Multi-Agent Area Coverage Control using Reinforcement Learning Techniques

by

Adekunle Akinpelu Adepegba

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

Mechanical Engineering
Faculty of Engineering
University of Ottawa

Abstract

An area coverage control law in cooperation with reinforcement learning techniques is proposed for deploying multiple autonomous agents in a two-dimensional planar area. A scalar field characterizes the risk density in the area to be covered yielding nonuniform distribution of agents while providing optimal coverage. This problem has traditionally been addressed in the literature to date using locational optimization and gradient descent techniques, as well as proportional and proportional-derivative controllers. In most cases, agents' actuator energy required to drive them in optimal configurations in the workspace is not considered. Here the maximum coverage is achieved with minimum actuator energy required by each agent.

Similar to existing coverage control techniques, the proposed algorithm takes into consideration time-varying risk density. These density functions represent the probability of an event occurring (e.g., the presence of an intruding target) at a certain location or point in the workspace indicating where the agents should be located. To this end, a coverage control algorithm using reinforcement learning that moves the team of mobile agents so as to provide optimal coverage given the density functions as they evolve over time is being proposed. Area coverage is modeled using Centroidal Voronoi Tessellation (CVT) governed by agents. Based on [1, 2] and [3], the application of Centroidal Voronoi tessellation is extended to a dynamic changing harbour-like environment.

The proposed multi-agent area coverage control law in conjunction with reinforcement learning techniques is implemented in a distributed manner whereby the multi-agent team only need to access information from adjacent agents while simultaneously providing dynamic target surveillance for single and multiple targets and feedback control of the environment. This distributed approach describes how automatic flocking behaviour of a team of mobile agents can be achieved by leveraging the geometrical properties of centroidal Voronoi tessellation in area coverage control while enabling multiple targets tracking without the need of consensus between individual agents.

Agent deployment using a time-varying density model is being introduced which is a function of the position of some unknown targets in the environment. A nonlinear derivative of the error coverage function is formulated based on the single-integrator agent dynamics. The agent, aware of its local coverage control condition, learns a value function online while leveraging the same from its neighbours. Moreover, a novel computational adaptive optimal control methodology based on work by [4] is proposed that employs the approximate dynamic programming technique online to iteratively solve the algebraic Riccati equation with completely unknown system dynamics as a solution to linear quadratic regulator problem. Furthermore, an online tuning adaptive optimal control algorithm is implemented using an actor-critic neural network recursive least-squares solution framework. The work in this thesis illustrates that reinforcement learning-based techniques can be successfully applied to non-uniform coverage control. Research combining non-uniform coverage control with reinforcement learning techniques is still at an embryonic stage and several limitations exist. Theoretical results are benchmarked and validated with related works in area coverage control through a set of computer simulations where multiple agents are able to deploy themselves, thus paving the way for efficient distributed Voronoi coverage control problems.

Acknowledgements

I am especially grateful to my supervisors Dr. Davide Spinello and Dr. Suruz Miah for their guidance and support over the course of my master's program research work. Their encouragement to unite my different research interests and their objective criticism has been a constant source of inspiration as I strive to always improve my work. They both gave me enough freedom to conduct my research with diligence and a careful attention to detail has helped me become a better independent researcher. Their insights and disciplined approach are both unmeasurable and invaluable. Working with them and witnessing their talents, dedication, and professionalism has been an immensely rewarding experience. It is indeed a genuine pleasure to acknowledge their contributions and I look forward to continue working with both of them as I pursue a Ph.D.

I am also deeply indebted to the authors whose works have been cited throughout this thesis. Special thanks are due to those authors who have significantly influenced my thinking about non-uniform coverage control and reinforcement learning. Their influence on this thesis is boundless for which I am eternally grateful.

I am especially indebted to my family and friends whose love, support and constant encouragement kept me grounded throughout my graduate studies. Above all, I owe a special debt of gratitude to my parents Mr. and Mrs. Adepegba. As educators, their passion for teaching and guiding young minds has implanted in me a passion and desire to be a seeker of knowledge which has cultivated my appetite for research. Lastly, I would like to appreciate my friends Tolani, Gifty, Dayo, Bunmi and Akin for their support and prayers during this program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The problem of cooperative multi-agent decision making and control is to deploy a group of agents over an environment to perform various tasks including sensing, data collection and surveillance. This topic covers a wide range of applications in various fields especially in military and civilian domains, such as harbour protection [7, 8, 9] perimeter surveillance [10, 11], search and rescue missions [12, 13], and cooperative estimation [14].

Relatively recently, researchers have proposed various solutions to several interesting sensor network coverage problems based on the work [15]. Recent contributions and meaningful extensions of the framework formulated in [3] can be found in [16, 17]. In [18] the problem of limited-range interaction between agents was addressed. The work described in this paper can be posed in the framework of the coverage control problem presented in [1, 9, 19, 20] which uses the geometrical notion of a Voronoi partition to assign the agents to different parts of an environment.

Many processes can leverage the utility of adaptive controllers capable of learning to optimize a certain cost function online in real time. Reinforcement Learning is an example of such learning methods. The user sets a goal by specifying a suitable reward function for the reinforcement learning controller which then learns to maximize a cumulative reward received over time in order to reach that goal. The proposed recursive least square actor-critic neural network solution implemented in this paper is preferred over the gradient descent approach [21, 22] as it offers significant advantages including the ability to extract more information from each additional observation [23, 24] and would thus be expected to converge with fewer training samples. Moreover, this thesis implements the synchronous discrete-time adaptation of both actor and critic neural networks.

A coverage optimization problem is considered for a mobile sensor network. The concept of locational optimization and Centroidal Voronoi Tessellation (CVT) is used to model the coverage metric provided by multi-agent search in a non-uniform coverage control environment. With this motivation, this thesis focuses on developing reinforcement Learning based controllers for cooperative coverage control. A major focus of this thesis is modeling non-uniform coverage control problem as a linear quadratic regulator problem and solving it using actor-critic neural network value function approximation and recursive least-squares methods, as the preferred reinforcement learning techniques.

1.1 Problem Statement

Given a two-dimensional planar workspace requiring area coverage and surveillance of randomly moving targets, agents with a limited sensing range need to work cooperatively to maximize the coverage of the workspace and provide good tracking of the targets. Consider a scenario where a group of agents are deployed to search and detect some targets in the described workspace. Their task is to spread out over the harbour to provide coverage while the agents broadcast their information about the environment to their neighbours. This information allows the agents to find where they are mostly needed and to aggregate in those areas. The probability of target existence in the harbour is prescribed by a time-varying density function but the exact location is unknown.

This thesis work proposes a multi-agent area coverage control law in conjunction with reinforcement learning techniques for deploying multiple autonomous agents in the workspace. Voronoi diagrams and centroidal Voronoi tessellation are used to determine the optimal position of mobile sensors for active sensing and coverage control of the workspace. The solution of the proposed control law is based on the hypothesis that a given point can be monitored given that its distance from the agent's position lies within the agent's Voronoi cell. The proposed algorithm employs linear-in-the-parameters actor-critic Neural Network approximators to implement online the recursive least-squares solution.

The scope of the thesis does assume two specific limitations with respect to deployment and task performance of the agents within the workspace. First, the total number of agents available is limited. In an unlimited scenario, it would be possible to deploy a large enough number of agents in the workspace so that each agent's Voronoi region is small enough that a target traveling through any part of the agent's target tracking region will be within the desired agent-to-target safe standoff tracking distance. This scenario removes the need for mobile agents and makes stationary area coverage sufficient. Second, the sensing range of the agents is limited, so that the agents are not aware of the entire workspace and correspondingly are not aware of all the targets throughout the workspace. However, both limitations are mitigated by using centroidal Voronoi tessellation to deploy the agents in a distributed manner whereby the multi-agent team only need to access information from adjacent agents (neighbours) while simultaneously providing dynamic target surveillance of the workspace. Although centroidal Voronoi tessellation is applied only in a static environment [3], the harbour is assumed to be stationary, therefore, centroidal Voronoi tessellation is a valid solution to the optimal coverage problem with active sensing.

This problem can be solved using novel actor-critic solution approach presented in this thesis to enable autonomous operation of the mobile actuators/sensors network and satisfy area coverage mission requirements. Particularly, the main contributions of this research are a set of promising solutions addressing the following questions:

> **Q1:** How to enable agents to react and adjust quickly throughout the harbour so that they can aggregate in high risk areas to adequately defend against threats and achieve maximum coverage?
>
> **Q2:** How to organize agents to achieve efficient coordination while learning to cooperate based only on local interaction with each other and incomplete knowledge?

Q3: How to optimize agents' information sharing with each other so that agents can simultaneously intercept a threat and maximize coverage while minimizing agent's actuator energy?

Q4: How to model an agent's dynamics, based on a mathematical formulation, so that an agent's optimal actions at each state, during an area coverage mission, are optimized?

1.2 Thesis Objectives and Contributions

The goal is of this thesis is to move and adjust a network of mobile agents with limited sensing capabilities to quickly achieve maximum coverage of a dynamically changing harbour-like environment and continuously maintaining minimum actuator energy required by each agent while ensuring that a defined coverage metric is maximized. The agents should speedily make decisions facilitated by reinforcement learning while achieving a common goal. Furthermore, they should be able to adjust to changing harbour dynamics and automatically aggregate to intercept detected threats in a coordinated fashion.

The contributions of the thesis includes the formulation of the coverage control problem as a linear quadratic regulator problem using a developed nonlinear error dynamics linearized for discrete-time multi-agent systems, where information flow is restricted by a communication graph structure. A two-pronged optimal control approach is proposed using a distributed energy-efficient local controller that maximizes coverage in such a way that agents employ minimum actuator energy to place themselves in optimal configurations which guarantee that agents maintain their energy during the deployment task. An adaptive controller using actor-critic neural network approximation that asymptotically drives agents in optimal configuration such that the coverage is maximized regardless of the complexity of the time-varying density throughout the workspace is developed.

1.3 Thesis Outline

The main concepts of this thesis are structured as follows. In Chapter 2, an overview of literature on key Voronoi-based area coverage problem and reinforcement learning theoretical techniques from computational intelligence point of view is presented under a unified framework. Also, the time-invariant and time-varying risk density functions are modelled. In Chapter 3, a first order control law is introduced and the nonlinear error dynamics is formulated. Also, an energy efficient control problem is defined as an optimal control problem. In Chapter 4, an online tuning algorithm for solving the CT algebraic Riccati Equation is introduced with some numerical simulations. Furthermore, an actor-critic solution using recursive least-squares with two neural network structures is presented and the proposed multi-agent area coverage control law in conjunction with reinforcement learning techniques is then illustrated. Following that, the key steps of the developed algorithm are

summarized. Finally, the Chapter concludes with some numerical simulations followed by conclusion and future research avenues presented in Chapter 5.

Chapter 2

Literature review and Theoretical Background

2.1 Introduction

This Chapter starts with a literature review on various applications of multi-robot system in area coverage and an overview of different reinforcement learning techniques. It briefly introduces a methodology outlining solution paths for solving the cooperative area coverage control problem. It provides the relevant theoretical framework for solving the research problem stated in Chapter 1. The solution format is explained and formulated.

In section 2.3, Voronoi diagrams and Centroidal Voronoi Tessellation (CVT) are presented as a theoretical framework for analyzing multi-agent area coverage control problems. The time-invariant and time-varying risk density which describes the model of the of risk are modeled. Also, the solution is formulated using a CVT-based adaptive optimization approach. Reinforcement learning is presented as a solution to optimal control problems in section 2.4. Markov decision processes which serves as the theoretical framework describing how the behaviour of agents changes over time and the convergence properties of reinforcement learning algorithms is examined in sub-section 2.4.2. In addition to Markov decision processes, optimal control theory serves as theoretical foundation upon which all reinforcement learning-based methods and algorithms is used to provide online synthesis of optimal control policies. These methods include dynamic programming, approximate dynamic programming, temporal-difference learning, adaptive critic (using actor-critic neural networks architecture) methods, and recursive least squares methods. Section 2.5 gives a brief description of networked multi-robot systems and environment coverage while section 2.6 presents a brief summary and discussion of the chapter.

2.2 Literature Review

The use of multi-agent teams in area coverage control has gained prominence in recent years. This is attributed to the fact that multi-agent teams can outperform a single agent

for the same task while expending less actuator energy in less time. Multi-agent teams can be used in a variety of area coverage mission including: surveillance in hostile environments (i.e. areas contaminated with biological, chemical or even nuclear wastes), environmental monitoring (i.e. air quality monitoring, forest monitoring) and law enforcement missions (i.e. border patrol) etc.

Several approaches have been proposed in the literature considering the optimal area coverage control problem using a team of mobile agents. In the last fifteen years, some researchers have offered various solutions to a lot of interesting sensor network coverage problems based on the work of [15]. Recent contributions and meaningful extensions of the framework formulated in [3] have been recommended in the literature [16, 17]. In [18] the problem of limited-range interaction between agents was addressed. A similar approach for a convex environment, is proposed in [25], where additionally the agents estimate a function indicating the relative importance of different areas in the environment, using information from the sensors. Another approach of providing simultaneous coverage and tracking (SCAT) of moving targets with agent networks is proposed in [26].

Reinforcement learning refers to an agent which interacts with its environment and modifies its actions. Learning happens through trial and error and is based on a cause and effect relationship between the actions and the rewards/punishment. Decisions/actions which lead to a satisfactory outcome are reinforced and are more likely to be taken when the same situation arises again. Several reinforcement learning methods have been developed and successfully applied in machine learning to learn optimal policies for finite-state finite-action discrete-time Markov decision processes.

The temporal difference method [27] for solving Bellman equations leads to a family of optimal adaptive controllers; that is, adaptive controllers that learn online the solutions to optimal control problems without knowing the full system dynamics. Temporal difference learning is true online reinforcement learning, wherein control actions are improved in real time based on estimating their value functions by observing data measured along the system trajectories [28].

Actor-critic (AC) methods, introduced by [29], implement the policy iteration algorithm online, where the critic is typically a Neural Network which implements policy evaluation and approximates the value function, whereas the actor is another neural network which approximates the control. The critic evaluates the performance of the actor using a scalar reward from the environment and generates a temporal difference error. The actor-critic neural networks are updated using gradient update laws based on the temporal difference error.

A policy iteration algorithm using Q-functions for the discrete-time problem and convergence to the state feedback optimal solution is proven in [30]. Convergence of policy iteration for continuous-time linear quadratic regulator was first proved [31]. [28] used a Hamilton Jacobi Bellman framework to derive algorithms for value function approximation and policy improvement, based on a discrete-time version of the temporal difference error. [32] also used the Hamilton Jacobi Bellman framework to develop a stepwise stable iterative approximate dynamic programming algorithm for continuous-time input-affine systems with an input quadratic performance measure. [33] proposed a least-squares successive ap-

proximation solution to the discrete time Hamilton Jacobi Bellman and continuous time Hamilton Jacobi Bellman, where a neural network is trained offline to learn the Hamilton Jacobi Bellman solution.

Since actor-critic methods are adaptable to online implementation, they have become an important subject of research, particularly in the controls community [22, 32, 34, 35]. The performance of approximate dynamic programming-based controllers has been successfully demonstrated on various nonlinear plants with unknown dynamics. [36] used a Dual Heuristic Programming based approximate dynamic programming for jet aircraft simulations. Various modifications to approximate dynamic programming algorithms have since been proposed [33].

Recently, [37] suggested an online actor-critic algorithm for solving a continuous-time infinite-horizon optimal control problem. Most recent work by [38] presents a cooperative control of multi-agent systems as an extension of optimal control and adaptive control design methods to multi-agent systems on communication graphs. It develops Riccati design techniques for general linear dynamics for cooperative state feedback design and cooperative dynamic output feedback design using neural adaptive design techniques for multi-agent nonlinear systems with unknown dynamics. It solves the linear quadratic regulator directly using the Hamiltonian function.

A combined deploy and search strategy for solving the area coverage control problem in an environment with the time-varying priority function using a centralized Voronoi partitioning is proposed in [1] and [19], where the mobile agents are autonomously deployed to maximize the reduction of uncertainty about the environment at each step.

2.3 Theoretical Framework of Coverage Control Problems

In this section, Voronoi diagrams and Centroidal Voronoi Tessellation (CVT) are presented as a theoretical framework for analyzing multi-agent area coverage control problems. In general, an agent refers to a dynamical system. However, in the context of this work the term agent is interchangeable with actuator, interceptor, vehicle or robot. A multi-agent system is defined as a cooperating group of autonomous mobile agents capable of motion, communication, sensing and computation. Also, the term target is interchangeable with threat or intruder. For the purpose of coordinating multiple agents to cover the points in the environment, it is desired to design a position-control strategy based on the centroidal Voronoi configuration.

Voronoi Tessellation

Voronoi Tessellation also known as Voronoi diagram, Voronoi decomposition or Voronoi partition involves the partitioning of a surface or a workspace into regions based on distances to points in a specific subset of the surface. These set of points are pre-specified and

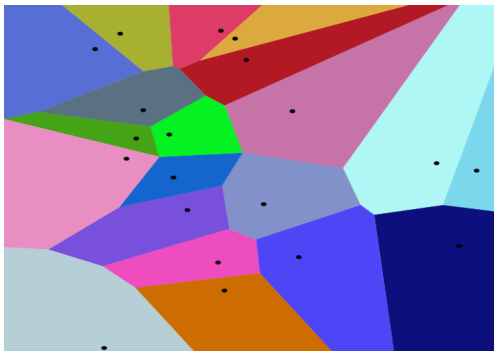


Figure 2.1: Euclidean based Voronoi diagram [5]

they are regarded as generating points or generators. Voronoi partitions can be described as a diagram obtained by drawing a line that is equidistant between two set of points and perpendicular to the line between them. A Voronoi cells or regions consist of all points closer to the generator in the cell than to any other generator. This approach is named after Georgy Voronoy.

Example: Voronoi tessellation has numerous application in many fields including science and technology. For example, If there exist a group of banks in a flat community and we aim to estimate the amount of customers of a given branch. Assuming that all other circumstances including quality of service, charges and benefits are the same in all branches, we can logically assume that the customer's choice of preference is determined by proximity in terms of distance. Therefore, the estimate of the number of customers can be obtained from the Voronoi cell of the aforementioned bank described by a point in the flat community. The Euclidean distance can be defined as

$$d[(b_1, b_2), (c_1, c_2)] = \sqrt{(b_1 - c_1)^2 + (b_2 - c_2)^2} \tag{2.1}$$

Likewise the Manhattan distance can be defined by

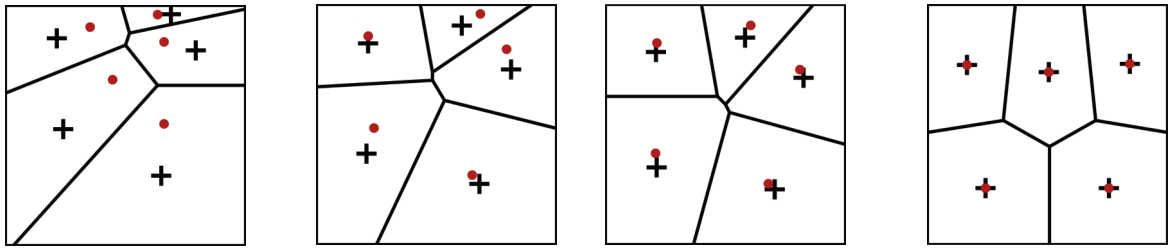
$$d[(b_1, b_2), (c_1, c_2)] = |b_1 - c_1| + |b_2 - c_2| \tag{2.2}$$

Various algorithms has been proposed for generating Voronoi partitions which include Incremental algorithm, Fortune algorithm, Divide and Conquer rule and Discretization algorithm.

Centroidal Voronoi Tessellation

A Voronoi tessellation is called centroidal when the generator of each of Voronoi region or cell is also the center of mass or the centroid. It can also be described as an optimal partition which corresponds to optimal distribution of the generating points. Numerous

algorithms has been used to generate centroidal Voronoi tessellation including Llyod's algorithm. The algorithm also known as Voronoi iteration or relaxation is named after Stuart P.Llyod for obtaining evenly spaced sets of points in a subsets of Euclidean spaces [6]. The Voronoi tessellation is obtained, the centroid is calculated and the generating points are moved to the centroids of the Voronoi partition.



