Optimal Multi-Robot Exploration For Static Environment

A DISSERTATION

Submitted in partial fulfillment of the

requirements for the award of the degree

of

Master of Technology

in

ELECTRICAL ENGINEERING

(With specialization in Systems and Control)

By

ABHIJEET MAZUMDAR

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE - 247 667 (INDIA)

May 2016

Candidate's Declaration

I hereby declare that this thesis which is being presented as the final evaluation of the dissertation **Optimal Multi-Robot Exploration For Static Environment** in partial fulfilment of the requirement of award of Degree of Master of Technology in Electrical Engineering with specialization in Instrumentation and Signal processing, submitted to the Department of Electrical Engineering, Indian Institute of Technology, Roorkee , India is an authentic record of the work carried out during a period from May 2015 to May 2016 under the joint supervision of Prof. Dr. G. N. Pillai (Department of Electrical Engineering, IIT Roorkee) and Prof. Dr.-Ing. Jorg Raisch (Control Systems, TU-Berlin, Germany). The matter presented in this report has not been submitted by me for the award of any other degree of this institute or any other institute.

Date:
Place: (Abhijeet Mazumdar)

Certificate

This is to certify that the above statement made by the candidate is correct to best of my knowledge.

Date:
Place:

Prof. Dr. G. N. Pillai
Professor
Department of Electrical Engineering
Indian Institute of Technology, Roorkee

Acknowledgements

This dissertation was an integral part of me for the last few months. I want to take this opportunity to thank all of them without whom this thesis could not be finished though this short acknowledgement is not enough to match their favours.

I am greatly indebted to my supervisors Prof. Dr. G. N. Pillai and Prof. Dr.-Ing. Jörg Raisch (Control Systems, TU-Berlin, Germany) for giving this wonderful opportunity to work under their joint supervision. The encouragement that I got from Prof. Pillai boosted my confidence and enthusiasm. Prof. Raisch was a great source of inspiration for me. The conversations with him always used to inject a lot of positive vibes within me.

My sincere gratitude goes to all the faculty members of Systems and Control, Department of Electrical Engineering, IIT Roorkee for their encouragements and help.

I thank DAAD for funding my research in Germany. This project was only possible because of DAAD. I am extremely happy that I was able to work in Computer Vision and Remote Sensing group of TU-Berlin in such a friendly yet sincere environment; I thank all my colleagues and friends back there.

I thank TU-Berlin International Relations Office (Akademisches Auslandsamt), who made my stay in Berlin a lot more smoother and enjoyable than what it turned out to be without them.

I feel fortunate to get my family and friends by my side, who were always a great source of inspiration for me. My parents were always great pillars of strength who have allways encouraged me to reach for my dreams, and I want to dedicate this work to them.

Begging pardon of all those well-wishers whose name could not be taken.

ABHIJEET MAZUMDAR

ABSTRACT

Persistent surveillance or exploration of any static environment requires the agent to cover the entire mission space in a fixed amount of time. In this thesis, a similar problem is addressed by deploying multiple agents and controlling their movement and direction by parameterizing their trajectories. It has been proven that in a one dimensional space, the best solution is to move the agent at maximum speed in a direction and then switch directions when points of interest are reached, after collecting information from those points. But in two dimensional spaces, such conclusions can no longer be drawn. In this thesis, the agent trajectories are represented by a parametric function which can be optimized. The points are associated with a time-varying uncertainty function which increases if the points are not within the sensing range of the agent. First, a single agent is considered and its trajectory is optimized by using different cost functions and initial conditions. Infinitesimal Perturbation Analysis(IPA) is used to calculate the cost function with respect to the trajectory parameters. A major part of this thesis is devoted to find an appropriate cost function which solves the persistent surveillance problem. This thesis also concentrates on providing a solution for obstacle avoidance. The problem considered here is highly non-convex and therefore global optimizing techniques must be used. Stochastic Comparison Algorithm is used to find a global optimal solution. The simulation results shows the comparison between all the methods used.

Contents

List of Figures

Chapter 1

Introduction

Recently there has been a wide technological advancement in the field of persistent surveillance of autonomous agents which are able to co-ordinate and co-operate with each other to perform various complex tasks due to the growing interest in applying autonomous vehicles for search and rescue operations. The wide diversity of the tasks and the mission space that the agents operate in imposes numerous requirements on their motion, control, sensing and actuation systems. To ensure that the robotic team performs efficient operation, it is necessary that the subtask allocation, trajectory planning, navigation, sensor integration and communication are adequate.

Persistent surveillance differs from any traditional surveillance tasks due to the need to monitor a dynamic environment and constantly adapt the agents path to perform required tasks. Examples of such monitoring tasks include basic area surveillance, patrol missions with automated unmanned vehicles, search and rescue operations and various applications in environment where an areas routine sampling is necessary. The entire area of the mission space should be monitored infinitely often which cannot be monitored by a network of stationary agents. The main difficulty which arises in the design of control strategies is balancing coverage of all the agents in the constantly changing environment by distributing the exploration effort among them while taking into account their individual dynamic capabilities and to ensure that the solution is time optimal even after satisfying all the motion and sensing constraints of the system.

