9, 2008.

[148] M. Lotfinezhad, B. Liand, and A. S. Sousa, "Adaptive cluster-based data collection in sensor networks with direct sink access," *Journal of Networks*, vol. 7, no. 7, pp. 884–897, 2008.

[149] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," *Proceedings of the IEEE INFOCOM*, pp. 629–640, 2004.

[150] D. R. Karger and C. Stein, "A approach to the minimum cut problem," *Journal of the ACM*, vol. 43, no. 4, pp. 601–640, 1996.

[151] W. Sheng, Y. Shen, and N. Xi, "Motion planning in robotized sensor networks for aircraft rivet inspection," *IEEE International Conference on Advanced Intelligent Mechatronics,IEEE/ASME*, pp. 638–643, 2005.

[152] M. J. Rattigan, M. Maier, and D. Jensen, "Graph clustering with network structure indices," *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[153] T. I. Zohdi, "Computational design of swarms," *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 2205–2219, 2003.

[154] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[155] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2:Improving the strength pareto evolutionary algorithm," *TIK-Report 103*-S*wiss federal Institute of Technology (*ETH*)* Z*urich*, pp. 1–21, 2001.

[156] L. Davis, "Handbook of genetic algorithms," *Van* N*ostrand Reinhold*, N*ew* Y*ork*, 1991.

[157] H. Lu and G. G. Yen, "Rank-density-based multi-objective genetic algorithm and benchmark test function study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 325–343, 2003.

[158] H. M. La and W. Sheng, "Robust adaptive control with leakage modification for a nonlinear model of ionic polymer metal composites (IPMC)," *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics (*ROBIO*), Bangkok, Thailand*, 2008.

[159] H. M. La and W. Sheng, "Flocking control of multiple agents in noisy environments," *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (*ICRA*),Anchorage, Alaska, USA*, 2010.

[160] H. M. La, R. S. Lim, and W. Sheng, "Hybrid system of reinforcement learning and flocking control in multi-robot domain," *Proceedings of the the Conference on Theoretical and Applied Computer Science (*TACS*), Stillwater, Oklahoma, USA*, 2010.

[161] S. Lima, "Back to the basics of anti-predatory vigilance: the group-size effect," *Journal of Theoretical Biology*, vol. 49, no. 1, pp. 11–20, 1995.

[162] W. Hamilton, "Geometry for the selfish herd," *Journal of Theoretical Biology*, vol. 31, no. 1, pp. 295–311, 1971.

[163] G. Turner and T. Pitcher, "Attack abatement: a model for group protection by combined avoidance and dilution," *American Naturalist*, vol. 128, no. 2, pp. 228–240, 1986.

[164] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 1998.

[165] "Center of excellence for great lakes, harmful algal bloom event response. also available at http://www.glerl.noaa.gov/res/centers/habs/habs.html,"

[166] D. Bauso, L. Giarre', and R. Pesenti, "Distributed consensus in networks of dynamic agents," *Proceedings of the 44nd IEEE Conference on Decision and Control and European Control Conference*, pp. 7054–7059, 2005.

[167] P. A. Bliman and G. Ferrari-Trecate, "Average consensus problems in networks of agents with delayed communications," *Proceedings of the 44nd IEEE Conference on Decision and Control and European Control Conference*, pp. 7066–7071, 2005.

[168] V. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," *Proceedings of the 44nd IEEE Conference on Decision and Control and European Control Conference*, pp. 2996–3000, 2005.

[169] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[170] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," *Proceedings of the American Control Conference*, pp. 951–956, 2003.

[171] L. Moreau, "Stability of multiagent systems with time–dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[172] C. Godsil and G. Royle, "Algebraic graph theory," *Graduate Texts in Mathematics, Springer*, vol. 207, 2001.

[173] R. A. Horn and C. R. Johnson, "Matrix analysis," *Cambridge University Press*, 1987.

[174] S. Nelson and M. Neumann, "Generalization of the projection method with applications to sor theory for hermitian positive semidefinite linear systems," *Numerische Mathematik*, vol. 51, pp. 123–141, 1987.

[175] A. Makarenko and H. Durrant-Whyte, "Decentralized data fusion and control in active sensor networks," *In Proceedings of the Seventh International Conference on Information Fusion*, 2004.

[176] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," SIAM *Review*, vol. 46, no. 4, pp. 667–689, 2004.

[177] E. W. Frew, C. Dixon, and J. E. M. Stachura, "Active sensing by unmanned aircraft systems in realistic communication environments," *the 2009* IFAC *Workshop on Networked Robotics*, 2009.

[178] H. Zhou and S. Sakane, "Mobile robot localization using active sensing based on bayesian network inference," *the Elsevier journal on Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 292–305, 2007.

[179] Wikipedia, "Fisher information," *http://en.wikipedia.org*.

VITA

Hung Manh La

Candidate for the Degree of

Doctor of Philosophy

Dissertation: COOPERATIVE CONTROL, LEARNING AND SENSING IN MOBILE SENSOR NETWORKS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Bac Giang, Vietnam, 1978.

Education:

Received the B.S. degree from Thai Nguyen University of Technology, Thai Nguyen city, Thai Nguyen, Vietnam, 2001, in Major Electrical Engineering
Received the M.S. degree from Thai Nguyen University of Technology, Thai Nguyen city, Thai Nguyen, Vietnam, 2003, in Major Electrical Engineering
Completed the requirements for the degree of Doctor of Philosophy with a major in Electrical Engineering Oklahoma State University in August, 2011.

Experience:

Research Assistant, Oklahoma State University, from August 2007 to August 2011.
Lecturer and researcher of the Electronics Engineering Department, Thai Nguyen University of Technology, from April 2005 to August 2007.
Lecturer and researcher of the Electrical Engineering Department, Thai Nguyen University of Technology, from September 2001 to April 2005.

Name: Hung Manh La Date of Degree: December, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: COOPERATIVE CONTROL, LEARNING AND SENSING IN MOBILE SENSOR NETWORKS

Pages in Study: 206 Candidate for the Degree of Doctor of Philosophy

Major Field: Electrical Engineering

Abstract

Scope and Method of Study:

Mobile sensor networks (MSNs) have great potential in many applications including environment exploring and monitoring; search and rescue; cooperative detection of toxic chemicals, etc. Motivated by the broad and important applications of MSNs and inspired by the cooperative ability and the intelligence of fish schools and bird flocks, this dissertation develops cooperative control, learning and sensing algorithms in a distributed fashion for MSNs to realize coordinated motion control and intelligent situational awareness.

Findings and Conclusions:

The proposed algorithms can allow MSNs to track a moving target efficiently in cluttered environments and even when only a very small subset of the sensor nodes know the information of the target; adjust their size (shrink/recover) in order to adapt to complex environments while maintaining the network connectivity and topology; form a lattice structure and maintain the cohesion even when the measurements are corrupted by noise; track multiple moving targets simultaneously and efficiently in a dynamic fashion; learn to evade the enemy (predators) in a distributed fashion while maintaining the network connectivity and topology; estimate and build the map of a scalar field. We conducted several experiments using both simulation and real mobile robots to show the effectiveness of the proposed algorithms. We also extended our framework to cooperative and active sensing in which the mobile sensors have the ability to adjust their movements to adapt to the environments in order to improve the sensing performance in a distributed fashion.

ADVISOR'S APPROVAL: _____