(a) First iteration **(b)** Second iteration **(c)** Third iteration **(d)** Fifteenth iteration

Figure 2.2: Llyod's algorithm [6]

Figure 2.2 shows an example of Llyod's algorithm with no threat or object on the plane. It shows the Voronoi diagram of the current points at each iteration. The plus sign denotes the centroids of the Voronoi region [6]. A centroidal Voronoi tessellation is found when the points in each cell are really close to the centroids.
Consider a group of n homegeneous agents deployed in a 2D area $\Omega \subset \mathbb{R}^2$, with the position of the ith, $i = 1, \ldots, n$, agent denoted by $\mathbf{p}_i = [x_i, y_i]^T$. We define a coverage metric as a local optimization function as

$$\mathcal{H}(p, \mathcal{V}) = \sum_{i=1}^{n} \mathcal{H}_i(\mathbf{p}_i) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} \max_i f(r_i)\phi(\mathbf{q})d\mathbf{q}, \tag{2.3}$$

where r_i is the distance between $\mathbf{p}_i$ and any point $\mathbf{q}$ in Ω such that $\Omega = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots, \cup \mathcal{V}_n$, where the area ($i$th Voronoi cell) $\mathcal{V}_i \subset \Omega$ belongs to agent i, $i = 1, \ldots, n$. The Voronoi region $\mathcal{V}_i$ of the ith agent is the locus of all points that are closer to it than to any other agents, *i.e.,*

$$\mathcal{V}_i = \{\mathbf{q} \in \Omega | \parallel \mathbf{q} - \mathbf{p}_i \parallel \leq \parallel \mathbf{q} - \mathbf{p}_j \parallel, \ i \neq j, \ i \in 1, 2, ..., n\}. \tag{2.4}$$

Agents i and j are called neighbours if they share an edge (i.e., if $V_i \cap V_j \neq 0$). The set of all neighbours of agent i is denoted by $N_i[j]$. Furthermore, we assume that the sensing performance function $f(r_i)$ of the ith agent is Lebesgue measurable and that it is strictly decreasing with respect to the Euclidean distance $r_i = \|\mathbf{q} - \mathbf{p}_i\|$. Hence, the sensing performance function of the ith, $i \in \mathcal{I}$, agent is defined as

$$f(r_i) = \mu \exp(-\Lambda r_i^2), \tag{2.5}$$

where $\mu, \Lambda > 0$. The function ϕ, the risk density function represents the probability of an event taking place over Ω. It also provides the relative importance of a point in the

workspace where agents should converge more towards area with higher risk. The coverage metric (2.3) encodes how rich or poor the coverage is, that is, an increase in the value of $\mathcal{H}$ signifies the corresponding distribution of the agents achieves better coverage of the area. We aim to maximize this function in order to solve the local optimization problem which is to achieve optimal configuration of the agents. One of the underlying assumption in this thesis is that the agents are fully actuated and omnidirectional which is not always the case with actual vehicle dynamics and control. In practice, mobile robots have physical limitations such bounds in velocity and torques which should not be exceeded and so should be considered in the design of the system. Also, the controller can be modified to account for the dynamics of the agents. The area Ω is defined as a convex environment and by letting

$$\widetilde{m}_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \widetilde{\phi}(\mathbf{q},\, \mathbf{p}_i) d\mathbf{q}, \quad \widetilde{\mathbf{c}}_{\mathcal{V}_i} = \frac{1}{\widetilde{m}_{\mathcal{V}_i}} \int_{\mathcal{V}_i} \mathbf{q}\widetilde{\phi}(\mathbf{q},\, \mathbf{p}_i) d\mathbf{q}, \tag{2.6}$$

$$\widetilde{\phi}(\mathbf{q},\, \mathbf{p}_i) = -2\phi(\mathbf{q})\Big(\partial f(r_i)/\partial(r_i^2)\Big) \tag{2.7}$$

where $\widetilde{m}_{\mathcal{V}_i}$ and $\widetilde{\mathbf{c}}_{\mathcal{V}_i}$ in (2.6) represent the generalized mass and centroid of the voronoi cell $\mathcal{V}_i$ respectively while (2.7) represents the generalized risk density function. The centroid of the voronoi cell also represent the critical point for the locational optimization function [15] i.e. centroidal Voronoi tessellation are the optimal partitions of the coverage metric $\mathcal{H}$ and if the agent's configuration give rise to centroidal Voronoi tessellation, it can be referred to as centroidal Voronoi configuration.

2.3.1 Modeling Area Coverage Control using Time-invariant Density function

Consider a first order dynamics defined by

$$\dot{\mathbf{p}}_i = \mathbf{u}_i, \tag{2.8}$$

where $\mathbf{p}(t)$ represents the position of the ith agent at the time, t while $\mathbf{u}_i(t) \in \mathbb{R}^2$ is the 2D velocity input of the ith agent. The continuous-time version of the Llyod algorithm proposed in [3] is

$$\dot{\mathbf{p}}_i = K_p(\mathbf{c}_{\mathcal{V}_i} - \mathbf{p}_i), \tag{2.9}$$

and it guarantees that the system converges to a centroidal Voronoi configuration. This well established result from [1] is summarized in the following Lemma.

Lemma 1 *Given an autonomous risk density $\phi(\mathbf{q}), \mathbf{q} \in \mathbf{\Omega}$, a multi-agent system with dynamics $\dot{\mathbf{p}}_i = -K_p(\mathbf{p}_i - \mathbf{c}_{\mathcal{V}_i})$, $i = 1, 2, \ldots, n$ converges asymptotically to the set of centroidal Voronoi configurations, where K_p is a constant of proportionality and a positive gain.*

By applying **Lemma 1** under the first order dynamics (2.8) and considering $\mathcal{H}$ as a Lyapunov function according to [3], it can be proven by LaSalle's invariance principle that the multi-agent system asymptotically converges to a centroidal Voronoi configuration.
Given that the lyapunov function $\mathcal{H}(\mathbf{p}_i)$, which defines the coverage metric only depends on $\mathbf{p}_i$ and not on time, t,

$$\begin{aligned}\frac{d}{dt}\boldsymbol{\mathcal{H}}(\mathbf{p}_i) &= \sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{p}_i}\boldsymbol{\mathcal{H}}(\mathbf{p}_i, t)\dot{\mathbf{p}}_i \\ &= \sum_{i=1}^{n} 2m_i(\mathbf{p}_i - \mathbf{c}_{\mathcal{V}_i})^T\Big(- K_p(\mathbf{p}_i - \mathbf{c}_{\mathcal{V}_i})\Big) \\ &= -2K_p \sum_{i=1}^{n} m_i\|\mathbf{p}_i - \mathbf{c}_{\mathcal{V}_i}\|^2\end{aligned}$$

According to [1, 3], asymptotic stability of the system under the feedback law (2.9) is guaranteed. Furthermore, it was shown in [6], that the control strategy in which agent moves towards the centroid of its Voronoi cell locally solves the area coverage control problem.

However, if ϕ is time-varying, Lloyd's coverage control law does not stabilize the multi-agent system to a centroidal Voronoi tessellation [1]. A new set of algorithms for handling the time-varying case is discussed in the next section.

2.3.2 Modeling Area Coverage Control using Time-varying Density function

In real world scenarios, many events can be modelled using time-varying risk density in preference to time-invariant risk density. This is important when the point of interest is dynamically changing in the environment to be covered. Practical examples include the presence of targets entering a domain in an attempt to damage a high value unit, 2011 slave lake wildfire in Alberta, Canada and the 2011 Fukushima Daiichi nuclear disaster in Fukushima, Japan. Various techniques have been proposed in solving the problem of coverage control with time-varing risk density. In [1], an approach for controlling a multi-agent system to provide surveillance over the domain of interest by employing optimal coverage ideas on general time-varying density functions was presented. A control law that causes the agents to track the density function while providing optimal coverage of the area was proposed. Cortes and Bullo [15] worked on vehicles that perform distributed sensing tasks and refer to them as active sensor networks. Such systems are being developed for applications in remote autonomous surveillance, exploration, information gathering, and automatic monitoring of transportation systems. The problem of simultaneously covering an environment with time-varying density functions and tracking intruders (SCAT) was addressed in [26]. A framework was proposed for solving three important problems in the field of cooperative robotics which includes (i) environment coverage, (ii) task assignment, and (iii) target (intruder) tracking.

In this thesis, an environment with time-varying risk density is considered, where ϕ is a function which quantifies the risk (or importance) associated to points in the domain. For example, in a harbor protection scenario, points which are considered to be more valuable and important to defend would have assigned higher values of ϕ. In the same scenario, external threats entering the domain would locally increase the risk, and make the function ϕ time varying and non-autonomous, as the kinematics of the threats is in general uncontrollable and independent of the kinematics of defending agents in the domain. A non-uniform risk results into a non-uniform distribution of the agents, as more of them are deployed in areas with higher risk and fewer are deployed in areas with lower risk (i.e lower value of $\phi(\mathbf{q}, t)$). In [15], Cortes and Bullo presented various density functions that lead the multi-vehicle network to predetermined geometric patterns including simple density functions that lead to segments, ellipses, polygons, or uniform distributions inside convex environments. The choice of risk density function in this thesis is influenced by the motion of uncontrolled objects inside a given area. This objects are external threats and are also referred to as targets. The motion of the targets varies the density function inside this area, thereby influencing the defined time-dependent coverage metric and therefore referred to as a non-autonomous coverage function. We define a risk density function which attains maximum value at the center of the targets and decreases at a far distance from its center. If the total density function is expressed by

$$\phi(\mathbf{q}, t) = \phi_0(\mathbf{q}) + \sum_{h=\mathbf{1}}^{L} \phi_h(\mathbf{q}, \mathbf{t}), \tag{2.10}$$

ϕ_0 is a biased density which represents a nonuniform distribution of risk in the absence of targets, and therefore it can be interpreted as a priori distribution that quantifies the distribution of risk assigned to the area in terms of value of different regions. If ϕ in uniform in Ω, it means that every region has the same value. The distribution function, $\phi_h(\mathbf{q}, t)$, which represent the risk generated by the hth target is defined by the bell shaped function (non-normalized Gaussian)[15]

$$\phi_h(\mathbf{q}, t) = e^{-\frac{1}{2}\left(\frac{(q_x - \bar{q}_{h,x}(t))^2}{\beta_x^2} + \frac{(q_y - \bar{q}_{h,y}(t))^2}{\beta_y^2}\right)}, \tag{2.11}$$

$\bar{\mathbf{q}}(t) = [\bar{q}_x(t), \bar{q}_y(t)]^T$ is the position of a moving target and $\beta_x, \beta_y > 0$ represent the shape parameters which control the size of the risk function. The total density function, $\phi(\mathbf{q}, t)$ is derived by substituting (2.11) into (2.10)

$$\phi(\mathbf{q}, t) = \phi_0(\mathbf{q}) + \sum_{h=1}^{L} e^{-\frac{1}{2}\left(\frac{(q_x - \bar{q}_{h,x}(t))^2}{\beta_x^2} + \frac{(q_y - \bar{q}_{h,y}(t))^2}{\beta_y^2}\right)}, \tag{2.12}$$

Therefore, the non-autonomous coverage metric, $\mathcal{H}$ is defined by

$$\mathcal{H}(\mathbf{p}, \mathcal{V}, t) = \sum_{i=1}^{n} \mathcal{H}_i(\mathbf{p}_i, \mathcal{V}_i, t) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} f(r_i)\phi(\mathbf{q}, t) d\mathbf{q}, \tag{2.13}$$

Since the risk density (2.12) is time-varying and the control law (2.9) does not stabilize the multi-agent system to centroidal Voronoi tessellation, we aim to model the area coverage problem as a reinforcement learning problem, permitting continuous and automatic adaptation of the agents' strategies to the environment which changes due to the fact that the risk density field is in general time varying. Each agent estimates the risk filed ϕ by assuming that a target in the area modifies the risk as described in (2.12). This requires the knowledge of the states of the targets, that in this work are assumed to be known by the agents, but that in general need to be estimated in a real world scenario. The estimation can be done by coupling a Bayesian estimator such as Kalman filter, particle filter or any of the numerous variations and refinements, or by using reinforcement learning algorithms also to estimate the risk. In both cases, a full-state estimator can be embedded on each of the agents where individual agent uses local measurements and shared data to update position and velocity information of the target(s). Typically, the target moves through a sensor field where the total area of interest is assumed to be well covered by the sensors. The sensor data can be fused so as to generate target tracks and estimates by distributing fusion operations over multiple processing nodes. The sensor reflects the fact that a point is the responsibility of the sensor that has the best sensing of the point and since Voronoi techniques are used in obtaining regions based on points closest to the centroids, the agents are able to track and estimate the quantity of the target effectively. Reinforcement learning causes the agents to move towards their centroids so that their motion can be adaptive to the evolution of the environment. Also, reinforcement learning helps in the design of an energy efficient controller since limited battery power is a major challenge in choice of mobile sensor networks.

2.4 Reinforcement Learning and Optimal Control Theory

Reinforcement learning is an area of machine learning where an actor or agent interacts with its environment and modifies its actions(control policy) based on stimuli received in response to its actions. It is also called action-based learning and it tends to emulate human behavior. Some advantages of using reinforcement learning for control problems is that an agent can be retrained easily to adapt to environment changes, and trained continuously while the system is online, improving performance all the time. The use of samples to optimize performance and the use of function approximation to deal with large environments. Most optimal control approaches are offline and require complete knowledge of the system dynamics. Even for linear systems, where the linear-quadratic regulator gives the closed-form analytical solution to the optimal control problem, the continuous-time algebraic Riccati equation/discrete-time algebraic Riccati equation is solved offline and requires exact knowledge of the system dynamics. Adaptive control provides an inroad to design controllers which can adapt online to the uncertainties in system dynamics, based on minimization of the output error (e.g. using gradient or least squares methods).

A reinforcement learning control system is shown in Figure 2.3, where the controller, based on state feedback and reinforcement feedback about its previous action, calculates

the next control which should lead to an improved performance. The reinforcement signal is the output of a performance evaluator function, which is typically a function of the state and the control.

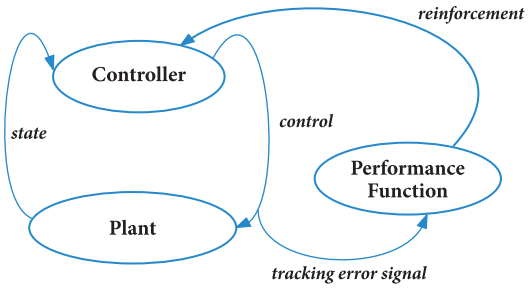


Figure 2.3: Reinforcement Learning Controller

2.4.1 Elements of reinforcement learning

This section discusses elements of reinforcement learning which includes four key components namely: (i) a control policy; (ii) a reward function; (iii) a value function; and (iv) a model of the environment. A control policy ($\mathbf{u}$) dictates the behaviour of the agent at a given time. In the progress of the area coverage mission, the policy constantly changes as the agent continually updates it until it achieves asymptotic convergence whereby the control policy approaches the optimal value of zero. This final value at convergence is called the optimal control policy ($\mathbf{u}^*$). In the context of the reinforcement learning framework, the variable $\mathbf{e}(t)$ represents the current system state $\mathbf{e}(t) \in \mathcal{S}$.

A reward function relays each perceived state of the environment to a scalar *reward or a reinforcement signal*. It is a means whereby the agent is rewarded for good behaviour and penalized for bad behaviour as dictated by the learning goal imposed on the agent. However, the objective of the agent is to exploit the total *reinforcement signal* associated with its task accomplishments. Whereas, a value function stipulates what is good behaviour for the agent in the long run. As a result, the *value* of a state is the sum total of reward an agent accumulates over the future, starting from its current state. Also, reinforcement learning is made possible through a conceptual model that mimics the behaviour of the environment, providing the agent with a means of deciding on a sequence of actions taking into account possible future situations before they are truly experienced.

2.4.2 Theoretical Framework: Markov Decision Process

The general framework of multi-agent systems, such as area coverage control problem is Markov Decision Processes. It provides the tools to model the multi-agent systems and provide the rational strategy to each agent to provide cooperative area coverage using optimal control theory. Area coverage control problems can be framed as generalized

Markov decision processes problem. In the next sections, the most relevant reinforcement learning concepts are discussed.

Markov decision processes provide a mathematical framework for modeling an agent's decision making process. In addition, it is a stochastic optimization problem where the goal is to find the optimal policy, a mapping between states and actions, which determines the agent's optimal actions at each state so as to maximize the discounted future reward, using a discount factor γ. Several reinforcement learning methods have been developed and successfully applied in machine learning to learn optimal policies for finite-state finite-action discrete-time Markov decision processes.

2.4.3 Solution Form: Optimal Control Theory

Adaptive optimal control refers to methods which learn the optimal solution online for uncertain systems. Reinforcement learning methods have been successfully used in Markov decision processes to learn optimal polices in uncertain environments. In [29] , it was shown that reinforcement learning is a direct adaptive optimal control technique. Owing to the discrete nature of reinforcement learning algorithms, many methods have been proposed for adaptive optimal control of discrete-time systems [21, 36, 39].

2.4.3.1 Linear Quadratic Regulator Control Problem

The discrete-time Linear quadratic regulator control problem is stated in this section as follows.

Given a controlled dynamical discrete-time system:$\mathbf{e}(k+1) = \mathbf{A}(k)\mathbf{e}(k) + \mathbf{B}(k)\mathbf{u}(k)$.

A running cost:$r\big(\mathbf{e}(k), \mathbf{u}(k)\big) = \mathbf{e}(k)^T\mathbf{Q}\mathbf{e}(k) + \mathbf{u}(k)^T\mathbf{R}\mathbf{u}(k)$

The problem is to find an optimal control policy that minimizes a cost function, $J(\mathbf{e}(k), \mathbf{u}(k))$.

Optimization Problem(OP)

$$(OP)\begin{cases} \textbf{Minimize} \quad J(\mathbf{e}(k),\mathbf{u}(k)) = \displaystyle\sum_{k=0}^{\infty} (\mathbf{e}(k)^T\mathbf{Q}\mathbf{e}(k) + \mathbf{u}(k)^T\mathbf{R}\mathbf{u}(k)) (\textit{running cost}) \\ \text{subject to} \quad \mathbf{u}(k) \in \mathcal{U}, \quad \mathbf{e}(k) \in \mathcal{X}, \quad k = 0, 1, \ldots \quad (\textit{state and control constraints}) \\ \mathbf{e}(k+1) = \mathbf{A}(k)\mathbf{e}(k) + \mathbf{B}(k)\mathbf{u}(k), \quad k = 0, 1, \ldots \quad (\textit{controlled dynamical model}) \\ \mathbf{e}(0) = \mathbf{e}_0 \quad (\textit{Initial State}) \end{cases}$$

where $\mathbf{e}(k) \in \chi \subseteq \mathbb{R}^n$, $\mathbf{u}(k) \in \mathcal{U} \subseteq \mathbb{R}^m$ are the system state and control input, respectively, χ and $\mathcal{U}$ are compact sets. Since the control law would be implemented on a digital computer, let $t = kT$, $k = 0, 1, \ldots$ and T is the sampling time. $\mathbf{A}$ represents the plant matrix while $\mathbf{B}$ represents the gain matrix. Matrices $\mathbf{Q}$ and $\mathbf{R}$ are strictly positive definite cost matrices.

2.4.3.2 Optimality and Approximation: The Hamilton-Jacobi-Bellman Equation

When solving linear quadratic regulator for linear systems, the online solution to the algebraic Riccati equation is often the preferred approach. Whereas, theoretically the value function approximation using neural network is an indirect solution method, whereby in the case of the infinite-horizon optimal control policy for a nonlinear system is obtained by solving the Hamilton Jacobi Bellman equation, which is essentially an algebraic equation.

When considering the case of the discrete-time linear quadratic regulator, the Hamilton Jacobi Bellman equation further reduces to the discrete time algebraic Riccati equation (DARE). However, for most cases, it is impossible to solve the Hamilton Jacobi Bellman equation since it is generally a nonlinear partial differential equation. Therefore, a value function approximation approach using neural network, is utilized to find the approximated solution to the Hamilton Jacobi Bellman equation, reduces to the DARE where the neural network weights are trained using samples from the recursive least-squares solver.

2.4.4 Dynamic Programming (DP) Methods

Dynamic programming was developed by R. E. Bellman in the later 1950s [40]. It can be used to solve control problems for nonlinear, time-varying systems, and it is straightforward to program. It is a method for determining optimal control solutions using Bellmans principle by working backward in time from some desired goal states. Designs based on dynamic programming yield offline solution algorithms, which are then stored and implemented online forward in time [28] which is unsuitable for real-time applications. DP algorithms are model-based and they include the policy evaluation step used for computing the value functions and the policy improvement step used in calculating an improved policy.