1.1 Related work

There has been a large amount of progress in the field of robotics on coverage control [1] and [2], trajectory optimization [**?**] and persistent surveillance tasks [3] [4] [5]. A simple discrete adaptive guidance system for a roving vehicle is described in [6]. It describes a rover with self-correcting path following capabilities for planetary explorations. The rover executes a pre-planned trajectory with a computer controlled autonomous guidance system in an unknown environment. Taking into account the obstacles of the terrain, the rover auto-corrects its path so as to match the planned trajectory as close as possible using a visual sensor. It calculates the angular dynamics of the path using the images to manoeuvre around any obstacle or trench. Similarly, the methods of trajectory optimization of a launch vehicle is discussed in [7] and [8]. The former provides insight on the practical method of using gradient approach [9] on the cost function and the adjoint variables to control the perturbations in the launch path. The latter describes the specific control for an ATLAS/CENTAUR flight using closed loop guidance in the exo-atmospheric phase.

In [1] and [2], the basic idea of coverage control and data collection is described in which the whole task is divided into three sub tasks namely coverage control, data source detection and data collection.The agents have to locate data sources and gather the information from them when they are in their close surroundings. In [10] coverage control was successfully applied to co-operative air and ground surveillance in which the unmanned aerial vehicles and ground vehicles are utilised to search for a finite number of targets. In [3], the idea of persistent surveillance using multiple unmanned aerial vehicles which work in harmony to successfully survey the whole mission space. In [11], a decentralized policy is developed in which the information gathering is divided into different agents that have to individually monitor their own sub-area allotted to them in an unknown environment. The agents estimate the location of their own target and sub-allot themselves to the area around it. [12] describes a coordination policy for multiple robots in which non-parametric methods are explored to control them distributively. The main aim is to collectively gather information in a decentralized manner where individual robots have their own environment state and weighted sample set. In [13], the concept of information decay is further added to [11] and the agents now have to monitor an environment where information is a time-variant function which decays eventually. This idea adds more definition to the cost function due to the automatic optimization of time while optimizing decay.

The task of control and motion planning for agents performing surveillance has been well studied in different frameworks [4] [14]. In [5], the concept of persistent surveillance in a changing environment with robots having limited sensing range was introduced. The changing environment is described as a growing function in those areas where the robot cannot reach. Therefore, the robot has to cover the whole area in the space so as to effectively sweep and monitor the change. Similarly in [15], the robots have to sweep a constantly changing environment where the environment is not a growing function but a time-changing Gaussian Random field. The main aim is to find the minimum infinite horizon cost cycle. The concept of iterative polynomial-based mobile robot trajectory extension was introduced in [16]. The mobile robot can effectively extend its path to adapt to the changes in command or the target and hence can perform in a changing environment. [17] introduces dynamic programming to optimize the non-linear trajectories of the robot. It is a very powerful method which divides the total problem into various sub-problems reducing the computational complexity. The results are stored and when the particular sub-problem arises again, the algorithm just uses the old solution instead of finding the same again.

The development of new optimization methods has led to the wide use of optimal control in controlling robotic arms and mobile agent trajectories. [18] and [19] introduce the use of optimization techniques to control unmanned aerial vehicles and quadcoptors in which the persistent surveillance problem is solved. The UAVs can sweep the area with different level of interests to prioritize the sub-areas. In [14] motion planning and controlling of the agents which perform persistent surveillance tasks are studied. In particular we see that a framework of optimal control is designed to obtain controls for multiple agents to address the persistent surveillance problem so as to minimize a parameter of uncertainty in the mission space. The concept of parameterizing the agent trajectory and adjusting the parameters to optimize the trajectory is taken from [16]. The mobile robots in all the previous cases are considered as point particles which cannot collide. But in real time, all the robots can collide with each other and therefore collision avoidance is also one of the major problems in any multi-robot environment. In [20] and [21], the problem of collision avoidance of multiple agents is taken into consideration. Neural network approach is used in [20] to plan a real-time collision-free path of robots. In [21], industrial robots are collectively controlled such that they do not collide with one another. The collision avoidance improves the efficiency of the whole multi-robot exploration.

For easier computation, there are many assumptions made in a system and therefore we obtain the optimal solution for specific cases where some parameters are fixed. If we take into account all the system parameters and all the cases under which they work, then almost every system can be represented as a hybrid system. [22] introduces hybrid systems and the basic methods to control any hybrid system which contains both continuous and discrete dynamics and therefore are complex. [23] and [24] introduce modelling and analysis of different hybrid systems with many examples. A model checker is introduced in [25] to model any hybrid system. A sliding mode control approach to control such systems is explained in [26]. The solution of the optimal multi agent persistent problem addressed in [27] defines a new approach in which the agent's path is defined by certain parameters which are updated by the gradient of the cost function. The system here is considered as a hybrid system and is modelled accordingly using various IPA algorithms [28].

1.2 Problem Statement

The main objective is to address the multi robot exploration problem by local trajectory optimization in two dimensional spaces. The robotic team has to cooperatively minimize the expected cost for detecting targets in a limited two-dimensional static environment. The robots have second order continuous dynamics and limited sensor range around their current position. Each robot is provided with an initial trajectory characterized by a finite number of parameters. The problem of determining optimal parameters for these trajectories will then be solved by a gradient analysis of the cost function with respect to the parameters. Then, the method will be extended for the case when the real target probability distribution or some obstacles in the environment are a-priori unknown, which requires an online adaptation of the robots planned trajectory. The method will be compared with alternative optimization approaches in a numerical search and rescue scenario. The agents have to detect and observe several points spread over a static two dimensional space over a specific period of time and reduce the overall cost.

1.3 Contributions

This thesis covers the solution of the problem stated by adopting optimal control policies and uncertainty parameters. The first contribution is the reduction of the computation time and also the time taken to sweep the whole mission space which is very low when compared to a discrete analysis of the problem using methods such as dynamic programming. The trajectory undertaken by the agent is very flexible and the parameters can be changed while in motion which makes it efficient and reduce cost further. The second contribution will be to represent the agent's path in form of function families which are parameterized and can be controlled and moulded effectively by changing their parameters. The third contribution is an efficient method for task scheduling and allocation for a robotic team which could be applied to various real world problems such as rescue missions and military surveillance and security. It can also be used in the supply situation to constantly supply any number of things. The fourth contribution is the obstacle handling capabilities which are discussed in this thesis where different agents monitor the area considering various obstacles in their path.

1.4 Outline

This thesis consists of six chapters. The first chapter is the introduction where the problem is introduced and all the previous works are stated and discussed. There are some contributions listed and the outline of the thesis is presented. The second chapter covers all the necessary theory and some derivations which are used to obtain the solutions. This chapter also includes some modifications and assumptions which were made to draw the conclusion. The third chapter covers the methodology undertaken to obtain the optimal solution of the problem. It covers the procedure and the solution proposed to solve the non-convex problem thereby reducing the cost. The fourth chapter covers the procedure and steps of the stochastic comparison continuous algorithm to achieve global optimality. The fifth chapter shows the pseudo codes and results obtained and compares the different methods with respect to the cost and efficiency. The sixth chapter covers the discussion and conclusion as well as the further possibilities of this thesis.

Chapter 2

Theory and Preliminaries

2.1 Preliminaries

2.1.1 Assumptions

The problem considered in this thesis is highly non-convex and has a lot of computational difficulties. And therefore, to solve the required multi-robot persistent surveillance problem, some assumptions are made:

- All the agents are considered as point particles, i.e. they cannot collide with one another.
- The agent velocity $u_n(t)$ is a bounded variable i.e. $u_n(t) \in [0, 1]$.
- Each agent can control its own orientation and speed.
- The agents have limited sensing range ($r_n = 1$).
- The Fourier series considered is of first order.

2.1.2 Notations used

M	number of points in the mission space
N	number of agents
$s_n(t)$	agent trajectory $n \in [1, ..., N]$
$u_n(t)$	scalar velocity of nth agent
r_n	sensing radius of nth agent
$\rho(t)$	function controlling agent position
σ_n	fourier series parameters
$P_m(s(t))$	joint probability f detecting any point
$Q_m(t)$	uncertainty of any point $[a_m, b_m]$ L
A	rate of increment of uncertainty.
B	rate of decrement of uncertainty.
$J()$	cost function
$D(w_m, s_n)$	distance between the agent and any point $n \in [1, ..., N]$ and $m \in [1, ..., M]$.
η	updation parameter

2.2 Basic Concepts

2.2.1 Hybrid Systems

Hybrid systems in general consist of the systems which are heterogeneous in nature or in composition and whose behaviour are defined by either processes or entities of very distinct characteristics. In this thesis, the term hybrid refers to the involvement of both continuous and discrete dynamic behaviour of the system. A hybrid system has the ability of encompassing a very large class of systems in its structure which provides more flexibility in dynamic phenomena modelling. Therefore these systems generally deal with equations which contain mixtures of discrete valued and logic (digital dynamics) and analog or continuous variables. The continuous dynamics of these systems are generally given by differential equations of second order or more whereas the discrete dynamics are given by a hybrid automata or by different countable states of the system.

Figure 1 describes a hybrid system in which the output depends on a trigger or an event which generally happens when a controlled variable hits certain boundaries or certain

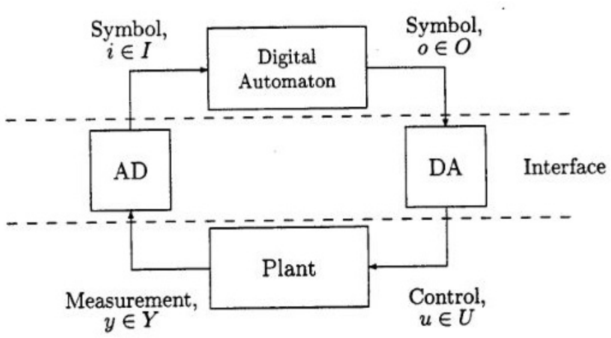

Figure 2.1: A Hybrid System

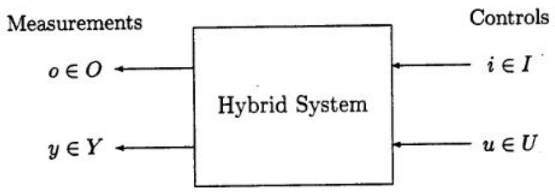

Figure 2.2: A Hybrid Control System

rules. To control such system we require certain measurements and discrete control rules and hence we obtain a system depicted in figure 2.

2.2.1.1 Studies in Hybrid Systems

The general research in hybrid systems are carried out under four different categories in which every category is as important as the other. Modelling consists of recognising and determining a hybrid system and to successfully model the system in terms of blocks or mathematical approach. This category captures the rich behaviour of the hybrid system and its sub-components. Analysisinvolves scrutinizing the issues concerning the hybrid systems and specific simulation and computational abilities of different analog, digital and hybrid systems. Then, the tools are developed for the verification and analysis of the hybrid systems.