2.4.5 Temporal-Difference (TD) Methods

The temporal difference (TD) method [27, 29] for solving Bellman equation (2.14) leads to a family of optimal adaptive controllers capable of learning online the solutions to optimal control problems without knowing the full system dynamics. TD learning is true online reinforcement learning, wherein control actions are improved in real time based on estimating their value functions by observing data measured along the system trajectories [28].

$$V^{\pi}(\mathbf{e}) = \sum_{u} \pi(e,u) \sum_{e'} P^{u}_{ee'}[R^{u}_{ee'} + \gamma V^{\pi}(\mathbf{e}')] \tag{2.14}$$

Equation (2.14) is the bellman equation [28] which forms the basis for temporal difference learning. $V^{\pi}(\mathbf{e})$ is considered as the predicted performance, $\Sigma_u \pi(e,u) \Sigma_{e'} P^{u}_{ee'} R^{u}_{ee'}$ is the observed one-step reward and $V^{\pi}(\mathbf{e}')$ is the current estimate of future behaviour. [34] identified two key advantages of TD methods over conventional prediction-learning methods: (i) they are more incremental and therefore easier to compute; (ii) they tend to make more efficient use of their experience since they converge faster and produce better predictions.

2.4.5.1 Actor-Critic Methods

Actor-critic methods, introduced by [29], implement the policy iteration algorithm online, where the critic is typically a neural network which implements policy evaluation and approximates the value function, whereas the actor is another neural network which approximates the control. The critic evaluates the performance of the actor using a scalar reward from the environment and generates a temporal difference error. The actor-critic neural networks, shown in Figure 2.4 are updated using recursive least-squares update laws based on the TD error.

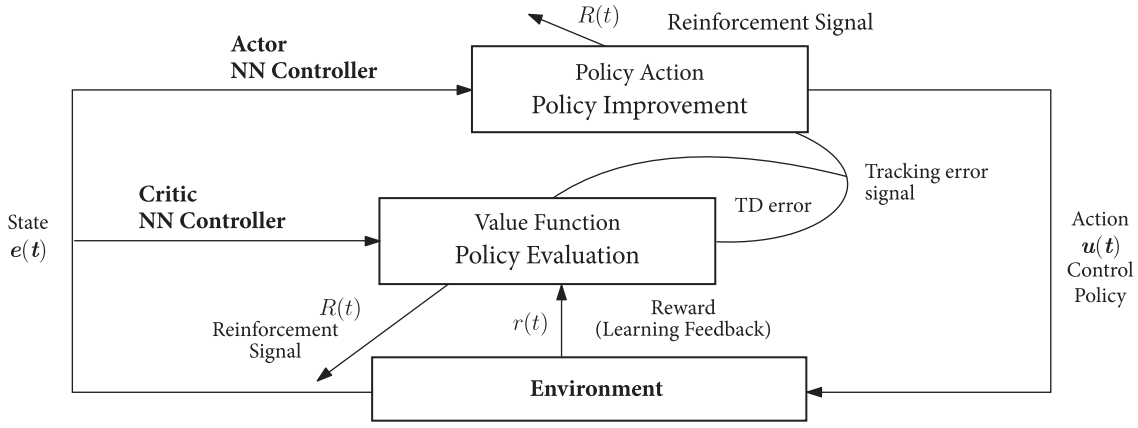


Figure 2.4: Actor-Critic NN Architecture Structure

Policy-Iteration based schemes (actor-critic learning): In the policy evaluation block, the critic neural network Controller essentially computes the value function under the current policy (assuming a fixed, stationary policy). In the policy action block, the actor neural network controller essentially performs some form of policy improvement, based on the policy iterations and is responsible for implementing some exploration process.

In terms of nonuniform area coverage control problem, the critic NN controller encodes the expected future reward $r(t)$ (used for policy evaluation of the value function) at the agent's current position. The change in the predicted value is compared to the actual reward, leading to the Temporal Difference (TD) error. The TD error signal is broadcast to the actor NN controller as part of the learning rule. The actor NN controller is responsible for the direction taken by the mobile agent which is governed by the control law, $\mathbf{u}(t)$.

Actor-critic methods are becoming the choice for implementing reinforcement learning because of two significant advantages: (i) they require minimal computation in order to select actions; (ii) they can learn an explicitly stochastic policy; that is, they can learn the optimal probabilities of selecting various actions rendering Markov decision process computations more efficient.

2.5 Environment Monitoring and Cooperative Robotic Teams

Multi-robot systems, also called robotic teams, are often used as framework for area coverage control problems. When coupled with multi-robot systems, reinforcement learning is used to acquire a wide spectrum of skills, ranging from basic behaviours like navigation to complex behaviours like area coverage, surveillance and environment monitoring.

2.6 Summary and Discussion

In this chapter we presented the tools that are needed to represent and solve decentralized coordination problems with multi-agent systems. An overview of several relevant optimal control-theoretical concepts that allow us to determine the outcomes of strategic interactions between agents were presented. Likewise, a distributed cooperative grouping behaviour of agents facilitated by the unique geometric properties of centroidal Voronoi tessellation was studied. Furthermore, reinforcement learning framework that helps individual agents coordinate their actions in a distributed and self-organizing way was discussed. The theory of Markov decision processes allows us to examine the global behaviour of the multi-agent system and study the convergence properties of learning algorithms. In the next chapter, the coverage control problem is formulated as a nonlinear derivative error function which serves as an optimal control problem. Also, the energy consumption of the agents during the coverage task is defined as an optimal control problem.

Chapter 3

Optimal control system formulation

3.1 Introduction

Chapter 3 provides a main contribution in this thesis with the formulation of the area coverage control problem stated in Chapter 1 as a linear quadratic regulator problem. This formulation makes it possible henceforth to solve the area coverage control problem as an optimal control problem using reinforcement learning techniques. Section 3.2 gives a mathematical formulation of nonlinear derivative error function from the coverage control optimization problem. It is formulated as a linear quadratic regulator problem where the system dynamics are linear and the cost is quadratic. In section 3.3, an optimal control approach is used in modeling an energy efficient area coverage control problem.

3.2 Formulation of nonlinear derivative error function

The first order dynamics of the ith agent under the proposed coverage control problem can be described by

$$\dot{\mathbf{p}}_i = \mathbf{u}_i \tag{3.1}$$

Given a time-varying density function $\phi(\mathbf{q}, t)$ as defined in (2.12) and a non-autonomous coverage metric,$\mathcal{H}(\mathbf{p}, \mathcal{V}, t)$ defined in (2.13). We define a time varying feedback plus feed-forward control law according to [15] as

$$\dot{\mathbf{p}}_i = \dot{\mathbf{c}}_{\mathcal{V}_i} + \left(K_p + \frac{\dot{m}_{\mathcal{V}_i}}{m_{\mathcal{V}_i}}\right)(\mathbf{c}_{\mathcal{V}_i} - \mathbf{p}_i). \tag{3.2}$$

where K_p is a positive gain, $m_{\mathcal{V}_i}$ and $\mathbf{c}_{\mathcal{V}_i}$ are the mass and centroid of a Voronoi cell of the ith agent respectively, $\dot{\mathbf{p}}_i$ represents the velocity of the ith agent. The time derivative of the mass and centroid is derived in [15] as

$$\dot{m}_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \dot{\phi}(\mathbf{q}, t) d\mathbf{q}, \quad \dot{\mathbf{c}}_{\mathcal{V}_i} = \frac{1}{m_{\mathcal{V}_i}} \left(\int_{\mathcal{V}_i} \mathbf{q}\dot{\phi}(\mathbf{q}, t) d\mathbf{q} - \dot{m}_{\mathcal{V}_i} \mathbf{c}_{\mathcal{V}_i} \right),$$

The positional error is defined as

$$\mathbf{e}_i = \mathbf{c}_{\mathcal{V}_i} - \mathbf{p}_i \tag{3.3}$$

Differentiating the centroid w.r.t time, we get

$$\dot{\mathbf{c}}_{\mathcal{V}_i} = \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\dot{\mathbf{p}}_i + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} \tag{3.4}$$

The derivative of the error function can now be written as

$$\dot{\mathbf{e}}_i = \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\dot{\mathbf{p}}_i + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} - \dot{\mathbf{p}}_i = \left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\dot{\mathbf{p}}_i + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} \tag{3.5}$$

Substituting (3.2) into (3.5), we get

$$\dot{\mathbf{e}}_i = \left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\left[\dot{\mathbf{c}}_{\mathcal{V}_i} + \left(K_p + \frac{\dot{m}_{\mathcal{V}_i}}{m_{\mathcal{V}_i}}\right)\mathbf{e}_i\right] + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} \tag{3.6}$$

Differentiating (3.3) w.r.t time, we get $\dot{\mathbf{e}}_i = \dot{\mathbf{c}}_{\mathcal{V}_i} - \dot{\mathbf{p}}_i$ and using the fact that $\dot{\mathbf{p}}_i = \mathbf{u}_i$, it then follows that $\dot{\mathbf{e}}_i = \dot{\mathbf{c}}_{\mathcal{V}_i} - \mathbf{u}_i$, or precisely

$$\dot{\mathbf{c}}_{\mathcal{V}_i} = \dot{\mathbf{e}}_i + \mathbf{u}_i \tag{3.7}$$

Substituting (3.7) into (3.6) we get

$$\dot{\mathbf{e}}_i = \left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\left[(\dot{\mathbf{e}}_i + \mathbf{u}_i) + \left(K_p + \frac{\dot{m}_{\mathcal{V}_i}}{m_{\mathcal{V}_i}}\right)\mathbf{e}_i\right] + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} \tag{3.8}$$

Expanding (3.8) and rearranging the terms we get

$$\left(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\dot{\mathbf{e}}_i = \left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\left(K_p + \frac{\dot{m}_{\mathcal{V}_i}}{m_{\mathcal{V}_i}}\right)\mathbf{e}_i + \left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\mathbf{u}_i + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} \tag{3.9}$$

The first-order nonlinear error dynamics is obtained from equation (3.9) as

$$\begin{aligned}\dot{\mathbf{e}}_i = \eta(\mathbf{e}_i, \mathbf{u}_i) = &\left[\left(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)^{-1}\left\{\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\left(K_p + \frac{\dot{m}_{\mathcal{V}_i}}{m_{\mathcal{V}_i}}\right)\right\}\right]\mathbf{e}_i + \\ &\left[\left(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)^{-1}\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\right]\mathbf{u}_i + \left(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)^{-1}\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}\end{aligned} \tag{3.10}$$

Similar to the framework in [2, 41], if λ_{max} denotes the eigenvalue with the largest magnitude of the matrix $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}$ and by using Neumann series, we can express $(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i})^{-1}$ as

$$(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i})^{-1} = 2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} + (\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i})^2 + \dots \tag{3.11}$$

where $|\lambda_{max}| < 0$. By letting $\dot{\mathbf{p}}_i$ depend entirely on $\mathbf{p}_j$, $j \in N_{\mathcal{V}_i}$, (as well as $\mathbf{p}_i$ itself), the series can be truncated after the first two entries

$$\left(2\mathbf{I}_2 - \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)^{-1} \approx \left(2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)$$

The first-order nonlinear derivative error function can now be rewritten in the form:

$$\begin{aligned}\dot{\mathbf{e}}_i = \eta(\mathbf{e}_i, \mathbf{u}_i) = \left[\left(2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\left(K_p + \frac{\dot{m_{\mathcal{V}_i}}}{m_{\mathcal{V}_i}}\right)\right]\mathbf{e}_i + \\ \left[\left(2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\right]\mathbf{u}_i + \left(2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}\end{aligned} \tag{3.12}$$

By linearizing (3.12) around optimal equilibrium points $(0, 0)$, then the corresponding linear model which defines the system's dynamic behaviour about the optimal equilibrium operating point can be represented as:

$$\dot{\mathbf{e}}_i(t) = \mathbf{A}_i(t)\mathbf{e}_i(t) + \mathbf{B}_i(t)\mathbf{u}_i(t) \tag{3.13}$$

where matrices, $\mathbf{A}_i$ and $\mathbf{B}_i$ are denoted by the Jacobian matrices of $\eta(\mathbf{e}_i, \mathbf{u}_i)$ with respect to $\mathbf{e}_i$ and $\mathbf{u}_i$ respectively as follows:

$$\mathbf{A}_i = \left[\left(2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\left(K_p + \frac{\dot{m}_{\mathcal{V}_i}}{m_{\mathcal{V}_i}}\right)\right]. \tag{3.14}$$

$$\mathbf{B}_i = \left[\left(2\mathbf{I}_2 + \frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}\right)\left(\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} - \mathbf{I}_2\right)\right]. \tag{3.15}$$

where $\mathbf{I}_2$ is a 2×2 Identity Matrix and the formulas used in the computation of the partial derivatives $\frac{\partial m_{\mathcal{V}_i}}{\partial t}$, $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}$ and $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}$ are defined next. The partial derivative of the mass of the ith agent with respect to time is defined as follows:

$$\frac{\partial m_{\mathcal{V}_i}}{\partial t} = \int_{\mathcal{V}_i} \dot{\phi}(\mathbf{q}, t) d\mathbf{q}, \tag{3.16}$$

where $\dot{\phi}$ is the rate of change of risk density with time, it can be computed as follows:

$$\dot{\phi}(\mathbf{q}, t) = \left(\dot{\overline{q}}_x \cdot \frac{(q_x - \overline{q}_x(t))^2}{\beta_x^2} + \dot{\overline{q}}_y \cdot \frac{(q_y - \overline{q}_y(t))^2}{\beta_y^2}\right) e^{-\frac{1}{2}\left(\frac{(q_x - \overline{q}_x(t))^2}{\beta_x^2} + \frac{(q_y - \overline{q}_y(t))^2}{\beta_y^2}\right)}$$

where $\dot{\overline{q}}_x$ and $\dot{\overline{q}}_y$ are the velocities of the targets in x and y directions respectively with the other parameters described in (2.11). The partial derivative of the centroid of the ith agent with respect to time and the partial derivative of its centroid with respect to position are given by [1]

$$\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t} = \frac{m_{\mathcal{V}_i} \int \mathbf{q}\dot{\phi}(\mathbf{q}, t) d\mathbf{q} - \frac{\partial m_{\mathcal{V}_i}}{\partial t} \int \mathbf{q}\phi(\mathbf{q}, t) d\mathbf{q}}{m_{\mathcal{V}_i}^2}. \tag{3.17}$$

$$\frac{\partial \mathbf{c}_{\mathcal{V}_i}^{(a)}}{\partial \mathbf{p}_i^{(b)}} = \sum_{j \in N\mathcal{V}_i} \left[\left(\int \phi(\mathbf{q}) q^{(a)} \frac{q^{(b)} - p_i^{(b)}}{\|p_j - p_i\|} d\mathbf{q} \right) / m_{\mathcal{V}_i} - \right. \\ \left. - \left(\int \phi(\mathbf{q}) \frac{q^{(b)} - p_i^{(b)}}{\|p_j - p_i\|} d\mathbf{q} \right) \left(\int \phi(\mathbf{q}) q^{(a)} d\mathbf{q} \right) / m_{\mathcal{V}_i}^2 \right]. \tag{3.18}$$

where $i \neq j$, $N_{\mathcal{V}_i}$ denotes the set of Voronoi neighbours of agent i with $j \in N_{\mathcal{V}_i}$. $\partial \mathbf{c}_{\mathcal{V}_i} / \partial \mathbf{p}_i$ is a block matrix. Note that (a, b) $\in$ (1, 2) since we are considering the case $\Omega \subset \mathbb{R}^2$ only.

$$\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i} = \begin{bmatrix} \frac{\partial \mathbf{c}_{\mathcal{V}_i}^{(a)}}{\partial \mathbf{p}_i^{(a)}} & \frac{\partial \mathbf{c}_{\mathcal{V}_i}^{(a)}}{\partial \mathbf{p}_i^{(b)}} \\ \frac{\partial \mathbf{c}_{\mathcal{V}_i}^{(b)}}{\partial \mathbf{p}_i^{(a)}} & \frac{\partial \mathbf{c}_{\mathcal{V}_i}^{(b)}}{\partial \mathbf{p}_i^{(b)}} \end{bmatrix}. \tag{3.19}$$

3.3 Optimal Control of Distributed Energy-Efficient Area Coverage Problem

In this section,we present a distributed optimal control approach for modeling energy-efficient cooperative area coverage control problem where the energy consumption is described as an optimization problem. Moarref and Rodrigues [42] showed that by tuning the ratio of the weight on the coverage criterion (s_i) to the weight on energy consumption (z_i), we can achieve a locally optimal coverage using less energy. However, our formulation of the energy-efficient feedback controller is correlated to Lloyd's algorithm in equation (2.9) where the proportionality constant is expressed as a function of the ratio s_i/z_i.

$$\mathbf{u}_i = \sqrt{s_i/z_i}(\mathbf{c}_{\mathcal{V}_i} - \mathbf{p}_i) = \sqrt{s_i/z_i}\mathbf{e}_i \tag{3.20}$$

Subject to $\dot{\mathbf{p}}_i = \mathbf{u}_i$, with $s_i > 0$,$z_i > 0$, $i \in \{1, \ldots, n\}$, where $K_p = \sqrt{s_i/z_i}$. Moreover, if $s_i z_i = 1$, $\forall i \in \{1, \ldots, n\}$, the multi-agent system converges to a centroidal Voronoi configuration [42].

The energy consumption (E_i) for each agent is defined by:

$$E_i(\mathbf{u}_i(t)) = \int_0^{T_f} \|\mathbf{u}_i(t)\|^2 dt \tag{3.21}$$

The energy consumption constraint requires that the sum of energy consumed by each agent cannot exceed the total energy consumption for the entire mobile actuators/sensors network. The non-negativity constraint also requires that all the agents in the network have sufficient energy to guarantee completion of the coverage mission. In other words, agents do not run out of energy during the deployment. Therefore, by adjusting the ratio of the weight on the coverage criterion to the weight on energy consumption, we can achieve a locally optimal coverage using less energy.

3.4 Summary and Discussion

In this Chapter, to answer the research question **Q4**, the derivative of the error function is formulated. The single-integrator (first-order) dynamics formulation is developed as a mathematical system model. As a result, this formulation is a major contribution in this thesis and to the field of area coverage control since it makes it possible to henceforth be able to study how individual agents, in a multi-agent system, can coordinate with each other using an explicit mathematical systems model. The methodology proposed for this thesis clearly shows two distinct solution paths whereby reinforcement learning is used in solving the area coverage control problem which include (i) an online policy iteration solution approach for solving the continuous-time algebraic Riccati equation for multi-agent linear systems and (ii) an online actor-critic reinforcement solution framework for solving the discrete-time algebraic Riccati equation for multi-agent linear systems both presented in Chapter 4.

Chapter 4

Optimal Control Solution Approach for Non-autonomous coverage

Chapter 4 presents the two proposed solution approaches that is used to solve the area coverage control problem formulated in the Chapter 3. The online policy iteration is presented in section 4.1 and the actor-critic solution approach is presented in section 4.3.

4.1 Riccati Equation Online Solution for Uncertain Linear Systems

In this section, we will be solving the optimal control problem formulated in chapter 3 using the online policy iteration solution approach proposed in [4] which uses a novel computational adaptive optimal control methodology that employs approximate dynamic programming technique to iteratively solve the algebraic Riccati equation online completely without any knowledge of the system dynamics matrices. This methodology is different from most approaches in literature since it outlines a computational adaptive optimal control design that solves the continuous-time algebraic Riccati equation with completely unknown internal system dynamics (i.e., the plant matrix and the control-input coupling matrix are not needed for the online solution).

This work done in this section shows that the computational adaptive control algorithm in [4] can be adapted to solve the area coverage problem formulated as a continuous time linear system in equation (3.13) and simulation results shows that the agents converge to to their respective centroids and the defined coverage metric (2.13) is maximized.

The chapter is organized as follows. In Section 4.1.1, a brief problem formulation is presented introducing the standard linear optimal control problem for continuous-time systems and the policy iteration technique. In Section 4.1.2, a practical online tuning adaptive optimal control solution is provided. In Section 4.1.3, the proposed multi-agent area coverage control using policy iteration technique algorithm is presented. In section 4.2, the effectiveness of the proposed algorithm is verified using MATLAB simulations and analyzed numerically.