The control category consists of developing different methods to control a hybrid system. Various hybrid controllers are synthesised which can issue continuous control as well as work for discrete decision making with respect to certain conditions and safety measures.The final step consists of designing the whole setup for a hybrid system considering all the inputs and outputs as well as finding new structures and schemes which will eventually lead to easier modelling, control and verification of the hybrid system.

2.2.1.2 Examples

As we know that a hybrid system is a system which has both continuous and discrete elements, the control action should involve the mixture of logical decision making and generation of continuous control laws. A very general and simple example of a hybrid system will be the hysteresis function itself. The switching condition of the system changes with the increase or decrease of the values of the function.

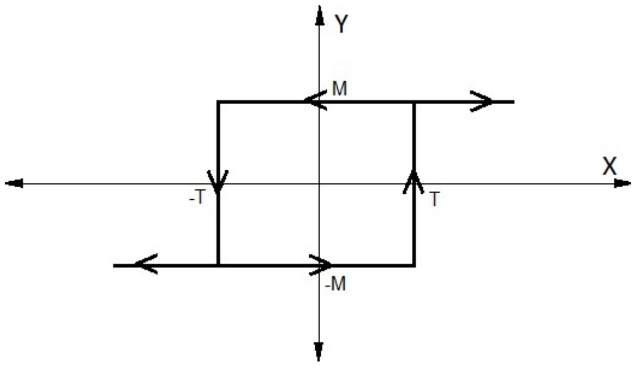

FIGURE 2.3: Hysteresis Function

The above hysteresis function can be modelled as a hybrid function which can be represented as follows:

$x = f(x, H(x - x_0))$, where x and x_0 denote the present and desired value respectively. This hybrid system is not piece-wise continuous and it has a memory of itself, and therefore we require a hybrid automaton to describe it.

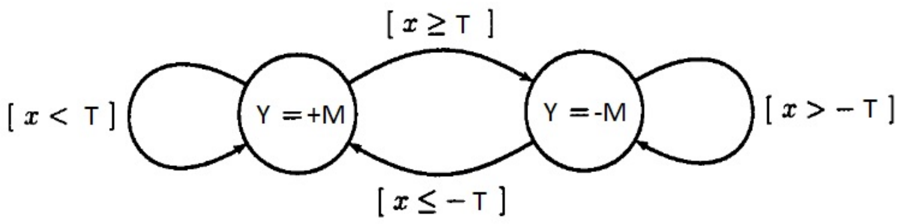

FIGURE 2.4: Hybrid Automaton

The above automaton describes the hybrid behaviour of the hysteresis function where the variable x is a continuously changing variable and depending on its value $(> Tor < -T)$, the term y undergoes a discrete transformation between M and M. More such examples would be simple pendulum, systems with relays and switches, computer disk drives, constrained robotic systems and highway systems.

There are different discrete phenomena that occur within a hybrid system they are basically the type of jumps that the system undergoes. They are divided into four categories:

- **Autonomous Switching**: It is the phenomenon of a hybrid system where the output changes discontinuously when the continuous input hits certain boundaries.
- **Autonomous jumps**: This is the phenomenon in which the continuous input of the hybrid system changes discontinuously when it hits certain boundaries.
- **Controlled switching**: It is the phenomenon in which the output changes abruptly to a specific value when a control signal is given to the input usually to reduce the cost.
- **Controlled jumps**: It is the phenomenon in which the continuous input state changes discontinuously in response to a control instruction, to reduce the overall cost.

Considering all the above phenomenon, a general hybrid system can be dynamically represented as:

$H = [D, S, J, M]$, where the individual constituents are represented as follows:

- D represents the set of discrete states that the system is involved with.
- S represents the continuous dynamics of the system $(S_n = [x_n, f(x_n)])$.
- J represents a collection of autonomous jump sets.
- M represents the collection of autonomous jump transition maps which represents the discrete dynamics of the hybrid system.

In this thesis, hybrid behaviour is observed in the uncertainty analysis of all the points of interest in the mission space which will be discussed and analysed further and the respective control action will be taken.

2.2.2 Optimal Control

Optimal control theory is a branch of control system that deals with the dynamics of the system mostly in terms of differential equations of the various states of the system. It is a mature discipline of mathematics which has numerous applications both in science and engineering. In this type of control, the inputs and outputs are generally represented by some functions of time and there are various states which determine the input output characteristics.

The very basic aim of any optimal control system is to minimize or maximize any cost function on which the entire system's efficiency depends on. Any system which can be represented in the state variable form can be optimally controlled. So in mathematical terms, any optimal control problem can be defined as:

$$\min_{x,u} J\left(x,u\right), \;\; Given,\; \dot{x} = f\left(x,u\right), \;\; for\ all\ x \in X\ and\ u \in U$$

The general approach in an optimal control is to find out the smallest possible variation of the functional which is to be optimised and then equating that variation to zero such that the resulting value of the input and the parameters may lead to a solution where there is no variation of the functional i.e. any minimum or maximum point where the slope is zero. This means the optimal minimised solution for any function $f(t)$ at any time τ will be $f^*(\tau,\ u^*(\tau))$ if for all $t\ \in\ [0,\ T]\,,\ f^*\left(t, u^*\left(t\right)\right) \leq f(t,u(t))$.