4.1.1 Problem Formulation

Given a continuous-time linear dynamical system defined by equation (4.1),

$$\dot{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{e} + \boldsymbol{B}\boldsymbol{u}. \tag{4.1}$$

where $\boldsymbol{e}(t) \in \mathbb{R}^n$ is the system state fully available for feedback control design; $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the control input, and the positive integers n and m, respectively, denote the dimensions of the state and the control vectors. If the continuous cost function is given by

$$\boldsymbol{J} = \int_0^\infty \Big(\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\Big)dt, \tag{4.2}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^T \geq 0$, $\boldsymbol{R} = \boldsymbol{R}^T > 0$ with $(\boldsymbol{A}, \boldsymbol{Q}^{1/2})$ observable. Additionally, positive semi-definite smooth function $\boldsymbol{Q}$:$\mathbb{R}^n \to \mathbb{R}_+$ penalizes the states, the positive definite matrix $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ penalizes the control effort, the plant matrix $\boldsymbol{A}$: $\mathbb{R}^n \to \mathbb{R}^n$ represents the internal dynamics of the system, and the input coupling matrix $\boldsymbol{B}$: $\mathbb{R}^n \to \mathbb{R}^{n\times m}$ is the matrix-valued input gain matrix. The design objective is to find a linear optimal control law defined by (4.3) which minimizes the continuous cost function given by equation (4.2).

$$\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{e}, \tag{4.3}$$

where $\boldsymbol{K}$ is a feedback gain matrix and $\boldsymbol{e}(t)$ is the positional error defining the motion of an agent's trajectory. By linear optimal control theory, when both $\mathbf{A}$ and $\mathbf{B}$ are accurately known, the solution to this problem can be found by solving the following well-known algebraic Riccati equation [4]

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = 0 \tag{4.4}$$

Equation (4.4) has a unique symmetric positive definite solution $\boldsymbol{P}^*$. The optimal feedback gain matrix $\boldsymbol{K}^*$ in (4.3) can be determined by

$$\boldsymbol{K}^* = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}^*. \tag{4.5}$$

It is usually difficult to directly solve $\boldsymbol{P}^*$ since equation (4.4) is nonlinear in $\boldsymbol{P}$. However, many efficient algorithms have been developed to numerically approximate the solution of equation (4.4) [4]. One of such algorithms was developed in [43], and is introduced in:

Theorem 4.1. [43] *Let $\boldsymbol{K}_0 \in \mathbb{R}^{m\times n}$ be any stabilizing feedback gain matrix, and let $\boldsymbol{P}_k$ be the symmetric positive definite solution of the Lyapunov equation*

$$(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_k)^T\boldsymbol{P}_k + \boldsymbol{P}_k(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_k) + \boldsymbol{Q} + \boldsymbol{K}_k^T\boldsymbol{R}\boldsymbol{K}_k = 0 \tag{4.6}$$

where $\boldsymbol{K}_k$, with $k = 1, 2, \ldots$, are defined recursively by:

$$\boldsymbol{K}_k = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_{k-1}. \tag{4.7}$$

Then, the following properties hold:

1. *$\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_k$ is Hurwitz,*
2. *$\boldsymbol{P}^* \leq \boldsymbol{P}_{k+1} \leq \boldsymbol{P}_k$,*
3. *$\lim_{k\to\infty} \boldsymbol{K}_k = \boldsymbol{K}^*$, $\lim_{k\to\infty} \boldsymbol{P}_k = \boldsymbol{P}^*$.*

In [43],it is shown that the sequence $\mathbf{P}_k$ is always decreasing and at every iteration k the matrix $\mathbf{P}_k$ is positive definite. Likewise, $\mathbf{P}_k - \mathbf{P}_{k-1}$ is always positive definite.

4.1.2 Continuous-Time Algebraic Riccati Equation (CARE) Online Implementation

In this section, the online policy iteration strategy that will be used in solving the area coverage control problem formulated in Chapter 3 is presented as described in [4]. The online tuning algorithm is implemented by solving for the cost matrix ($\boldsymbol{P}$) and the feedback gain matrix ($\boldsymbol{K}$) by least-squares (LS) technique. The control policy is updated after convergence using equation (4.3).

CT Adaptive Finite-Horizon Optimal Control Solution for Linear Systems

This section discusses the online learning strategy that recursively calculates the finite horizon optimal cost, and solves the finite horizon linear quadratic regulator problem. The algorithm is proposed in five steps:

Step 1: The algorithm is initialized assuming a stabilizing $\boldsymbol{K}_0$ is known. Then, for each $k \in \mathbb{Z}_+$, we seek to solve a symmetric positive definite matrix $\boldsymbol{P}_k$ satisfying (4.6), and obtain a feedback gain matrix $\boldsymbol{K}_{k+1} \in \mathbb{R}^{m\times n}$ using $\boldsymbol{K}_{k+1} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_k$ [4].

The CT linear system can be written as

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_k e + \boldsymbol{B}(\boldsymbol{K}_k\boldsymbol{e} + \boldsymbol{u}) \tag{4.8}$$

where $\boldsymbol{A}_k = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_k$.

Solving (4.8) using (4.6) and (4.7) it follows that

$$\begin{aligned} &\boldsymbol{e}(t+\delta t)^T\boldsymbol{P}_k\boldsymbol{e}(t+\delta t) - \boldsymbol{e}^T(t)\boldsymbol{P}_k\boldsymbol{e}(t) = \\ &= \int_t^{t+\delta t}\Big[\boldsymbol{e}(\tau)^T(\boldsymbol{A}_k^T\boldsymbol{P}_k + \boldsymbol{P}_k\boldsymbol{A}_k)\boldsymbol{e}(\tau) + 2(\boldsymbol{u}+\boldsymbol{K}_k\boldsymbol{e}(\tau))^T\boldsymbol{B}^T\boldsymbol{P}_k\boldsymbol{e}(\tau)\Big]d\tau = \\ &= -\int_t^{t+\delta t}\boldsymbol{e}(\tau)^T\boldsymbol{Q}_k\boldsymbol{e}(\tau)d\tau + 2\int_t^{t+\delta t}\big(\boldsymbol{u}+\boldsymbol{K}_k\boldsymbol{e}(\tau)\big)^T\boldsymbol{R}\boldsymbol{K}_{k+1}\boldsymbol{e}(\tau)d\tau \end{aligned} \tag{4.9}$$

where $\boldsymbol{Q}_k = \boldsymbol{Q} + \boldsymbol{K}_k^T\boldsymbol{R}\boldsymbol{K}_k$. Note that in equation (4.9), the term $\boldsymbol{e}(\tau)^T(\boldsymbol{A}_k^T\boldsymbol{P}_k + \boldsymbol{P}_k\boldsymbol{A}_k)\boldsymbol{e}(\tau)$ which depends on the matrix $\mathbf{A}$ is replaced by $\boldsymbol{e}(\tau)^T\boldsymbol{Q}_k\boldsymbol{e}(\tau)$. Likewise the term $\boldsymbol{B}^T\boldsymbol{P}_k$ involving $\mathbf{B}$ is replaced by $\boldsymbol{R}\boldsymbol{K}_{k+1}$. This eliminates the need for the system matrices $\mathbf{A}$ and $\mathbf{B}$. This makes the tuning algorithm completely model free.

Step 2: Given a stabilizing $\boldsymbol{K}_k$, a pair of matrices ($\boldsymbol{P}_k$, $\boldsymbol{K}_{k+1}$), with $\boldsymbol{P}_k = \boldsymbol{P}_k^T > 0$, satisfying (4.6) and (4.7) can be uniquely determined without knowing A or B, under certain condition as proposed by [4]. To obtain the least-squares (LS) solution, we define the following two operators:

$$\boldsymbol{P} \in \mathbb{R}^{n\times n} \to \hat{\boldsymbol{P}} \in \mathbb{R}^{\frac{1}{2}n(n+1)} \quad \text{and} \quad \boldsymbol{e} \in \mathbb{R}^n \to \overline{\boldsymbol{e}} \in \mathbb{R}^{\frac{1}{2}n(n+1)}$$

where

$$\begin{aligned} \hat{\boldsymbol{P}} &= \Big[p_{11},\ 2p_{12},\ldots,2p_{1n},\ p_{22},\ 2p_{23},\ldots 2p_{n-1,n},\ p_{nn}\Big]^T \\ \overline{\boldsymbol{e}} &= \Big[e_1^2,\ e_1e_2,\ldots,e_1e_n,\ e_2^2,\ e_2e_3,\ldots,e_{n-1}e_n,\ e_n^2\Big]^T \end{aligned}$$

In addition, by Kronecker product representation, we have

$$\boldsymbol{e}^T\boldsymbol{Q}_k\boldsymbol{e} = (\boldsymbol{e}^T \otimes \boldsymbol{e}^T)\,\mathrm{vec}(\boldsymbol{Q}_k)$$

and

$$(\boldsymbol{u} + \boldsymbol{K}_k\boldsymbol{e})^T\boldsymbol{R}\boldsymbol{K}_{k+1}\boldsymbol{e} = \Big[(\boldsymbol{e}^T \otimes \boldsymbol{e}^T)(I_n \otimes \boldsymbol{K}_k^T\boldsymbol{R}) + (\boldsymbol{e}^T \otimes \boldsymbol{u}^T)(I_n \otimes \boldsymbol{R})\Big]\,\mathrm{vec}(\boldsymbol{K}_{k+1})$$

where $\otimes$ indicates the kronecker delta and $\mathrm{vec}(\boldsymbol{Q}_k)$ is defined to be the $n \times n$-vector formed by stacking the columns of $(\boldsymbol{Q}_k) \in \mathbb{R}^{n\times n}$ on top of another.

Step 3: For positive integer l, we define the computational matrices $\delta_{e_ie_i} \in \mathbb{R}^{l\times\frac{1}{2}n(n+1)}$, $I_{e_ie_i} \in \mathbb{R}^{l\times n^2}$, $I_{\boldsymbol{e_iu_i}} \in \mathbb{R}^{l\times mn}$, such that

$$\begin{aligned}
\delta_{\boldsymbol{ee}} &= \Big[\overline{e}(t_1) - \overline{e}(t_0),\ \overline{e}(t_2) - \overline{e}(t_1), \ldots, \overline{e}(t_l) - \overline{e}(t_{l-1})\Big]^T \\
I_{\boldsymbol{ee}} &= \left[\int_{t_0}^{t_1}(e\otimes e)d\tau,\ \int_{t_1}^{t_2}(e\otimes e)d\tau, \ldots, \int_{t_{l-1}}^{t_l}(e\otimes e)d\tau\right]^T \\
I_{\boldsymbol{eu}} &= \left[\int_{t_0}^{t_1}(e\otimes u)d\tau,\ \int_{t_1}^{t_2}(e\otimes u)d\tau, \ldots, \int_{t_{l-1}}^{t_l}(e\otimes u)d\tau\right]^T
\end{aligned}$$

where $0 \le t_0 < t_1 < \cdots < t_l$.

Step 4: Also, we define the least-squares (LS) matrices $\Theta \in \mathbb{R}^{l\times\left[\frac{1}{2}n(n+1)+mn\right]}$ and $\Xi \in \mathbb{R}^l$ as:

$$\Theta_k = \Big[\delta_{\boldsymbol{ee}},\ -2I_{\boldsymbol{ee}}(I_n \otimes \boldsymbol{K}_k^T\boldsymbol{R}) - 2I_{\boldsymbol{eu}}(I_n \otimes \boldsymbol{R})\Big],$$

and

$$\Xi_k = -I_{\boldsymbol{ee}}\,\mathrm{vec}(\boldsymbol{Q}_k).$$

Step 5: For any given stabilizing gain matrix $\boldsymbol{K}_k$, (4.9) implies the least-squares (LS) matrices Θ_k and Ξ_k can be expressed by the following matrix form of linear equation as:

$$\Theta_k \begin{bmatrix} \widehat{\boldsymbol{P}}_k \\ \mathrm{vec}(\boldsymbol{K}_{k+1}) \end{bmatrix} = \Xi_k \tag{4.10}$$

Notice that if Θ_k has full column rank, the least-squares solution that gives the optimal gain matrix $\boldsymbol{K}$ can be directly obtained as follows:

$$\begin{bmatrix} \widehat{\boldsymbol{P}}_k \\ \mathrm{vec}(\boldsymbol{K_{k+1}}) \end{bmatrix} = \Big[(\Theta_k)^T(\Theta_k)\Big]^{-1}(\Theta_k)^T\Xi_k \tag{4.11}$$

The computational adaptive optimal control algorithm that solves the linear quadratic regulator problem formulated in Chapter 3 and the multi-agent area coverage control using policy iteration algorithm that solves the area coverage problem and move the agents to optimal configuration are presented in the next section.

4.1.3 Optimal Control Algorithm for multi-agent area coverage control

In this section, the online tuning algorithm proposed by [4], that solves equation (4.11) for the optimal matrices $\boldsymbol{P}^*$ and $\boldsymbol{K}^*$ using recursive least square is presented in Algorithm 1 while the coverage control algorithm that achieves centroidal Voronoi configuration of the agents and maximizes the coverage metric is described in Algorithm 2. The least-squares solution in Algorithm 1 requires at least $N = n(n+1)/2$ points, which is the number of independent elements in the matrix $\boldsymbol{P}$.

Algorithm 1:(Computational adaptive Optimal Control Algorithm)

Step 1: Initialization: Let $k = 0$ and K_0 is stabilizing; for the agent i employ $\boldsymbol{u}_i = -\boldsymbol{K}_0\boldsymbol{u}_i + \varepsilon$, $t \in [t_0,\, t_l]$, as the input on the time interval $[t_0, t_l]$, where $\boldsymbol{K}_0$ is stabilizing and ε is the exploration noise. Compute matrices $\boldsymbol{\delta_{ee}}$, $\boldsymbol{I_{ee}}$ and $\boldsymbol{I_{eu}}$.

Step 2: Solve $\hat{\boldsymbol{P}}_k$ and $\boldsymbol{K}_k$ from (4.11).

Step 3: Let $k \leftarrow k+1$, and repeat **Step 2** until $\|\boldsymbol{P}_k - \boldsymbol{P}_k| \leq \epsilon$ for $k \geq 1$, where the constant $\epsilon > 0$ is a predefined small threshold.

Step 4: Use $\boldsymbol{u_i^*} = -\boldsymbol{K}_k\boldsymbol{e}_i$ as the approximated optimal control input policy.

Algorithm 2:(MAACC-RL PI Algorithm)

The proposed MAACC-RL PI that converges the agents to their respective centroidal Voronoi configuration is presented below:
Step 1: Initialize all parameters such as the initial time and agent's initial position and define the bounds for the workspace.

Step 2: Measure the Risk density ϕ and compute the Voronoi region $\mathcal{V}_i$ obtaining the $m_{\mathcal{V}_i}$, $\mathbf{c}_{\mathcal{V}_i}$, $\dfrac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}$, $\dfrac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}$ and $\dfrac{\partial m_{\mathcal{V}_i}}{\partial t}$.

Step 3: Update the feedback law using the optimal control policy obtained from **Algorithm 1** above.

Step 4: The ith agent's new position is updated using the kinematic model (3.1)

Step 5: If agent $\mathbf{p}_i$ converges to its centroid $\mathbf{c}_i$, **stop procedure**, else go to **Step 2**

4.2 Simulation Results for Policy Iteration Solution Approach

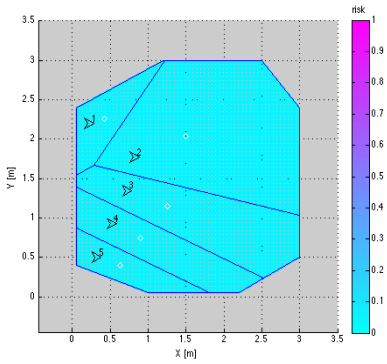


Figure 4.1: Initial configuration of all agents

The effectiveness of the MAACC-RL PI algorithm which uses the online tuning framework described in [4] and the correctness of relevant theories presented are verified in this section. Also, it demonstrates the applicability of the developed iterative technique in solving area coverage problems. As an application, a harbour protection scenario was considered and the proposed approach is implemented using two scenarios which are presented in Sections 4.2.1 and 4.2.2 for tracking single and multiple targets respectively while providing area coverage. We define the harbour as a convex polygonal region with vertices (1.0, 0.05), (2.2, 0.05), (3.0, 0.5), (3.0, 2.4), (2.5, 3.0), (1.2, 3.0), (0.05, 2.4) and (0.05, 0.4). The prior risk in the harbour defined in equation (2.10), $\phi_0 = 5 \times 10^{-2}, \forall \mathbf{q} \in \Omega$. The density function parameters $\beta_x = 0.4$ and $\beta_y = 0.5$. 5 Agents are placed in the harbour at initial positions (0.20, 2.20), (0.80, 1.78), (0.70, 1.35), (0.50, 0.93), and (0.30, 0.50) m as shown in figure 4.1. The Voronoi region is obtained using the generalized model (2.6) and the sensor performance function is defined by

$$f(r_i) = \mu e^{-\Lambda r_i^2} \tag{4.12}$$

with $\mu = 1$ and $\lambda = 0.5$. In this thesis, it is assumed that the agents have the knowledge of the state of the targets in the harbour. To estimate the performance of the algorithm, we define the ith agents discrete time value function as

$$V(\mathbf{e}_i(k)) = \sum_{\kappa=k}^{\infty} (\mathbf{e}_i(\kappa)\mathbf{Q}\mathbf{e}_i(\kappa) + \mathbf{u}_i^T(\kappa)\mathbf{R}\mathbf{u}_i(\kappa)) \tag{4.13}$$

and its discrete time cost function is defined by

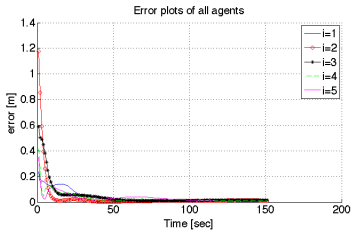
$$J(\mathbf{e}_i(k)) = \sum_{k=0}^{\infty} \left(\mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) + {\mathbf{u}_i}^T(k)\mathbf{R}\mathbf{u}_i(k)\right). \tag{4.14}$$

4.2.1 One Moving Target

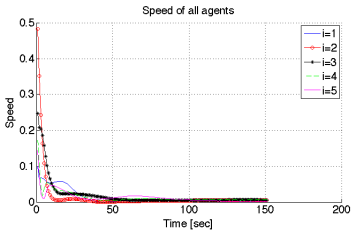
In the first scenario, a single mobile target comes into the harbour from $(2.2, 3.0)$ and moves diagonally with constant velocity, and the agents react to neutralize the target while maintaining centroidal Voronoi configuration. The optimal values of the cost matrix($\boldsymbol{P}^*_{CARE}$), feedback gain matrix $\boldsymbol{K}^*_k$ and the control policy $\boldsymbol{u}^*_k$ obtained when the agents converged to their centroids are presented in Table 4.1 as well as the time taken by each agent to converge to its centroid and the graphs shown in figure 4.2.

Table 4.1: Optimal values at Centroidal Voronoi configuration using MAACC-RL PI: Single target scenario

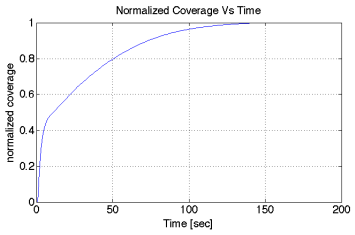
Agent	Optimal P-CARE Matrix	Optimal Policy	Optimal Control (m/s)	Convergence Time (s)
		$\boldsymbol{K_k}=\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}$	$\boldsymbol{u_k^*}=\boldsymbol{K_k e_k}$	
1	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2100 & 0.0037\\ 0.0037 & 0.2087\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4102 & -0.0041\\ 0.0041 & 0.4102\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0079\\ 0.0020\end{bmatrix}$	**34**
2	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2053 & -0.0005\\ -0.0005 & 0.2056\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4111 & 0.0004\\ -0.0004 & 0.4111\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0061\\ 0.0021\end{bmatrix}$	**12**
3	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2048 & 0.0017\\ 0.0017 & 0.2113\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4094 & 0.0069\\ -0.0069 & 0.4094\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}-0.0052\\ 0.0061\end{bmatrix}$	**39**
4	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2082 & 0.0004\\ 0.0004 & 0.2130\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4133 & 0.0002\\ -0.0002 & 0.4133\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0046\\ 0.0063\end{bmatrix}$	**38**
5	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2081 & 0.0001\\ 0.0001 & 0.2111\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4141 & 0.0037\\ -0.0037 & 0.4141\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0075\\ 0.0030\end{bmatrix}$	**33**



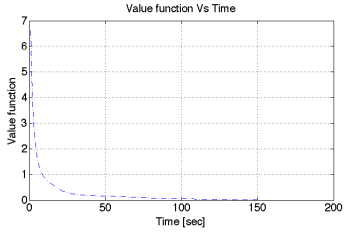
(a) Evolution of the error trajectory



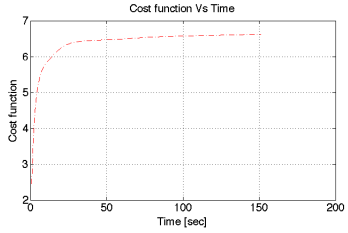
(b) Evolution of control signals trajectory



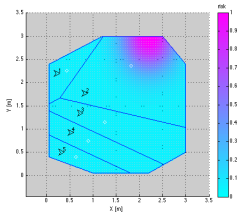
(c) Evolution of Normalized Coverage Metric



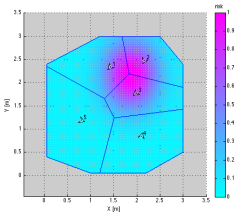
(d) Optimal Value Function at $T = 90$ s



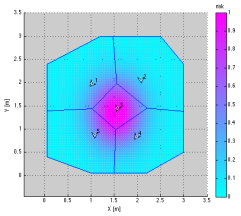
(e) Optimal Cost Function



(f) Initial Configuration at $k = 1$ s



(g) Coverage Configuration at $k = 90$ s



(h) Coverage Final Configuration at $k = 150$ s

Figure 4.2: Neutralizing Single Target Linear Motion using MAACC-RL PI Algorithm: Coverage Plots

Trajectories of the Euclidean norm of the error variables between the agents and their centroids and trajectories of the control signal (speed) of all agents are shown in figures 4.2-(a), and (b) with both graphs showing asymptotic convergence. Also, Figure 4.2(b) shows that the MAACC-RL PI controller regulates the states asymptotically to zero. The coverage metric plot shown in figure 4.2(c) demonstrate that all the agents are guaranteed to configure themselves optimally and the coverage is maximized as desired. Figures 4.2-(d) and (e) show that the value and cost functions are optimized as expected. Figure 4.2(f) shows the agents at initial configurations and directed towards their centroids, a target represented by a purple circular region enters into the harbour. In figures 4.2-(g) and (h), the agents moves towards the target while maintaining centroidal Voronoi configuration with maximum coverage of the area for optimal coverage control after 90s and 150s respectively.

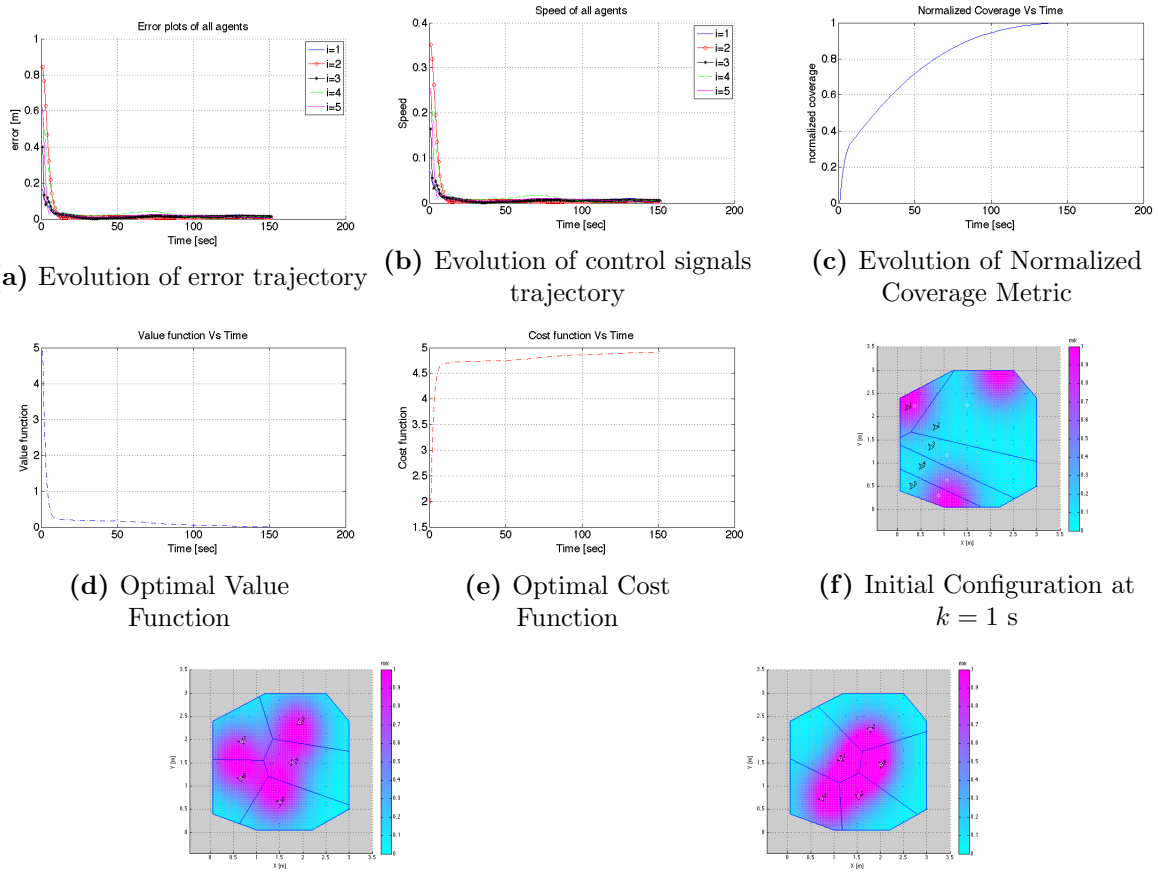
4.2.2 Three Moving Targets

This section summarizes the results obtained from the second simulation scenario with three mobile targets entering the harbour at the same time from initial positions (1.1, 0.1),(0.2, 2.4),(2.2, 3.0). Similar to the first scenario, the agents react to neutralize the target while protecting the harbour. In this case, the agents are divided into three groups based on closeness to the targets. Similar to section 4.2.1, table 4.2 shows the optimal values obtained when the agents converged to their centroids as well as their convergence time. The graphs are shown in figure 4.3

Table 4.2: Optimal values at Centroidal Voronoi configuration using MAACC-RL PI: Three targets scenario

Agent	Optimal P-CARE Matrix	Optimal Policy	Optimal Control (m/s)	Convergence Time (s)
		$\boldsymbol{K_k}=\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}$	$\boldsymbol{u_k^*}=\boldsymbol{K_k e_k}$	
1	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2083 & 0.0004\\ 0.0004 & 0.2068\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4139 & -0.0002\\ 0.0002 & 0.4139\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0063\\ 0.0027\end{bmatrix}$	**11**
2	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2096 & -0.0001\\ -0.0001 & 0.2093\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4175 & 0.0005\\ -0.0005 & 0.4175\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0051\\ 0.0010\end{bmatrix}$	**12**
3	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2117 & -0.0007\\ -0.0007 & 0.2180\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4159 & 0.0076\\ -0.0076 & 0.4159\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}-0.0065\\ 0.0048\end{bmatrix}$	**19**
4	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2047 & 0.0000\\ 0.0000 & 0.2079\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4103 & 0.0026\\ -0.0026 & 0.4103\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0035\\ 0.0074\end{bmatrix}$	**30**
5	$\boldsymbol{P}^*_{\text{CARE}}=\begin{bmatrix}0.2096 & 0.0007\\ 0.0007 & 0.2087\end{bmatrix}$	$\boldsymbol{K_k}=\begin{bmatrix}0.4138 & 0.0009\\ -0.0009 & 0.4138\end{bmatrix}$	$\boldsymbol{u_k^*}=\begin{bmatrix}0.0077\\ 0.0028\end{bmatrix}$	**19**

The plots obtained from the simulation are presented next.



(a) Evolution of error trajectory

(b) Evolution of control signals trajectory

(c) Evolution of Normalized Coverage Metric

(d) Optimal Value Function

(e) Optimal Cost Function

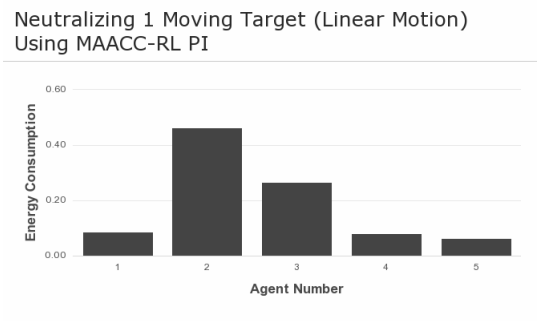
(f) Initial Configuration at $k = 1$ s

(g) Coverage Configuration at $k = 90$ s

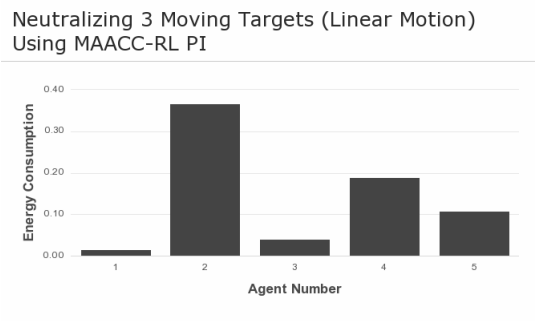
(h) Coverage Final Configuration at $k = 150$ s

Figure 4.3: Neutralizing Multiple Targets Using MAACC-RL PI Algorithm: Coverage Plots

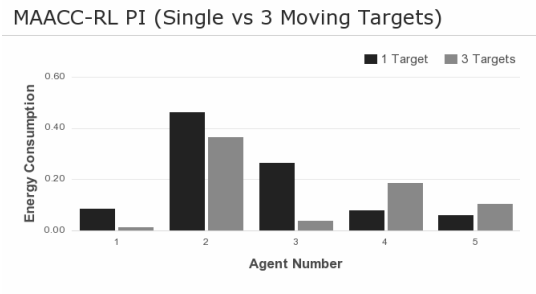
Trajectories of the Euclidean norm of the error variables between the agents and their centroids and trajectories of the control signal (speed) of all agents are shown in Figure 4.3-(a), and (b) with both graphs showing asymptotic convergence. Also, figures 4.3(b) shows that the MAACC-RL PI controller regulates the states asymptotically to zero. The coverage metric plot shown in figure 4.3(c) demonstrate that all the agents are guaranteed to configure themselves optimally and the coverage is maximized as desired irrespective of the number of targets in the harbour. Figures 4.3-(d) and (e) show that the value and cost functions are optimized as expected. Figure 4.3(f) shows the agents at initial configurations and directed towards their centroids with three targets entering into the harbour. In figures 4.3-(g) and (h), the agents moves towards the targets while maintaining centroidal Voronoi configuration with maximum coverage of the area for optimal coverage control after 90s and 150s respectively. Areas with higher risk are densely populated by the agents. Figure 4.4(a and b) shows the energy consumed during the area coverage task in each scenario presented and Figure 4.4(c) shows the comparison of the energy used by each agents in both scenarios.



(a) Neutralizing single target



(b) Neutralizing multiple targets



(c) Single target Vs. Multiple targets

Figure 4.4: Energy consumption comparison in both scenarios using MAACC-RL PI Algorithm

4.2.3 Summary and Discussion

In this section, an online tuning algorithm is introduced to solve the continuous time linear quadratic regulator coverage control problem formulated in chapter 3. The results presented in the numerical simulation are directly applicable in practice since they provide means to solve the continuous time adaptive finite-horizon optimal control problem. The proposed approach achieved centroidal Voronoi Configuration for the agents.

4.3 Actor-Critic Optimal Control Solution Approach using Reinforcement Learning

This section uses optimal control theory and approximate dynamic programming to develop an online adaptive reinforcement learning-based solution for discrete-time version of the optimal control problem (3.13) developed from the centroidal Voronoi tessellation-based area coverage control problem. An actor-critic is developed using two neural network structures namely the actor and critic neural networks.

Section 4.3 starts with a brief description of the system model and optimal control problem. Thereafter, an actor-critic neural network online tuning algorithm is proposed to solve the optimal control problem. The actor-critic neural network is implemented using

a reinforcement learning framework based on a recursive least-squares solution with the synchronous discrete-time adaptation of both actor and critic neural networks to guarantee convergence and closed-loop stability. This is one of the main contributions presented in this thesis. The proposed algorithm used to provide area coverage control by the agents is presented afterwards. Section 4.4 present simulation results that illustrate the effectiveness of the theoretical framework presented.

4.3.1 Problem Formulation

Consider the discrete-time linear system given by

$$\mathbf{e}_i(k+1) = \mathbf{A}_i(k)\mathbf{e}_i(k) + \mathbf{B}_i(k)\mathbf{u}_i(k), \quad k = 0, 1, \ldots, \tag{4.15}$$

where $\mathbf{e}_i \in \mathbb{R}^2$, $\mathbf{u}_i \in \mathbb{R}^m$, $\mathbf{A}_i \in \mathbb{R}^{2\times 2}$, $\mathbf{B}_i \in \mathbb{R}^{2\times 2}$, $\mathbf{e}_i(0) = \mathbf{e}_0$ is the initial state and $\mathbf{e}_i(k) \in \chi \subseteq \mathbb{R}^n$, $\mathbf{u}_i(k) \in \mathcal{U} \subseteq \mathbb{R}^m$ are the system state and control input, respectively.

Given a two-dimensional area denoted by Ω, the problem is to find an optimal control policy $\mathbf{u}_i^*((\mathbf{e}_i(k))$ such that $\lim_{k\to\infty} \mathbf{e}_i(k) = 0$ and $\lim_{k\to\infty} V((\mathbf{e}_i(k)) = 0$ where V is the Value function. For the optimal linear quadratic regulator problem, there exists a control policy input that minimizes the value function given as

$$V(\mathbf{e}_i(k)) = \sum_{\kappa=k}^{\infty} \left(\mathbf{e}_i^T(\kappa)\mathbf{Q}\mathbf{e}_i(\kappa) + \mathbf{u}_i^T(\kappa)\mathbf{R}\mathbf{u}_i(\kappa)\right) \tag{4.16}$$

where $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ are positive definite matrices i.e. $\mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) > 0, \forall \mathbf{e}_i \neq 0$ and $\mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) = 0$ when $\mathbf{e}_i = 0$. Note that the second term of the value function (4.16) takes into account the asymptotic energy consumption of the agents. Equation (4.16) can be rewritten as an update value function in the form:

$$\begin{aligned} V(\mathbf{e}_i(k)) &= \mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i^T(k)\mathbf{R}\mathbf{u}_i(k) + \sum_{\kappa=k+1}^{\infty} \left(\mathbf{e}_i^T(\kappa)\mathbf{Q}\mathbf{e}_i(\kappa) + \mathbf{u}_i^T(\kappa)\mathbf{R}\mathbf{u}_i(\kappa)\right) \\ V(\mathbf{e}_i(k)) &= \mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i^T(k)\mathbf{R}\mathbf{u}_i(k) + V(\mathbf{e}_i(k+1)). \end{aligned} \tag{4.17}$$

Hence, we find the control inputs $\mathbf{u}_i$ such that the value function (4.16) is minimized.

$$V^*(\mathbf{e}_i(k)) = \min_{\mathbf{u}_i(k)} \left(\mathbf{e}_i(k)^T\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i(k)^T\mathbf{R}\mathbf{u}_i(k) + V^*(\mathbf{e}_i(k+1))\right) \tag{4.18}$$

Model (4.18) is the discrete-time Hamilton-Jacobi-Bellman equation. The optimal control, $\mathbf{u}_i^*$ can be obtained by finding the gradient of the right hand side of (4.18) and setting it to zero i.e. $\left(\dfrac{\partial V^*(\mathbf{e}_i(k))}{\partial \mathbf{u}_i} = 0\right)$, we obtain

$$\frac{\partial\left(\mathbf{e}_i^T(k)\mathbf{Q}\mathbf{e}_i(k) + \mathbf{u}_i(k)^T\mathbf{R}\mathbf{u}_i(k)\right)}{\partial \mathbf{u}_i} + \left(\frac{\partial \mathbf{e}_i(k+1)^T}{\partial \mathbf{u}_i}\right)\left(\frac{\partial V_i^*(\mathbf{e}_i(k+1))}{\partial \mathbf{e}_i(k+1)}\right) = 0$$

$$2\mathbf{u}_i\mathbf{R} + \mathbf{B}_i^T(\mathbf{e}_i(k))\frac{\partial V^*(\mathbf{e}_i(k+1))}{\partial \mathbf{e}_i(k+1)} = 0$$

$$\mathbf{u}_i^* = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}_i^T\frac{\partial V^*(\mathbf{e}_i(k+1))}{\partial \mathbf{e}_i(k+1)}$$

Note that $V^*(\mathbf{e}_i(k))$ represents the value function consistent with the optimal control policy $\mathbf{u}_i^*(\mathbf{e}_i(k))$. Since the HJB equation is a nonlinear equation, it is generally difficult or impossible to obtain its solution. Therefore we propose an actor critic NN approximation algorithm that approximates both the value function and the control policy.

4.3.2 Actor-Critic Reinforcement Learning Solution Approach

Select a value function approximation (critic neural network) and a control action approximation structure (actor neural network) in the framework of [21] as

$$\hat{V}_j\big(\mathbf{e}_i(k)\big) = \mathbf{w}_{\mathbf{c,j}}^{\mathbf{T}}\boldsymbol{\rho}\big(\mathbf{e_i(k)}\big) \tag{4.19}$$

$$\hat{\mathbf{u}}_j\big(\mathbf{e}_i(k)\big) = \mathbf{W}_{\mathbf{a,j}}\boldsymbol{\sigma}\big(\mathbf{e_i(k)}\big) \tag{4.20}$$

respectively, where $\hat{V}$ and $\hat{\mathbf{u}}$ are estimates of the value function and control action respectively, $\mathbf{w_{c,j}} \in \mathbb{R}^{\mathbf{N_c}}$ represents the weight vector of critic neural network while $\mathbf{w_{a,j}} \in \mathbb{R}^{\mathbf{N_a \times 2}}$ represents the weight matrix of actor neural network, N_c and N_a are the number of neurons in the critic and actor neural network respectively, j is the iteration step for both actor and critic neural network, $\boldsymbol{\rho}_j(\mathbf{e}_i(k))$ and $\boldsymbol{\sigma}_j(\mathbf{e}_i(k))$ are the activation functions of the critic and actor neural network respectively. Since the initial conditions required that $\hat{V}_j(0) = 0$ and $\hat{\mathbf{u}}_j(0) = 0$, the activation functions are defined as $\boldsymbol{\rho}(0) = 0$ and $\boldsymbol{\sigma}(0) = 0$.

The critic neural network activation function vector [21] is given as

$$\boldsymbol{\rho}_j(\mathbf{e}_i(k)) \equiv [\boldsymbol{\rho}_1(\mathbf{e}_i(k))\boldsymbol{\rho}_2(\mathbf{e}_i(k))\dots\boldsymbol{\rho}_L(\mathbf{e}_i(k))]^T,$$

while the critic neural network weight vector [21] at iteration j is $\mathbf{w_{c,j}} \equiv [\mathbf{w_c^{1,1}}, \mathbf{w_c^{1,2}}, \mathbf{w_c^{1,3}}, \mathbf{w_c^{1,4}}, \mathbf{w_c^{1,5}}]^{\mathbf{T}}$. The actor neural network weight vector activation function is given as

$$\boldsymbol{\sigma}_j(\mathbf{e}_i(k)) \equiv [\sigma_1(\mathbf{e}_i(k))\ \sigma_2(\mathbf{e}_i(k))\dots\sigma_M(\mathbf{e}_i(k))]^T,$$

while the actor neural network vector weight is given as $\mathbf{W_{a,j}} \equiv \begin{bmatrix} w_a^{1,1} & w_a^{1,2} & w_a^{1,3} & w_a^{1,4} & w_a^{1,5} \\ w_a^{2,1} & w_a^{2,2} & w_a^{2,3} & w_a^{2,4} & w_a^{2,5} \end{bmatrix}^{\mathbf{T}}$.

The target value function and control function can be defined as

$$\begin{aligned} V_{j+1}\big(\mathbf{e}_i(k)\big) &= \boldsymbol{e}_i(k)^T\boldsymbol{Q}\boldsymbol{e}_i(k) + \boldsymbol{u}_i(k)^T\boldsymbol{R}\boldsymbol{u}_i(k) + V_j\big(\boldsymbol{e}_i(k+1)\big) \\ &= \boldsymbol{e}_i(k)^T\boldsymbol{Q}\boldsymbol{e}_i(k) + \boldsymbol{u}_i(k)^T\boldsymbol{R}\boldsymbol{u}_i(k) + \mathbf{w}_{\mathbf{c,j}}^{\mathbf{T}}\boldsymbol{\rho}\big(\mathbf{e_i(k+1)}\big) \end{aligned} \tag{4.21}$$

$$\boldsymbol{u}_{j+1}\big(\mathbf{e}_i(k)\big) = -\frac{1}{2}\boldsymbol{R}^{-1}\boldsymbol{B}_i^T(k)\nabla\boldsymbol{\rho}^T\big(e_i(k+1)\big)\mathbf{w_{c,j}} \tag{4.22}$$

where V_{j+1} and $\boldsymbol{u}_{j+1}$ are defined as the target update value function and control action update respectively.