There are various approaches within optimal control which are used to find out the optimal solution like the direct method, Lagrange multiplier, Hamiltonian approach, etc. Here we use the Pontryagin minimum principle and the Euler-Lagrange equations to determine the optimal cost using a Hamiltonian approach. Generally a cost function minimisation system has a fixed starting point and a free ending point, i.e. $x(t_0)$ is fixed and $x(t_f)$ is free. The concept is shown in the following figure:

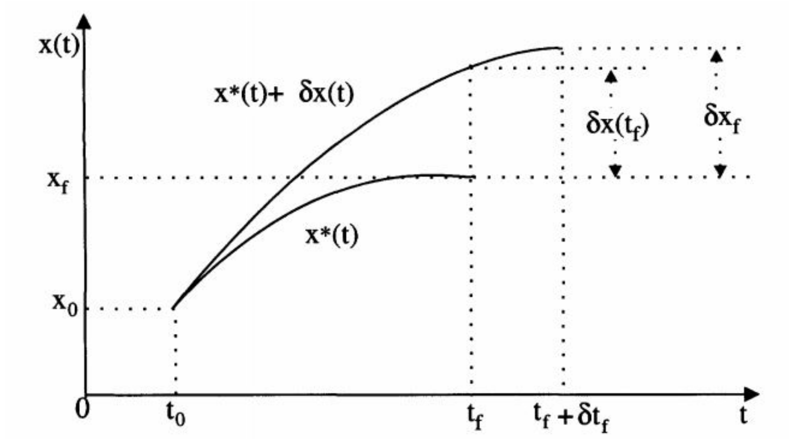

Figure 2.5: Free end point and time system

2.2.3 Local Optimization

Local optimization is a method of finding the locally optimum value for any function i.e. to maximize or minimize the function within a specified range of input values. It is called local optimal value because it does not guarantee the best solution or the 'global' solution for any function or system. Gradient descent is a widely used local optimization method. The gradient descent algorithm uses the slope of the cost function to determine the optimal solution. The main aim of this optimization technique is to converge to a local optimum. The gradient of the function at any point is taken. Then, **the solution is stepped in the negative direction of the gradient** and the process is repeated. The algorithm will eventually converge to a local optimum where the gradient is zero (which corresponds to a local minimum). Its counterpart, the gradient ascent, finds the local maximum by stepping it towards the positive direction of the gradient. They are both first-order algorithms because they take only the first derivative of the function.

If $f(x)$ is the function which has to be locally optimized, an initial value x_0 is selected and we have$f(x_0)$. Depending on the requirement, we can either find the local minimum or maximum by finding the gradient $\nabla f(x_0)$. Intuitively, the gradient provides the slope of the function at the initial point and its direction will point to an increase in the function. And therefore we have to change the value of the input so as to decrease the value of the function which would be in the negative of the gradient. $x_{k+1} = x_k - \lambda \nabla f(x_k)$

The term $'\lambda'$ ensures that the algorithm makes very small changes to the variable such that it doesn't cross the local optimum. Given stable conditions (a certain choice of λ), it is guaranteed that $f(x_{k+1}) \leq f(x_k)$.

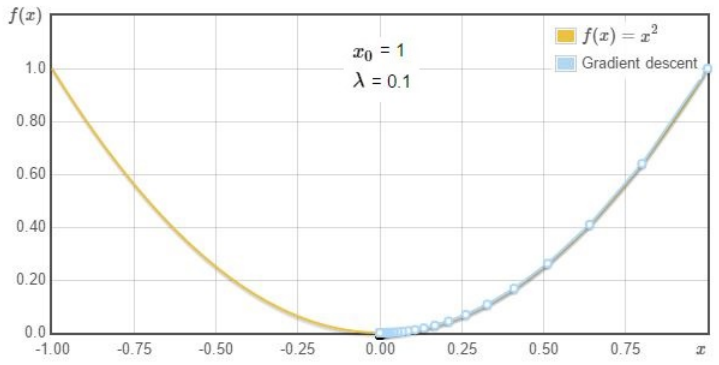

FIGURE 2.6: Gradient Descent

The above plot shows the results of the gradient descent method applied to a simple square function. We can observe form the plot how the method finds the local optimum of the function ($f(x) = 0$). Each point depicts the function value after every iteration of the gradient descent with a fixed value of λ as 0.1. There are some other methods in which the value of λ also changes after every iteration so as to make the descent smoother.

2.2.4 Convex and Non-Convex Optimization

A convex optimization problem maintains the properties of a linear programming problem which are easy to compute whereas a non convex problem maintains the properties of a non-linear programming problem. In a convex optimizing problem, all the constraints are convex functions which usually converge to a particular global optimum. Many examples and control techniques of such systems are described in [29] All the linear and conic functions are convex in nature and usually converge to an optimum or prove that they have no solution. A simple convex function is shown below:

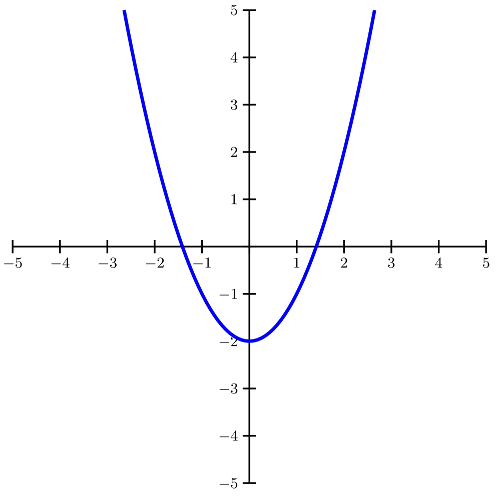

Figure 2.7: A Convex Function - Parabola

A nonconvex optimization problem may have multiple locally optimal points and it can take a lot of time to identify whether the problem has no solution or if the solution is global. A non-convex function is neither convex nor concave i.e. which has lots of ups and downs. A simple example will be a sine function.