4.3.3 Least-Squares Update Rule for Tuning Critic Neural Network

The generalized temporal difference error equation for the critic neural network can be derived as:

$$\delta_{c,j}(k) = \mathbf{w_{c,j}^T}\boldsymbol{\rho}\big(\mathbf{e_i(k)}\big) - \mathbf{V_{j+1}}\big(\mathbf{e_i(k)}\big) \tag{4.23}$$

Define the critic neural network least-squares error $E_{c,j}^2$ to be minimized as

$$E_{c,j}^2 := \sum_{l=1}^{N_t} \frac{1}{2}\big(\delta_{c,j}^l(k)\big)^2 = \frac{1}{2}\sum_{l=1}^{N_t}\Big[\mathbf{w_{c,j}^T}\boldsymbol{\rho}^{\mathbf{l}}\big(\mathbf{e_i(k)}\big) - \mathbf{V_{j+1}^l}\big(\mathbf{e_i(k)}\big)\Big]^{\mathbf{2}} \tag{4.24}$$

where $(l = 1, \ldots, N_t)$ and N_t = Number of training steps.

Differentiating $E_{c,j}^2$ in terms of $\mathbf{w_c}$ we get

$$\frac{\partial E_{c,j}^2}{\partial \mathbf{w}_{c,j+1}} = \left(\frac{\partial E_{c,j}^2}{\partial \delta_{c,j}}\right)\left(\frac{\partial \delta_{c,j}}{\partial \mathbf{w}_{c,j+1}}\right) = \frac{1}{2}\sum_{l=1}^{N_t}\Big\{2\Big[\mathbf{w_{c,j+1}^T}\boldsymbol{\rho}^{\mathbf{l}}\big(\mathbf{e_i(k)}\big) - \mathbf{V_{j+1}^l}\big(\mathbf{e_i(k)}\big)\Big]\Big\}\boldsymbol{\rho}^{\mathbf{l}}\big(\mathbf{e_i(k)}\big) \tag{4.25}$$

Equating (4.25) to zero and rearranging the equation, we obtain the critic weight update least square solution of the form:

$$\hat{\mathbf{w}}_{c,j+1}\big(\mathbf{e}_i(k)\big) = \left(\sum_{l=1}^{N_t}\boldsymbol{\rho}^l\big(\boldsymbol{e}_i(k)\big)\boldsymbol{\rho}^l\big(\boldsymbol{e}_i(k)\big)^T\right)^{-1}\left(\sum_{l=1}^{N_t}\boldsymbol{\rho}^l\big(\boldsymbol{e}_i(k)\big)V_{j+1}^l\big(\mathbf{e}_i(k)\big)\right). \tag{4.26}$$

4.3.4 Least-Squares Update Rule for Tuning Actor Neural Network

Likewise, we define the actor neural network training error as

$$\delta_{a,j}(k) = \mathbf{w_{a,j+1}^T}\boldsymbol{\sigma}\big(\mathbf{e_i(k)}\big) - \mathbf{u_{j+1}}\big(\mathbf{e_i(k)}\big) \tag{4.27}$$

where $\delta_{a,j} \in \mathbb{R}^m$ is the actor training error. We define the actor neural network least-squares error $E_{a,j}^2$ to be minimized as

$$E_{a,j}^2 := \sum_{l=1}^{N_t} \frac{1}{2}\big(\delta_{a,j}^l(k)\big)^2 = \frac{1}{2}\sum_{l=1}^{N_t}\Big[\mathbf{W_{a,j+1}^T}\boldsymbol{\sigma}^{\mathbf{l}}\big(\mathbf{e_i(k)}\big) - \mathbf{u_{j+1}^l}\big(\mathbf{e_i(k)}\big)\Big]^{\mathbf{2}} \tag{4.28}$$

Differentiating $E_{a,j}^2$ in terms of $\mathbf{w_a}$, we obtain

$$\frac{\partial E_{a,j}^2}{\partial \mathbf{W}_{a,j+1}} = \left(\frac{\partial E_{a,j}^2}{\partial \boldsymbol{\delta}_{a,j}}\right)\left(\frac{\partial \boldsymbol{\delta}_{a,j}}{\partial \mathbf{W}_{a,j+1}}\right) = \frac{1}{2}\sum_{l=1}^{N_t}\Big\{2\Big[\mathbf{w_{a,j+1}^T}\boldsymbol{\sigma}^{\mathbf{l}}\big(\mathbf{e_i(k)}\big) - \mathbf{u_{j+1}^l}\big(\mathbf{e_i(k)}\big)\Big]\Big\}\boldsymbol{\sigma}^{\mathbf{l}}\big(\mathbf{e_i(k)}\big). \tag{4.29}$$

Equating (4.29) to zero and rearranging the equation, we obtain actor neural network training target function as follows:

$$\boldsymbol{u}_{j+1}\big(\mathbf{e}_i(k)\big) = \widehat{\mathbf{W}}_{a,j}^T \boldsymbol{\sigma}\big(\mathbf{e}_i(k)\big) = -\frac{1}{2}\boldsymbol{R}^{-1}\boldsymbol{B}_i^T(k)\nabla\boldsymbol{\rho}^T\big(\mathbf{e}_i(k+1)\big)\widehat{\mathbf{w}}_{c,j} \tag{4.30}$$

While the actor weight update least square solution is of the form:

$$\hat{\mathbf{W}}_{a,j+1}\big(\mathbf{e}_i(k)\big) = \left(\sum_{l=1}^{N_t} \boldsymbol{\sigma}^l\big(\boldsymbol{e}_i(k)\big)\boldsymbol{\sigma}^l\big(\boldsymbol{e}_i(k)\big)^T\right)^{-1}\left(\sum_{l=1}^{N_t} \boldsymbol{\sigma}^l\big(\boldsymbol{e}_i(k)\big)\mathbf{u}_{j+1}^l\big(\mathbf{e}_i(k)\big)\right). \tag{4.31}$$

The least square solution weights $\hat{\mathbf{w}}_{c,j}$ and $\hat{\mathbf{W}}_{a,j}$ guarantees system stability as well as convergence to the optimal value and control [22]. The actor-critic neural network weights are synchronized as follows:

$$\hat{\mathbf{W}}_{a,j+1}\big(\mathbf{e}_i(k)\big) := -\Big[\alpha\big(\hat{\mathbf{W}}_{a,j+1}\big(\mathbf{e}_i(k)\big) - \hat{\mathbf{w}}_{c,j+1}\big(\mathbf{e}_i(k)\big)\big) + (1-\alpha)\hat{\mathbf{W}}_{a,j+1}\big(\mathbf{e}_i(k)\big)\Big] \tag{4.32}$$

where α is the learning rate for the Actor-Critic neural network. Note that for the inverse of matrix of the critic $[\boldsymbol{\rho}(\boldsymbol{e}_i(k))\boldsymbol{\rho}^T(\boldsymbol{e}_i(k))]^{-1}$ and actor $[\boldsymbol{\sigma}(\boldsymbol{e}_i(k))\boldsymbol{\sigma}^T(\boldsymbol{e}_i(k))]^{-1}$ to exist, one needs the basis functions $\boldsymbol{\rho}(\boldsymbol{e}_i(k))$ and $\boldsymbol{\sigma}(\boldsymbol{e}_i(k))$ to be linearly independent and the number of random states to be greater than or equal to the number of neurons, N_a and N_c for the actor and critic respectively (i.e., the matrix determinants must not be zero).
The output of the critic is used in the training process of the actor so that Least-Squares control policy can be computed recursively. In the proposed least square framework, the learning process employs a weight-in-line actor and critic neural network implemented using a recursive least-squares algorithm to approximate both the actor and critic weights which are tuned synchronously using (4.32).
At each iteration step, the least square solver collects the data needed to calculate states $\mathbf{e}_i(k)$, and control update $\mathbf{u}_{j+1}(\mathbf{e}_i(k))$, and then finds the weight vectors and matrix $\hat{\mathbf{w}}_{c,j+1}$ and $\hat{\mathbf{W}}_{a,j+1}$ satisfying (4.26) and (4.31) respectively, both of which are transferred to the corresponding actor and critic neural networks to update their weights. The actor neural network generates the control input $\hat{\mathbf{u}}_{j+1}(\mathbf{e}_i(k))$ while the critic neural network generates the value function output $\hat{V}_{j+1}(\mathbf{e}_i(k))$. The actor least square solver generates the optimal action weights, $(\hat{\mathbf{W}}_{a,j+1})$ while the critic LS solver generates the optimal value weights,$(\hat{\mathbf{w}}_{c,j+1})$, when $\|\hat{V}_{j+1}(\mathbf{e}_i(k)) - \hat{V}_j(\mathbf{e}_i(k))\| \leq \epsilon$ and $\|\hat{\mathbf{w}}_{c,j+1}(k) - \hat{\mathbf{w}}_{c,j}(k)\| \leq \epsilon$ and $\|\hat{\mathbf{W}}_{a,j+1}(k) - \hat{\mathbf{W}}_{a,j}(k)\| \leq \epsilon$ which is then used in generating the optimal control policy (where ϵ is a defined convergence tolerance).
The optimal cost matrix $(\mathbf{P}^*_{DARE})$ can be obtained from the optimal critic weight vector similar to [21] by

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} w_c^{1,3} & w_c^{1,4} \\ w_c^{1,4} & w_c^{1,5} \end{bmatrix} \tag{4.33}$$

The optimal gain matrix at time instant k, $\mathbf{L}_k^*$ is updated by

$$\mathbf{L}_k^* = -\mathbf{R}^{-1}\mathbf{B}_i^T\mathbf{P}^* \tag{4.34}$$

Similarly, the optimal gain matrix at time instant k, $\mathbf{L}_k^*$ is related to the optimal synchronized actor weights [21] and therefore can also be updated by

$$\mathbf{L}_k^* = \begin{bmatrix} w_a^{1,1} & w_a^{1,2} \\ w_a^{2,1} & w_a^{2,2} \end{bmatrix} \tag{4.35}$$

4.3.5 Multi-Agent Area Coverage Control Reinforcement Learning Algorithm using Neural Network

The flowchart of the proposed multi-agent area coverage control reinforcement learning algorithm using Neural network (MAACC-RL NN) that move the agents to their respective centroidal Voronoi configuration while providing area coverage is described and shown in figure 4.5 below:

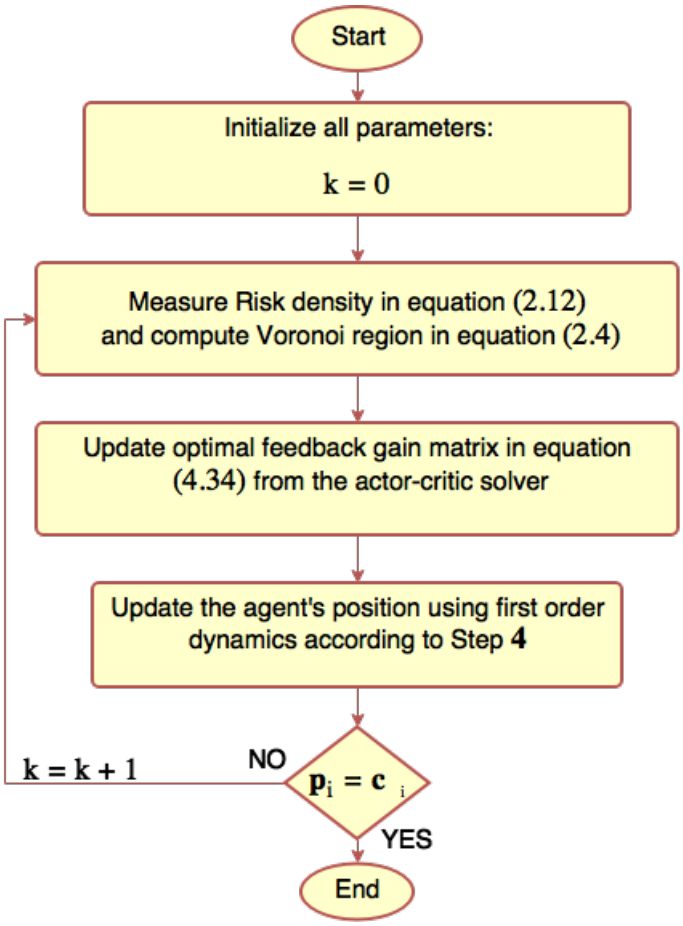


Figure 4.5: High level steps(flowchart) of the proposed MAACC-Rl NN algorithm

Step 1: Initialize all parameters such as the initial time and agent's initial position and define the bounds for the workspace. Likewise define the learning rate (α),convergence tolerance (ϵ) and the initial weight vector($\mathbf{w}_c$) and matrix ($\mathbf{W}_a$).

Step 2: Measure the Risk density ϕ and compute the Voronoi region $\mathcal{V}_i$ obtaining $m_{\mathcal{V}_i}$, $\mathbf{c}_{\mathcal{V}_i}$, $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial \mathbf{p}_i}$, $\frac{\partial \mathbf{c}_{\mathcal{V}_i}}{\partial t}$ and $\frac{\partial m_{\mathcal{V}_i}}{\partial t}$.

Step 3: Obtain the optimal feedback gain matrix, (4.34) using actor-critic neural network approximation technique described in section 4.3.2 .

Step 4: Update the ith agent position using first order dynamics (3.1) with $\mathbf{u}_i = \mathbf{L}_k^* \mathbf{e}_i(k)$

Step 5: If agent $\mathbf{p}_i$ converges to its centroid $\mathbf{c}_i$, **stop procedure**, else go to **Step 2**

4.4 Simulation Results for Actor-Critic Solution Approach

This section demonstrates the applicability of the developed technique. It validates the proposed distributed multi-agent area coverage control reinforcement learning algorithm using Neural Network via numerical simulations.

To evaluate the efficiency of the proposed algorithms, several simulations with five agents and identical monitoring constraints, are performed in a 2D harbour-like environment with convex polygon shape. The theoretical framework which shows that the agents should achieve centroidal Voronoi configuration while providing maximum coverage of the area using minimum actuator energy is validated through simulation results. Furthermore, the results obtained using the proposed algorithm are compared to those obtained using the control law proposed by [15] defined in equation (3.2) and thus labeled as CORTES-CVT in this thesis.

Several performance measures of optimality are considered which includes: (1) the minimum energy consumption as a measure of the agents' overall energy expenditure during a coverage mission; (2) the norm of the error signal $\|e(t)\|$ over simulation time; (3) the action signal $u(t)$ as a measure of the agent's influence on the coverage environment; (4) the coverage metric $\boldsymbol{\mathcal{H}}$ over time (2.13) as a measure of the overall coverage quality (accuracy); and (5) the evolution of the value function $V(e(t))$ over time (4.13). These metrics are used in benchmarking simulation results throughout this section.

We define the harbour as a convex polygonal region with vertices (1.0, 0.05), (2.2, 0.05), (3.0, 0.5), (3.0, 2.4), (2.5, 3.0), (1.2, 3.0), (0.05, 2.4) and (0.05, 0.4). The prior risk in the harbour is defined as $\phi_0 = 5 \times 10^{-2}, \forall \mathbf{q} \in \Omega$. The density function parameters $\beta_x = 0.4$ and $\beta_y = 0.5$. 5 agents are placed in the harbour at initial positions (0.20, 2.20), (0.80, 1.78), (0.70, 1.35), (0.50, 0.93), and (0.30, 0.50) m. The Voronoi region is obtained using the generalized model (2.6) and the sensor performance function is defined by (4.12) with $\kappa = 1$ and $\lambda = 0.5$. The value function and the cost function are defined by (4.13) and (4.14) respectively. The sampling time in all simulations is chosen to be 1s and the simulation is conducted for 150s. In all simulations, the target moves at a constant speed of 0.01m/s inside the harbour.

The neuron of the neural network that approximates the value function and control for critic and actor neural networks are defined as $\boldsymbol{\rho}(\mathbf{e}(k)) = [e_1 \ e_2 \ e_1^2 \ 2e_1e_2 \ e_2^2]$ and $\sigma(\mathbf{e}(k)) =$

$[e_1\ e_2\ e_1^2\ 2e_1e_2\ e_2^2]$ respectively. The derivative of the critic NN neuron vector (activation function) in terms of the error $\mathbf{e}(k)$ is given as $\frac{\partial \boldsymbol{\rho}(\mathbf{e}(k))}{\partial \mathbf{e}(k)} = \begin{bmatrix} 1 & 0 & 2e_1 & 2e_2 & 0 \\ 0 & 1 & 0 & 2e_1 & 2e_2 \end{bmatrix}^T$. $\mathbf{Q} = I_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, and $\mathbf{R} = I_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The actor-critic learning rate $\alpha = 0.0009$ and the proportional gain, $k_p = 1$.

Different scenarios were simulated which includes (i) Single target scenario with linear motion using MAACC-RL NN, (ii) Single target scenario with linear motion using CORTES-CVT, (iii) Multiple targets scenario with linear motion using MAACC-Rl NN, (iv) Multiple targets scenario with linear motion using CORTES-CVT, (v) Single target scenario with oscillatory motion using MAACC-RL NN, (vi) Single target scenario with oscillatory motion using CORTES-CVT. All scenarios are simulated under similar condition. Next section evaluates and summarizes simulation results obtained.

4.4.1 Single target scenario with linear motion using MAACC-RL NN

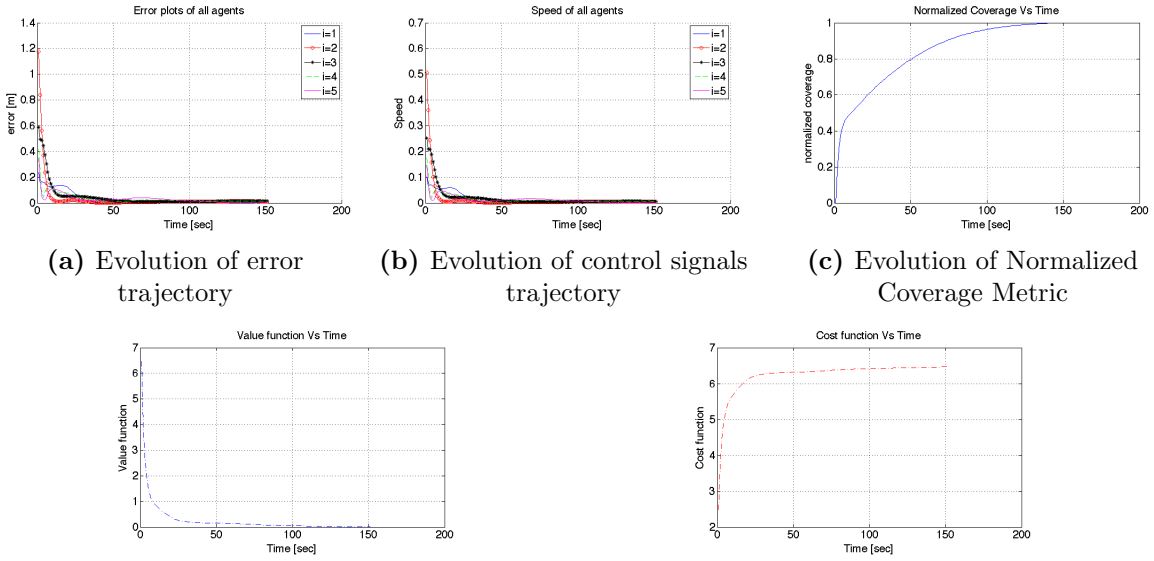
This section presents results of simulated scenario with a single moving target with linear motion using the proposed MAACC-RL NN Algorithm for coverage control. A single mobile target comes into the harbour from (2.2, 3.0) and moves diagonally with constant velocity. The agents follow the target while maintaining centroidal Voronoi configuration. The optimal values obtained when the agents achieve centroidal Voronoi configuration is summarized in Table 4.3 and the simulation graphs are shown in figure 4.6 below.

Table 4.3: Optimal values at centroidal Voronoi configuration using MAACC-RL NN: Single target scenario

Convergence Parameter	**Optimal Solution Matrices**		**Optimal Control**
		$\boldsymbol{L_k = R^{-1}B^TP}$	$\boldsymbol{u_k^* = L_k e_k}$
Agent 1	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2200 & 0.0037 \\ 0.0037 & 0.2186 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4299 & 0.0046 \\ -0.0041 & -0.4300 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0069 \\ 0.0016 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4065 & -0.0041 & 0.0000 & 0.0000 & 0.0000 \\ 0.0045 & 0.4064 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$		
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2200 & 0.0037 & 0.2186 \end{bmatrix}$		
The synchronized control (Actor NN) weights converge to:	$\mathbf{w^{*T}_{a,sync}} = \begin{bmatrix} -0.4065 & 0.0041 & 0.2202 & 0.0037 & 0.2188 \\ -0.0045 & -0.4064 & 0.2202 & 0.0037 & 0.2188 \end{bmatrix}$		
Agent 2	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2145 & -0.0005 \\ -0.0005 & 0.2147 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4295 & -0.0004 \\ 0.0003 & -0.4294 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0069 \\ 0.0030 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4055 & 0.0003 & -0.0000 & 0.0000 & 0.0000 \\ -0.0004 & 0.4055 & -0.0000 & 0.0000 & 0.0000 \end{bmatrix}$		

Table 4.3 Continued from previous page

Convergence Parameter	Optimal Solution Matrices	Optimal Control
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2145 & -0.0005 & 0.2147 \end{bmatrix}$	
The synchronized control (Actor NN) weights converge to:	$\mathbf{w_{a,sync}^{*T}} = \begin{bmatrix} -0.4055 & -0.0003 & 0.2147 & -0.0005 & 0.2149 \\ 0.0004 & -0.4055 & 0.2147 & -0.0005 & 0.2149 \end{bmatrix}$	
Agent 3	$\boldsymbol{P}^*_{\mathrm{DARE}} = \begin{bmatrix} 0.2157 & 0.0017 \\ 0.0017 & 0.2225 \end{bmatrix}$ $\boldsymbol{L_k} = \begin{bmatrix} -0.4312 & -0.0073 \\ 0.0075 & -0.4308 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} -0.0051 \\ 0.0064 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4072 & 0.0074 & -0.0000 & 0.0000 & 0.0000 \\ -0.0072 & 0.4076 & -0.0000 & 0.0000 & 0.0000 \end{bmatrix}$	
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2157 & 0.0017 & 0.2225 \end{bmatrix}$	
The synchronized control (Actor NN) weights converge to:	$\mathbf{w_{a,sync}^{*T}} = \begin{bmatrix} -0.4072 & -0.0074 & 0.2159 & 0.0017 & 0.2227 \\ 0.0072 & -0.4076 & 0.2159 & 0.0017 & 0.2227 \end{bmatrix}$	
Agent 4	$\boldsymbol{P}^*_{\mathrm{DARE}} = \begin{bmatrix} 0.2152 & 0.0005 \\ 0.0005 & 0.2201 \end{bmatrix}$ $\boldsymbol{L_k} = \begin{bmatrix} -0.4271 & -0.0001 \\ 0.0002 & -0.4268 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0049 \\ 0.0069 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4033 & 0.0002 & -0.0000 & 0.0000 & 0.0000 \\ -0.0001 & 0.4036 & -0.0000 & 0.0000 & 0.0000 \end{bmatrix}$	
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2152 & 0.0005 & 0.2201 \end{bmatrix}$	
The synchronized control (Actor NN) weights converge to:	$\mathbf{w_{a,sync}^{*T}} = \begin{bmatrix} -0.4033 & -0.0002 & 0.2153 & 0.0005 & 0.2203 \\ 0.0001 & -0.4036 & 0.2153 & 0.0005 & 0.2203 \end{bmatrix}$	
Agent 5	$\boldsymbol{P}^*_{\mathrm{DARE}} = \begin{bmatrix} 0.2142 & 0.0001 \\ 0.0001 & 0.2174 \end{bmatrix}$ $\boldsymbol{L_k} = \begin{bmatrix} -0.4262 & -0.0039 \\ 0.0039 & -0.4260 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0074 \\ 0.0041 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4024 & 0.0038 & -0.0000 & 0.0000 & 0.0000 \\ -0.0038 & 0.4026 & -0.0000 & 0.0000 & 0.0000 \end{bmatrix}$	
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2142 & 0.0001 & 0.2174 \end{bmatrix}$	
The synchronized control (Actor NN) weights converge to:	$\mathbf{w_{a,sync}^{*T}} = \begin{bmatrix} -0.4024 & -0.0038 & 0.2144 & 0.0001 & 0.2176 \\ 0.0038 & -0.4026 & 0.2144 & 0.0001 & 0.2176 \end{bmatrix}$	



(a) Evolution of error trajectory

(b) Evolution of control signals trajectory

(c) Evolution of Normalized Coverage Metric

(d) Optimal Value Function

(e) Optimal Cost Function

Figure 4.6: Single target scenario with linear motion using MAACC-RL NN: Coverage Plots

Interestingly, the optimal critic weights converges to the solution of the algebraic Riccati equation ($\mathbf{P}^*_{DARE}$ matrix) (4.33). Also, the optimal feedback gain matrix, $\mathbf{L}^*_k$ computed using equation (4.34) is similar to that obtained from the synchronized actor weights, $\mathbf{W}^{*T}_{a,sync}$ using equation (4.35).

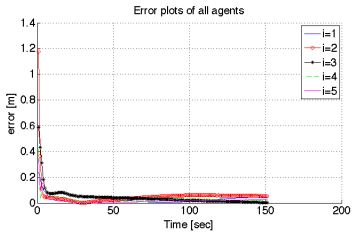
Trajectories of the Euclidean norm of the error variables between the agents and their centroids and trajectories of the control signal (speed) of all agents are shown in Figure 4.6-(a) and (b) with both graphs showing asymptotic convergence. Also, Figure 4.6(b) shows that the MAACC-RL NN controller regulates the states asymptotically to zero. The coverage metric plot shown in figure 4.6(c) demonstrate that all the agents are guaranteed to configure themselves optimally and the coverage is maximized as desired. Figures 4.6-(d) and (e) show that the value and cost functions are optimized as expected.

Next section presents CORTES-CVT results for neutralizing a single target modeled with linear motion.

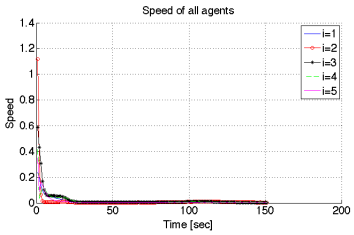
4.4.2 Single target scenario with linear motion using CORTES-CVT

In this case, a single target comes into the harbour under similar conditions as section 4.4.1. The plots obtained from the simulation are shown in Figure 4.7 below.

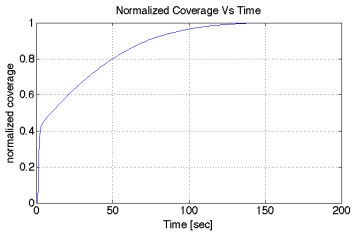
The trajectories of the Euclidean norm of the error variables and trajectories of the control signal (speed) of all agents are shown in figures 4.7-(a) and (b) with both graphs showing asymptotic convergence. The coverage metric plot shown in figure 4.7(c) demonstrate that



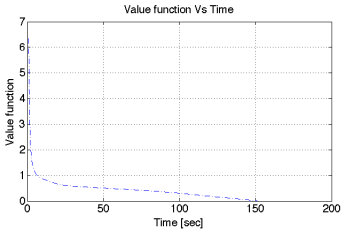
(a) Evolution of error trajectory



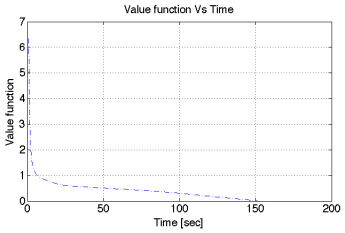
(b) Evolution of control signals trajectory



(c) Evolution of Normalized Coverage Metric



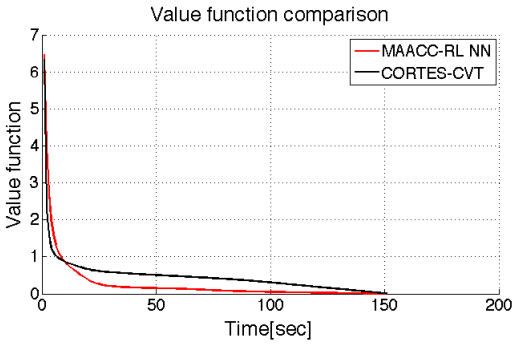
(d) Optimal Value Function



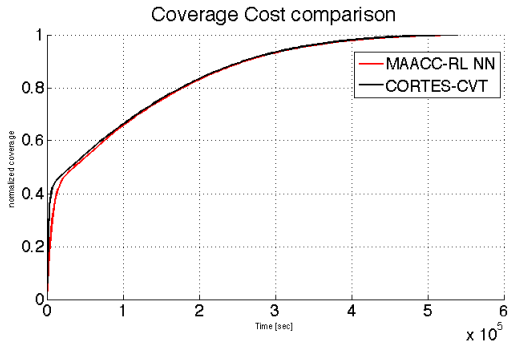
(e) Optimal Cost Function

Figure 4.7: Single target scenario with linear motion using CORTES-CVT: Coverage Plots

all the agents are guaranteed to configure themselves in an optimal fashion. Figures 4.7-(d) and (e) show that the value and cost functions are optimized as expected. The energy consumed in this scenario is compared to the energy used in section 4.4.1. Also, the value function and coverage metric are compared in figure 4.8. The value function converges faster using the developed algorithm while the coverage metric is maximized at almost the same rate.



(a) Value function comparison



(b) Coverage metric comparison

Figure 4.8: Performance of Proposed algorithm Vs CORTES-CVT

4.4.3 Multiple targets scenario with linear motion using MAACC-RL NN

This section demonstrates the scenario with three mobile targets entering the harbour at the same time from initial positions (1.1, 0.1),(0.2, 2.4) and (2.2, 3.0). The agents moves

so as to neutralize the targets while protecting the harbour. In this case, the agents divide themselves into three groups based on closeness to the targets. Table 4.4 shows the optimal values obtained when optimal configuration of all agents to their respective centroids was achieved and figure 4.9 shows the simulation graphs.

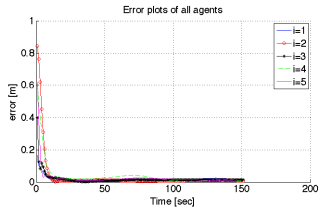
Table 4.4: Optimal values at centroidal Voronoi configuration using MAACC-RL NN: Three targets scenario

Convergence Parameter	**Optimal Solution Matrices**		**Optimal Control**
		$\boldsymbol{L_k = R^{-1}B^TP}$	$\boldsymbol{u_k^* = L_k e_k}$
Agent 1	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2142 & 0.0004 \\ 0.0004 & 0.2129 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4257 & 0.0006 \\ -0.4259 & 0.0002 \\ -0.0001 & -0.4260 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0075 \\ -0.0033 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4021 & -0.0001 & -0.0000 & 0.0000 & 0.0000 \\ 0.0002 & 0.4021 & -0.0000 & -0.0000 & 0.0000 \end{bmatrix}$		
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2142 & 0.0004 & 0.2129 \end{bmatrix}$		
The synchronized control (Actor NN) weights converge to:	$\mathbf{w^{*T}_{a,sync}} = \begin{bmatrix} -0.4021 & 0.0001 & 0.2144 & 0.0004 & 0.2131 \\ -0.0002 & -0.4021 & 0.2144 & 0.0004 & 0.2131 \end{bmatrix}$		
Agent 2	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2155 & -0.0001 \\ -0.0001 & 0.2153 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4295 & -0.0005 \\ 0.0005 & -0.4295 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0077 \\ 0.0021 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4056 & 0.0005 & -.0000 & 0.0000 & 0.0000 \\ -0.0005 & 0.4056 & -0.0000 & -0.0000 & 0.0000 \end{bmatrix}$		
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2155 & -0.0001 & 0.2153 \end{bmatrix}$		
The synchronized control (Actor NN) weights converge to:	$\mathbf{w^{*T}_{a,sync}} = \begin{bmatrix} -0.4056 & -0.0005 & 0.2157 & -0.0001 & 0.2155 \\ 0.0005 & -0.4056 & 0.2157 & -0.0001 & 0.2155 \end{bmatrix}$		
Agent 3	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2174 & -0.0007 \\ -0.0007 & 0.2240 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4277 & -0.0078 \\ 0.0078 & -0.4273 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} -0.0067 \\ 0.0049 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4042 & 0.0077 & -0.0000 & 0.0000 & 0.0000 \\ -0.0077 & 0.4045 & -0.0000 & -0.0000 & 0.0000 \end{bmatrix}$		
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2174 & -0.0007 & 0.2240 \end{bmatrix}$		
The synchronized control (Actor NN) weights converge to:	$\mathbf{w^{*T}_{a,sync}} = \begin{bmatrix} -0.4035 & -0.0057 & 0.2247 & 0.0009 & 0.2213 \\ 0.0056 & -0.4033 & 0.2247 & 0.2247 & 0.2213 \end{bmatrix}$		
Agent 4	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2105 & 0.0000 \\ 0.0000 & 0.2138 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4221 & -0.0027 \\ 0.0027 & -0.4219 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0038 \\ 0.0071 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.3982 & 0.0026 & -0.0000 & 0.0000 & 0.0000 \\ -0.0026 & 0.3984 & -0.0000 & -0.0000 & 0.0000 \end{bmatrix}$		

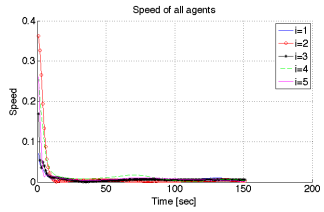
Table 4.4 Continued from previous page

Convergence Parameter	Optimal Solution Matrices		Optimal Control
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2105 & 0.0000 & 0.2138 \end{bmatrix}$		
The synchronized control (Actor NN) weights converge to:	$\mathbf{w_{a,sync}^{*T}} = \begin{bmatrix} -0.3982 & -0.0026 & 0.2107 & 0.0000 & 0.2140 \\ 0.0026 & -0.3984 & 0.2107 & 0.0000 & 0.2140 \end{bmatrix}$		
Agent 5	$\boldsymbol{P}^*_{\text{DARE}} = \begin{bmatrix} 0.2154 & 0.0007 \\ 0.0007 & 0.2147 \end{bmatrix}$	$\boldsymbol{L_k} = \begin{bmatrix} -0.4256 & -0.0009 \\ 0.0010 & -0.4256 \end{bmatrix}$	$\boldsymbol{u_k^*} = \begin{bmatrix} 0.0076 \\ 0.0028 \end{bmatrix}$
The control (Actor NN) weights converge to:	$\mathbf{w_a^{*T}} = \begin{bmatrix} 0.4020 & 0.0010 & 0.0000 & 0.0000 & 0.0000 \\ -0.0009 & 0.4020 & 0.0000 & -0.0000 & 0.0000 \end{bmatrix}$		
The value function (Critic NN) weights converge to:	$\mathbf{w_c^{*T}} = \begin{bmatrix} 0.0000 & 0.0000 & 0.2154 & 0.0007 & 0.2147 \end{bmatrix}$		
The synchronized control (Actor NN) weights converge to:	$\mathbf{w_{a,sync}^{*T}} = \begin{bmatrix} -0.4020 & -0.0010 & 0.2156 & 0.0007 & 0.2149 \\ 0.0009 & -0.4020 & 0.2156 & 0.0007 & 0.2149 \end{bmatrix}$		

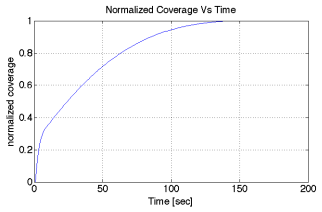
Similar to section 4.4.1, the optimal critic weights converges to the solution of the algebraic Riccati equation ($\mathbf{P}^*_{DARE}$ matrix). Also, the optimal feedback gain matrix, $\mathbf{L}^*_k$ computed using equation 4.34 is similar to that obtained from the synchronized actor weights, $\mathbf{W}^{*T}_{a,sync}$ using equation 4.35.



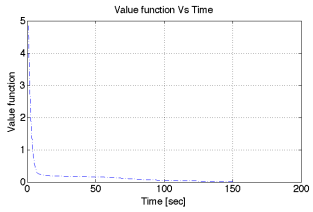
(a) Evolution of error trajectory



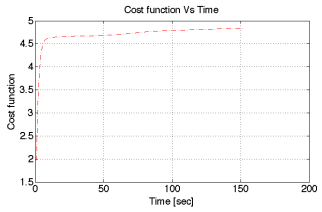
(b) Evolution of control signals trajectory



(c) Evolution of Normalized Coverage Metric



(d) Optimal Value Function



(e) Optimal Cost Function

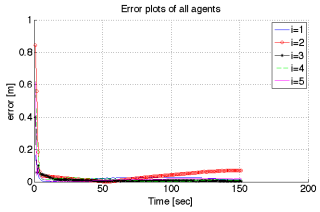
Figure 4.9: Multiple targets scenario with linear motion using MAACC-Rl NN: Coverage Plots

Trajectories of the Euclidean norm of the error variables between the agents and their centroids and trajectories of the control signal (speed) of all agents are shown in Figure 4.9-(a), and (b) with both graphs showing asymptotic convergence. Also, Figure 4.9(b) shows that

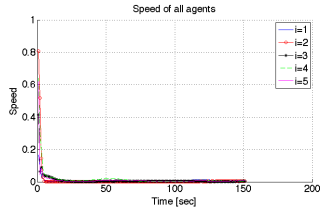
the proposed MAACC-RL NN controller regulates the states asymptotically to zero.The coverage metric plot shown in figure 4.9(c) demonstrate that all the agents are guaranteed to configure themselves optimally and the coverage is maximized as desired irrespective of the number of targets in the harbour. Figures 4.9-(d) and (e) show that the value and cost functions are optimized as expected. Also, areas with higher risk are densely populated by the agents.

4.4.4 Multiple targets scenario with linear motion using CORTES-CVT

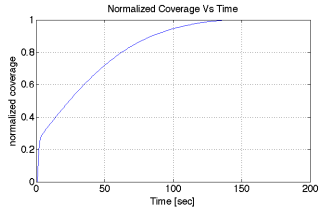
This section demonstrates the scenario with three mobile targets entering the harbour at the same time from initial positions (1.1, 0.1),(0.2, 2.4) and (2.2, 3.0) similar to section 4.4.3. The agents moves to neutralize the targets while protecting the harbour using the CORTES-CVT control law. In this case, the agents divide themselves into three groups based on closeness to the targets. Figure 4.10 shows the simulation graphs obtained.



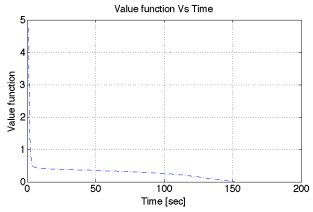
(a) Evolution of error trajectory



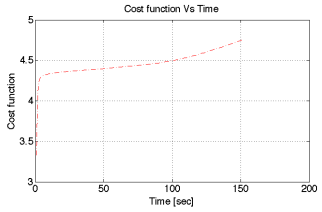
(b) Evolution of control signals trajectory



(c) Evolution of Normalized Coverage Metric



(d) Optimal Value Function



(e) Optimal Cost Function

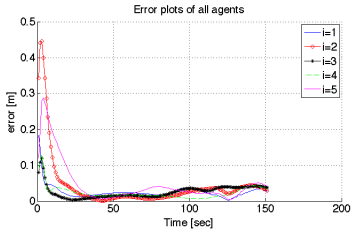
Figure 4.10: Multiple targets scenario with linear motion using CORTES-CVT: Coverage Plots

The agents converged faster under the multiple targets scenario using the proposed algorithm by comparing the performance measures shown in figure 4.9 with those in figure 4.10. The energy comparison is also shown later in this section.

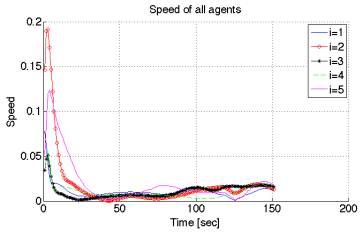
Next section presents results for neutralizing a **single** target modeled with an oscillating motion in the harbour.

4.4.5 Single target scenario with oscillatory motion using MAACC-RL NN

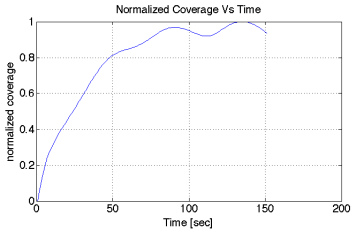
This section presents results of simulated scenario with a single moving target with oscillating motion using the proposed MAACC-RL NN algorithm to move the agents to optimal configuration. The target comes into the harbour from (0.05, 1.5) moving in an oscillatory manner and the agents are as usual expected to follow the targets and protect the harbour.