The basic difference between the two categories is that in convex optimization problem there can be only one optimal solution, which is globally optimal although there might be cases in which there is no feasible solution to the problem. Hence, the efficiency in time of the convex optimization problem is much better. And therefore, convex problem usually is much easier to deal with in comparison to a non-convex problem.

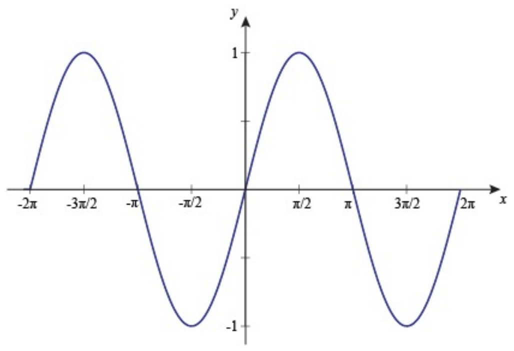

FIGURE 2.8: A Non-Convex Function - Sine Function

Non-convex optimization involves various steps to solve the problem. Formulating the problem in mathematical terms is very important. Recently, many different techniques are used to solve a non-convex problem such as simulated annealing and other evolutionary algorithms along with other heuristics which lead to local solution. The methods such as dynamic programming, mixed integer programming and interval methods are used to probably converge to a globally optimal solution.

2.2.5 IPA Algorithm

Infinitesimal Perturbation Analysis (IPA) is a method which calculates the gradient of any cost function. It has a unique property of finding out the gradient of any performance measure with respect to many variables. The finite differencing method requires the user to differentiate twice, once with x and again with $(x + \Delta x)$, for a small value of Δx, to find the gradient with respect to single parameter. If the cost function is $J(w, x)$, the finite differential will be $\frac{1}{\Delta x}[J(w_1, x + \Delta x) - J(w_2, x)]$.

The viewpoint in which the Perturbation Analysis differs from other conventional methods is that it considers that there is more knowledge about the system in the output analysis of a single experiment run on any Discrete Event Dynamic System(DEDS). The method which perturbation analysis uses is to rebuild a perturbed sample path from the initial path with very slight change in parameters. Perturbation algorithms generally consist of two paths: generation and propagation. Any path in a discrete event dynamic system, the IPA algorithm introduces imaginary perturbations in the event times while calculating the gradient with respect to a single parameter. . It is more efficient because in the end, the IPA algorithm assumes that the event changes in time in the initial and the perturbed path ae the same.

The IPA algorithm consists of the following events:

1. Exogenous events. Any event is called as exogenous if a discrete change has occurred in any variable at an event time t_k which does not depend on the control vector σ which should satisfy $\frac{\partial t_k}{\partial \sigma}$. These events are usually the events which have uncontrolled input process changes .

2. Endogenous events. Any event is called as endogenous if at any time instant t_k, the event has a continuously differentiable function f_k such that $t_k = min[t > t_{k1} : g_k(x(\sigma, t), \sigma) = 0]$

3. Induced events. An event at time t_k is induced if it is triggered by the occurrence of another event at time $t_m \leq t_k$. The triggering can be by any of the above three events.

2.2.6 Stochastic Comparision Algorithm

Stochastic optimization is an area of much importance and is the one that provides huge challenge in theoretical as well as practical point of analysis. The discrete event systems have given rise to various problems including the analysis and design of complex stochastic systems where the function to be optimized often cannot be represented in a closed form and hence, there has to be a direct observation from the actual data to optimise the function. Stochastic optimization can generally be divided into two categories i.e. discrete and continuous optimization. In this thesis we consider the continuous optimization where the objective function depends on a continuous parameter vector. One common approach should be the gradient estimator which drives the whole process towards a minimum point but the main problem is that the gradient approach may easily lead a system to a local minima if there are multiple local minima in a system. To overcome the problem of settling down in a local minima, it is absolutely necessary for the optimization to willingly move towards a bad neighbourhood so that the trajectory may jump out of the local minima and move towards the global minima. There are many processes applied to solve this particular problem, one would be Simulated Annealing, but however it requires the accurate evaluation of the objective function values. Furthermore, there is no theoretical for the SA algorithm which can be applied to the stochastic optimization problems. General optimization search is usually based on the neighbouring points, which means that if the neighbouring area is small, then it will be hard to jump out of any local minima and if the area is large, occasionally a very bad move is possible which will eventually lead to inefficient optimization.

The stochastic continuous algorithm takes into account a very wide spread of initial conditions so that the local minima with respect to each starting condition is found out and compared with each other, eventually finding out the global optima. This process converges slowly but is highly efficient and provides better results. The Stochastic algorithm for the continuous optimization which is described in [30] is used in this thesis to converge to a global optimum. the steps are as follows:

- Initialize the sample sets S and Z.
- Initialize $X_0 = S_0$, $n = 0$.
- For a given sample $X_n = S_n$, choose the next sample point from Z_n.
- For a chosen set $Z_n = r_n$, set

  $$X_{n+1} = \begin{cases} Z_n \ , & \textit{with a probability } p_n \\ X_n \ , & \textit{with a probability } (1 - \ p_n) \end{cases}$$

  where $p_n = P[g(r_n) < g(s_n)]$
- Replace n by n+1 and go to Step 3.

Chapter 3

Obtaining a Locally Optimal Solution

3.1 Agent trajectories

The vital part of this thesis is to decide the function which will be assigned to the agents so that they could be easily parameterized and can provide valuable feedback control. The agent trajectories are the most important part of the solution as changing them constantly based on previous results will lead to a global optimal solution. The available trajectories are Lissajous functions, Fourier series and elliptical trajectories. In this research, we select the Fourier series because of its flexibility and ease of usage to change the parameters and obtain the optimal solution. Fourier series trajectories are well known to approximate any type of periodic curves of arbitrary shape by dividing the periodic function into sum of many simple sinusoidal functions. Therefore in the frequency domain, they provide a wider base when there is non-uniform distribution of differently weighted sampling points and irregular mission space. We consider N mobile robots spread across a two dimensional rectangular mission space $\Omega \equiv$ [0, R_1] X [0, R_2] $\in R^2$. The position of any agent at any time t is given by:

$$s_n(t) = [s_n^x(t), s_n^y(t)],\ where\ s_n^x(t)\ \in [0, R_1]\ \ and\ \ s_n^y(t)\ \in [0, R_2] \tag{3.1}$$

The agents follow dynamics as below:

$$\dot{s}_n^x(t) = \ u_n(t) cos\theta_n(t) \ , \quad \dot{s}_n^y(t) = \ u_n(t) sin\theta_n(t) \tag{3.2}$$

Where $u_n(t)$ is the scalar velocity of the nth agent and $\theta_n(t)$ is the angle relative to the positive direction which lies between 0 and 2π. As we assign Fourier series to all the mobile agents, we assume that γ is the order of the Fourier series. The agents path in terms of Fourier series can be determined as follows:

$$s_n^x(t) = \ a_{n,0}^x + \sum_{\gamma=1}^{\Gamma_n^x} a_{n,\gamma}^x \sin\left(2\pi\gamma f_n^x \rho_n(t) + \ \varphi_{n,\gamma}^x\right) \tag{3.3}$$

$$s_n^y(t) = \ a_{n,0}^y + \sum_{\gamma=1}^{\Gamma_n^y} a_{n,\gamma}^y \sin\left(2\pi\gamma f_n^y \rho_n(t) + \ \varphi_{n,\gamma}^y\right) \tag{3.4}$$

It is assumed that all the robots can control and change their own speed and direction with the change in their parameters. General assumptions made are that each robot is a point particle and the individual robots cannot collide with one another. The sensing range taken is one unit of the space which means that the robot can sense everything within one unit radius around it. In our two dimensional approach, each agents trajectory is represented by individual parametric equations for all

$$s_n^x(t) = f(\sigma_n, \rho_n(t)), \qquad\qquad s_n^y(t) = g(\sigma_n, \rho_n(t)) \tag{3.5}$$

n=1,,N and $\sigma_n = [\sigma_n^1, \sigma_n^2, \sigma_n^3, \sigma_n^\Gamma]^T$ is the vector of the parameters by which we control the agent trajectories. The position of the agent over time is controlled by the function $\rho_n(t)$. We can observe that the functions f(.) and g(.) depend on the inputs $u_n(t)$ and $\theta_n(t)$ and hence the parameter vector indirectly will affect them and therefore there is a clear dependence of the functions f(.) and g(.) on the parameter vector σ_n. Also this clear dependence allows us to find out the derivative with respect to the parameters.

$$\frac{\partial s_n^x}{\partial \sigma_n} = \frac{\partial f(\sigma_n, \rho_n(t))}{\partial \sigma_n} \ \ and \ \ \frac{\partial s_n^y}{\partial \sigma_n} = \frac{\partial g(\sigma_n, \rho_n(t))}{\partial \sigma_n} \tag{3.6}$$

3.2 Cost Function

Let us assume that there are M points spread uniformly over the entire mission space which serve as objects of interest for the multiple agents. There is also a probability function associate with each point $[a, b]$ which determines the probability of each point to be detected by any mobile agent over the entire mission space. It is also assumed that the probability is 1 if $[a, b] = s_n$ and that the probability function is a monotonically non-increasing function in the Euclidean distance D(w,s_n) = $||w$ - $s_n||$ between w and s_n. This captures the reduced effectiveness of the agents sensor over its finite range of a single unit. Therefore the probability is zero if D(w, s_n) is greater than one unit. Therefore the probability function can be given as:

$$p_n(w, s_n(t)) = \{ \begin{array}{cc} 1 - D(w, s_n(t)), & if\ D(w, s_n)\ \leq 1 \\ 0 \quad , & if\ D(w, s_n)\ > 1 \end{array} . \tag{3.7}$$