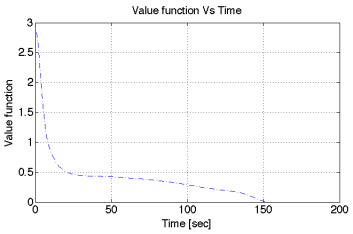
(a) Evolution of the error trajectory



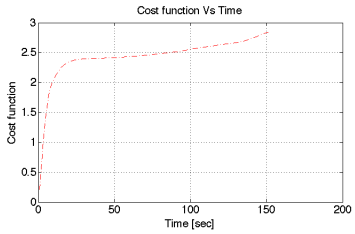
(b) Evolution of control signals trajectory



(c) Evolution of Normalized Coverage Metric



(d) Optimal Value Function



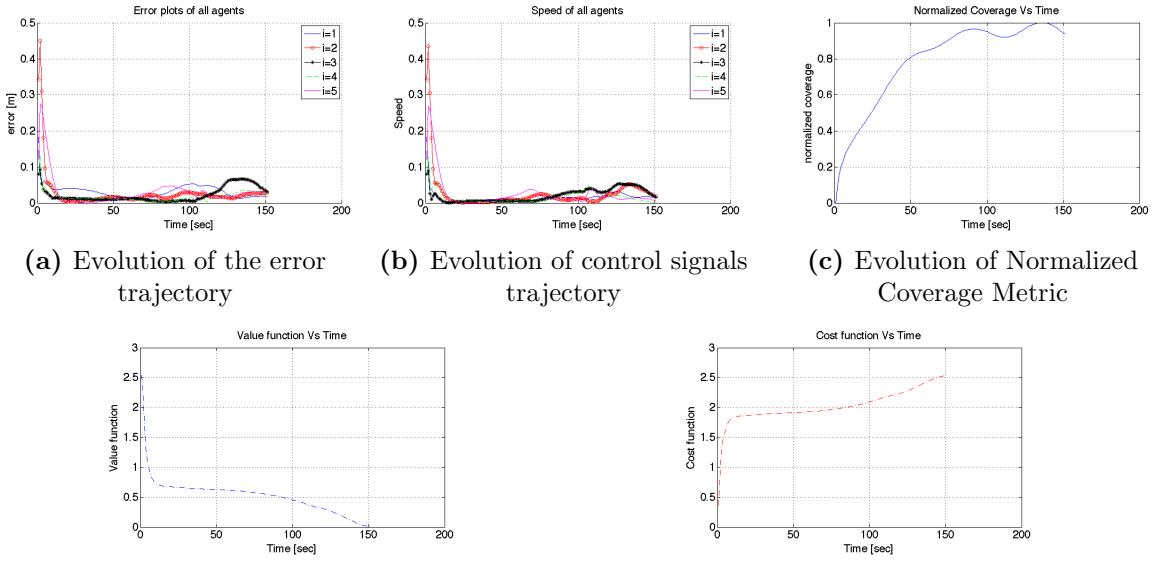
(e) Optimal Cost Function

Figure 4.11: Single target scenario with oscillatory motion using MAACC-RL NN: Coverage Plots

Trajectories of the Euclidean norm of the error variables between the agents and their centroids and trajectories of the control signal (speed) of all agents are plotted in figures 4.11-(a) and (b) with both graphs showing asymptotic convergence. the figure also shows the agents follows the oscillatory motion of the target. The coverage metric plot shown in figure 4.9(c) demonstrate that the agents are guaranteed to configure themselves in an optimal fashion as the target motion changes. Figures 4.10-(d) and (e) show that the value and cost functions are optimized as expected.

4.4.6 Single target scenario with oscillatory motion using CORTES-CVT

This section presents results of simulated scenario with a single moving target with oscillating motion similar to section 4.4.5 using the CORTES-CVT control law to move the agents to optimal configuration. The target comes into the harbour from (0.05, 1.5), moving in an oscillatory manner and the agents are expected to follow the targets and protect the harbour.



(a) Evolution of the error trajectory

(b) Evolution of control signals trajectory

(c) Evolution of Normalized Coverage Metric

(d) Optimal Value Function

(e) Optimal Cost Function

Figure 4.12: Single target scenario with oscillatory motion using CORTES-CVT: Coverage Plots

The error and control graphs shown in figure 4.11 shows smoother convergence as the agents follow the oscillating trajectory of the threat when compared to those in figure 4.12. The energy consumed during this task is compared in the next section.

4.4.7 Energy Consumption Comparison Analysis

We show that energy consumption by agents is influenced by several factors including: (i) Method type (a) policy iteration method (MAACC-RL PI) (b) actor-critic approximation method (MAACC-RL NN) (c) distributed approach for area coverage proposed in [15] (CORTES-CVT); (ii) Target Motion type (a) linear (b) oscillatory motion; and (iii) Number of targets (a)single target (b) multiple targets. Table 4.5 summarizes the results of the total energy consumed during the coverage task.

On comparison by method used, in the single target with linear motion scenario, simulation results show that CORTES-CVT consumed 169.9% more energy than MAACC-RL NN which also used 4.33% more energy than the MAACC-RL PI. Likewise, in the multiple targets(3 targets) scenario, CORTES-CVT consumed 192.11% more energy than the MAACC-RL NN and also used 3.19% more energy than MAACC-RL-PI. Interestingly, in the single target oscillatory motion scenario, the MAACC-RL PI used 48.67% more energy than the MAACC-RL NN while CORTES-CVT used 162.69% more energy than the MAACC-RL NN.

Comparing by motion type and number of targets, MAACC-RL PI consumed 33.29% more energy in the single target linear motion scenario than that used in the multiple targets linear motion scenario and 46.9% more energy than that used in the oscillatory motion scenario. Furthermore, the MAACC-RL NN algorithm used 34.78% more energy in the single

target linear motion scenario than the energy used in the multiple targets scenario and used 127.9% more energy than the oscillatory motion scenario. Lastly, the CORTES-CVT approach consumed 24.55% more energy during the single target linear motion scenario than during the multiple targets scenario and 134.2% more energy than the oscillatory motion scenario.

Table 4.5: Energy consumption simulation results by method type and motion type

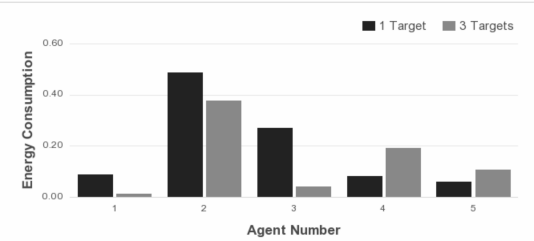
Motion Type	**MAACC-RL PI**	**Percent Change**	**MAACC-RL NN**	**Percent Change**	**CORTES-CVT**
Linear – 1 Target	0.9613	**4.33**%	1.003	**169.9**%	2.7077
Percent Change	**33.29**%		**34.78**%		**24.55**%
Linear – 3 Targets	0.7212	**3.19**%	0.7442	**192.11**%	2.1739
Oscillating – 1 Target	0.6543	**48.67**%	0.4401	**162.69**%	1.1561
Oscillating – 1 Target Vs. Linear – 1 Target **(Percent Change)**	**46.9**%		**127.9**%		**134.2**%

The figure 4.13 shows total energy comparison by method type and by motion type using a chart.

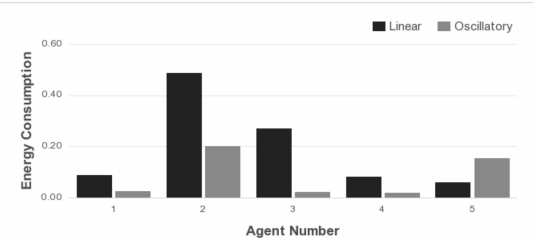
Figure 4.13: Total Energy Consumption By Method Type and Motion Type

The comparison between different methods and motion type are also presented in the

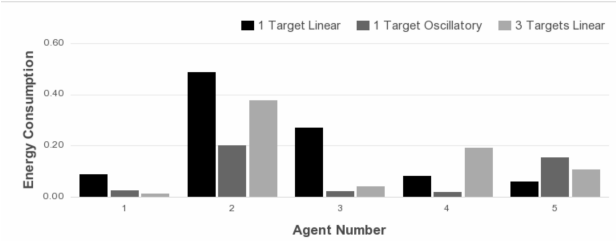
charts in figure 4.14.



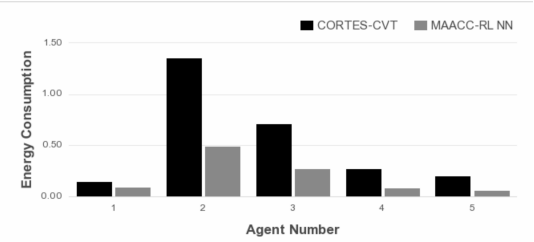
(a) Linear Motion Scenario



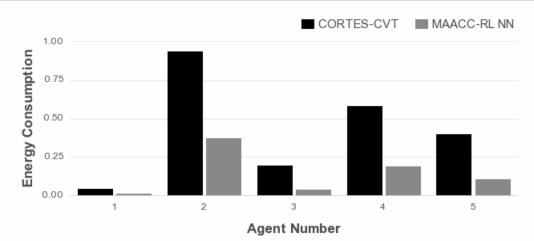
(b) Linear vs Oscillating Scenario



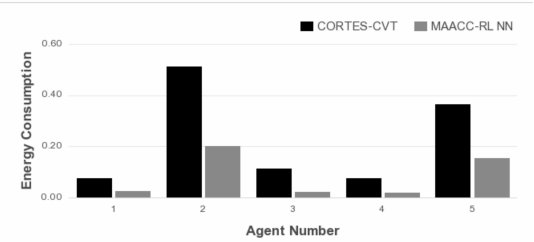
(c) All Motion Scenarios



(d) 1 target Linear Motion: CORTES-CVT Vs. MAACC-RL NN



(e) 3 targets Linear Motion: CORTES-CVT Vs. MAACC-RL NN



(f) 1 target Oscillatory Motion: CORTES-CVT Vs. MAACC-RL NN

Figure 4.14: Energy Consumption Charts

4.4.8 Summary and Discussion

In this section, an area coverage control law in cooperation with reinforcement learning techniques was proposed to solve the coverage control problem formulated as an optimal control problem in Chapter 3. The proposed MAACC-RL NN algorithm is able to asymptotically converge the agents to optimal configurations while protecting the harbour. The algorithm also solves the area coverage control problem regardless of the complexity of workspace risk density. Once the agents sense the existence of target(s), they compute their corresponding Voronoi cells such that more agents are deployed to areas with higher risk while providing optimal coverage with the underlying kinematic assumption that agents can generate velocities equal or larger than that of the targets, so that pursuit is possible. Due to the convexity of region, the centroid is always inside the Voronoi region. Therefore, the Voronoi approach has implicit collision avoidance between the agents and there is no possibility that two agents in the workspace will run into each other. However,

in a practical scenario, collision with the target is avoided since the agents are required to eliminate the target(s) at a defined safe distance. The agents maximize a defined coverage metric where the workspace is assumed to be free of obstacles indicating richness of coverage and they achieve centroidal Voronoi configuration using minimum actuator energy. In conclusion, simulations results also validated the theoretical framework and indeed exhibit the desired behaviours in the virtual environment as well as in theory.

Chapter 5

Conclusion and future work

The work in this thesis illustrates that reinforcement learning-based techniques can be successfully applied to coverage control. This section discusses the summary, theoretical challenges, conclusions, and future work.

5.1 Thesis Summary and Discussion

Motivated by the need to design an area coverage control solution in a distributed mobile actuator/sensor network, we proposed two approaches namely MAACC-RL PI and MAACC-RL NN that allow autonomous mobile agents to efficiently cooperate and coordinate their behaviour and thus achieve optimal deployment as described in this work.

Q1: *How to enable agents move and adjust quickly throughout the harbour so that they can aggregate in high risk areas to adequately defend against threats and achieve maximum coverage?*

In chapter 4, we answered research question **Q1**, by designing an adaptive controller that has the ability to retrain continuously online, thereby increasing it performance all the time. It also adjust to changing environment. The modeled time-varying risk density function is estimated by the agents. It takes into account the presence of risk in the harbour and it encodes two terms which is the biased density which shows priori risk in the domain indicating relative importance of different areas of the harbour and the second term denotes the presence of external threat. The proposed solutions robust, adaptable and scalable in that agents are sub-grouped based on the influence of target(s) in the harbour.
Q2: *How to organize agents to achieve efficient coordination while learning to cooperate based only on local interaction with each other and incomplete knowledge?*

In chapters 4, we answered research question **Q2**, by proposing a multi-agent area coverage control law in cooperation with reinforcement learning techniques for deploying multiple autonomous agents in the workspace. One significant advantage of reinforcement learning in control theory is the ability to retrain continuously and adapt to changes in the environment, thereby improving its performance all the time. This way, the learning

of the agents is dramatically improved while simultaneously providing area coverage and target surveillance.
Q3: *How to optimize agents information sharing with each other so that agents can simultaneously intercept a threat and maximize coverage while minimizing agents actuator energy?*

In chapter 3, we answered **Q3**, by showing a distributed optimal control approach for modeling energy-efficient cooperative area coverage control problem. An advantage of this is highlighted by the fact that agents do not run out of energy during a deployment mission. Using the methodology proposed in this research, agents are able to efficiently cooperate and coordinate their behaviour in a distributed manner and achieve their coverage mission objectives while minimizing agents actuator energy.

Q4: *How to model an agents dynamics, based on a mathematical formulation, so that an agent's optimal actions at each state, during an area coverage mission, are optimized?*

In Chapter 3, we answered research question **Q4**, by formulating the derivative of the error function which is used to study and model the behaviour of agents. As a result, this formulation is a major contribution in this thesis since it makes it possible to henceforth be able to study how individual agents, in a multi-agent system, can coordinate and cooperate with each other using an explicit mathematical systems model.

5.2 Thesis Conclusion

The proposed coverage control law in cooperation with reinforcement learning techniques, where multiple autonomous agents are able to asymptotically converge to optimal configurations while providing maximum coverage of a 2D environment was investigated. Partitioning of the workspace employed the well-known centroidal Voronoi tessellation technique. The proposed approaches were able to solve the coverage control problem regardless of the complexity of workspace risk density. In addition, the novel multi-agent area coverage control using solution framework developed in this thesis outperforms related work in area coverage by introducing a more robust and scalable distributed solution, especially suited for cooperative area coverage control problems.

5.3 Future Research Work

Although very encouraging results were obtained in this research and the four key research questions were answered in the simulation cases performed, more research questions have been triggered after completing this thesis. Ample opportunities exist to further extend and enhance the proposed approach. The following are suggestions for future research: We studied agent interactions and convergence times in dynamic networks characterized without communication delays. It is assumed that all communication between agents and

all sensing of agents locations is always accurate and instantaneous. However, communication delays are inherent in a mobile actuator/sensor network where agents constantly change their position while keeping track of their neighborhood changing positions by sharing local information throughout the entire network. Nevertheless, such delays can be remedied through the use of wireless/GPS sensor networks. Moreover, this research approach is adaptable if disturbance or noise is introduced in the environment. One needs to study the relationship between, for example, the convergence times of agents incorporating communication delays. In addition, we explored energy consumption involving cooperation and coordination between agents based only on local information.

Additionally, a potential future research avenue of the current work is to develop coverage control methods coupled with state estimation of each agents. This way, agents will be able to deploy themselves in the presence of stochastically intermittent communication network, which is not considered in this thesis. This research could also be extended to the 3D case to develop a strategy for area coverage and surveillance of urban-like environments which is not considered in this thesis.

Finally, autonomous agent deployment in a mobile actuator/sensor network is without a doubt a challenging undertaking. However, many more distributed systems exist where autonomous agents need to cooperate and coordinate in order to solve their unrelenting design and operational objectives. We believe the research presented in this thesis provides the tools and methodology with which to study real life applications of distributed multi-agent systems in cooperation with reinforcement learning.

References

[1] Sung Lee. Controlled coverage using time-varying density functions. In Daniel Vey, editor, *4th IFAC Workshop on Distributed Estimation and Control in Networked Systems (2013)*. Elsevier BV, sep 2013.

[2] Diaz-Mercado Lee and Egerstedt. Multirobot control using time-varying density functions. *Robotics, IEEE Transactions*, 31(2):489–493, 2015.

[3] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 243–255. IEEE, 2002.

[4] Yu Jiang and Zhong-Ping Jiang. Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica*, 48(10):2699–2704, 2012.

[5] Y. Gu. "introduction to robot planning". Lecture note, West Virginia University.

[6] Stuart P Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 2006.

[7] Enrico Simetti, Alessio Turetta, Giuseppe Casalino, and Matteo Cresta. Towards the use of a team of usvs for civilian harbour protection: The problem of intercepting detected menaces. In *OCEANS 2010 IEEE-Sydney*, pages 1–7. IEEE, 2010.

[8] Zygmunt Kitowski. Architecture of the control system of an unmanned surface vehicle in the process of harbour protection. In *Solid State Phenomena*, volume 180, pages 20–26. Trans Tech Publ, 2012.

[9] Suruz Miah, Bao Nguyen, François-Alex Bourque, and Davide Spinello. Nonuniform deployment of autonomous agents in harbor-like environments. *Unmanned Systems*, 2(04):377–389, 2014.

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

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

[12] Jinwen Hu, Lihua Xie, Kai-Yew Lum, and Jun Xu. Multiagent information fusion and cooperative control in target search. *Control Systems Technology, IEEE Transactions on*, 21(4):1223–1235, 2013.

[13] Mohamad K Allouche and Abdeslem Boukhtouta. Multi-agent coordination by temporal plan fusion: Application to combat search and rescue. *Information Fusion*, 11(3):220–232, 2010.

[14] Davide Spinello and Daniel J Stilwell. Distributed full-state observers with limited communication and application to cooperative target localization. *Journal of Dynamic Systems, Measurement, and Control*, 136(3):031022, 2014.

[15] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[16] Francesco Bullo, Jorge Cortés, and Sonia Martinez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.

[17] Sonia Martinez, Jorge Cortes, and Francesco Bullo. Motion coordination with distributed information. *Control Systems, IEEE*, 27(4):75–88, 2007.

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

[19] S. G. Lee and M. Egerstedt. Multi-Robot Control Using Time-Varying Density Functions. *ArXiv e-prints*, April 2014.

[20] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2000.

[21] Asma Al-Tamimi, Frank L Lewis, and Murad Abu-Khalaf. Discrete-time nonlinear hjb solution using approximate dynamic programming: convergence proof. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(4):943–949, 2008.

[22] Vamvoudakis and Lewis. Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem. pages 3180–3187, 2009.

[23] Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3):233–246, 2002.

[24] Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

[25] M. Schwager, F. Bullo, D. Skelly, and D. Rus. A ladybug exploration strategy for distributed adaptive coverage control. In *Proc. of the International Conference on Robotics an Automation (ICRA 08)*, pages 2346–2353, May 19–23 2008.

[26] Luciano CA Pimenta, Mac Schwager, Quentin Lindsey, Vijay Kumar, Daniela Rus, Renato C Mesquita, and Guilherme AS Pereira. Simultaneous coverage and tracking (scat) of moving targets with robot networks. In *Algorithmic Foundation of Robotics VIII*, pages 85–99. Springer, 2008.

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[28] Draguna L. Vrabie Frank L. Lewis and Vassilis L. Syrmos. *Optimal control*. John Wiley and Sons, Inc., New Jersey, 2012.

[29] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, (5):834–846, 1983.

[30] Steven J Bradtke, B Erik Ydstie, and Andrew G Barto. Adaptive linear quadratic control using policy iteration. In *American Control Conference, 1994*, volume 3, pages 3475–3479. IEEE, 1994.

[31] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[32] Lendaris Murray, Cox and Saeks. Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Cybernetics, IEEE Transactions on*, 32(2):140–153, 2002.

[33] Murad Abu-Khalaf and Frank L Lewis. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach. *Automatica*, 41(5):779–791, 2005.

[34] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[35] Draguna Vrabie, O Pastravanu, Murad Abu-Khalaf, and Frank L Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, 45(2):477–484, 2009.

[36] Silvia Ferrari and Robert F Stengel. An adaptive critic global controller. In *American Control Conference, 2002. Proceedings of the 2002*, volume 4, pages 2665–2670. IEEE, 2002.

[37] Kyriakos G Vamvoudakis and Frank L Lewis. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888, 2010.

[38] Frank L Lewis and Draguna Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE*, 9(3):32–50, 2009.

[39] Travis Dierks and Sarangapani Jagannathan. Optimal control of affine nonlinear continuous-time systems using an online hamilton-jacobi-isaacs formulation. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3048–3053. IEEE, 2010.

[40] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[41] Park Lee and Choi. Integral reinforcement learning for continuous-time input-affine nonlinear systems with simultaneous invariant explorations. *Neural Networks and Learning Systems, IEEE Transactions on*, 2014.

[42] Miad Moarref and Luis Rodrigues. An optimal control approach to decentralized energy-efficient coverage problems. In *World Congress*, volume 19, pages 6038–6043, 2014.

[43] David Kleinman. On an iterative technique for riccati equation computations. *Automatic Control, IEEE Transactions on*, 13(1):114–115, 1968.