As we have multiple agents in our analysis, we have to consider the joint probability of each point which may be detected by any mobile agent and not just consider a single agent while calculating the probability function. We have to assume that there must be at least a single point over the total space which cannot be detected initially by all the agents. Therefore the total joint probability of detecting any point when all the agents are considered assuming detection independence is given as:

$$P_m(s(t)) = 1 - \sum_{n=1}^{N} [1 - \ p_n(w, s_n(t))] \tag{3.8}$$

3.3 Uncertainty based analysis

It this approach, a certain time varying factor of uncertainty $'Q'$ is associated with every point $[a, b]$ which increases at a fixed rate A if the point is not being sensed by any mobile agent ($P_m(s(t)) = 0$) and decreases by a probabilistic rate B when it is being sensed by any agent. Moreover the uncertainty cannot be negative because of its obvious reasons. Therefore the final modelled dynamics of the uncertainty function is:

$$\dot{Q}_m(t) = \ \{ \begin{array}{cc} 0 & , \ if\ Q_m(t) = 0, \quad A_m \leq BP_m \\ A_m - BP_m(s(t)) & , \ otherwise \end{array} \tag{3.9}$$

Also, it is assumed that all the initial conditions $Q_m(0), m = 1, , M$ are given and $B > Am > 0$ for all $m = 1, , M$. This ensures that the uncertainty strictly decreases when $[a, b] = s_n$. Therefore the aim here will be to reduce the total uncertainty of the points in the mission space by controlling through $u_n(t), \theta_n(t)$ and hence controlling all the movement of the mobile agents so that they visit every point and reduce the overall uncertainty within a fixed time horizon. Therefore the cost function is given by:

$$\min_{u_n(t),\ \theta_n(t)} J = \int_0^T \sum_{m=1}^{M} Q_m(t)dt \tag{3.10}$$

Subject to the uncertainty dynamics (3.9) and the agent dynamics (3.2) and control constraints $0 \leq \theta_n(t) \leq 2pi, 0 \leq u_n(t) \leq 1, t \in [0, T]$ and constraints $s_n(t) \in \omega$ for all $t \in [0, T], n = 1, , N$.

3.4 Applying Optimal Control

We apply the Hamiltonian approach to solve the required optimal control problem. The state vector is taken as $x(t) = [s_1^x(t), s_1^y(t), , s_N^x(t), s_N^y(t),\ Q_1(t), ,\ \ Q_M(t)]^T$ and the co-state vector $\vartheta(t) = \ [\mu_1^x(t), \mu_1^y(t), , \mu_N^x(t), \mu_N^y(t),\ \lambda_1(t), ,\ \ \lambda_M(t)]^T$. Due to the discontinuity of dynamics of $Q_m(t)$, the optimal state trajectory solution may contain a boundary arc where $Q_m(t) = 0$ for any m, otherwise the state will evolve in an interior arc. The derived Hamiltonian is:

$$H = \sum_m Q_m(t) + \sum_m \lambda_m.\dot{Q}_m(t) + \sum_n \mu_n^x.u_n(t).cos\ \theta_n(t) + \sum_n \mu_n^y.u_n(t).sin\ \theta_n(t) \tag{3.11}$$

And the co-state equations $\dot{\vartheta} = \ -\frac{\partial H}{\partial x}$ are:

$$\dot{\lambda}_m(t) = \ -\frac{\partial H}{\partial Q_m} = \ -1 \tag{3.12}$$

$$\begin{aligned} \dot{\mu}_n^x(t) &= -\tfrac{\partial H}{\partial s_n^x} = -\textstyle\sum_m \tfrac{\partial}{\partial s_n^x}.\lambda_m.\dot{Q}_m(t) \\ &= -\sum_{[a_m,b_m]\ \in\ R(s_n)} \frac{B\lambda_m(s_n^x -\ a_m)}{D(a_m, b_m, s_n(t))} \sum_{i\neq n}^{N} [1 -\ p_i(w_m, s_i(t))] \end{aligned} \tag{3.13}$$

$$\dot{\mu}_n^y(t) = -\frac{\partial H}{\partial s_n^y} = -\sum_m \frac{\partial}{\partial s_n^y}.\lambda_m.\dot{Q}_m(t)$$

$$= - \sum_{[a_m,b_m] \in R(s_n)} \frac{B\lambda_m(s_n^y - b_m)}{D(a_m,b_m,s_n(t))} \sum_{i\neq n}^{N}[1 - p_i(w_m,s_i(t)] \tag{3.14}$$

Moreover, the above Hamiltonian can also be represented after some algebraic operations and by combining the trigonometric terms, we obtain the following:

$$H = \sum_m Q_m(t) + \sum_m \lambda_m.\dot{Q}_m(t) + \sum_n sgn(\mu_n^y(t)).u_n(t).\frac{sin(\ \theta_n(t) + \varphi_n(t))}{\sqrt{(\mu_n^x(t))^2 + (\mu_n^y(t))^2}} \tag{3.15}$$

$\varphi_n(t)$ is defined such that $\tan(\varphi_n(t)) = \frac{\mu_n^x(t)}{\mu_n^y(t)}$. Applying the Pontryagin minimum principle to the Hamiltonian equation with optimum values of $\theta_n(t)$ and $u_n(t)$, it can be immediately concluded that it is absolutely necessary for the above optimal control problem to satisfy $u_n^*(t) = 1$. The optimal value $\theta_n^*(t)$ can be found out by solving $\frac{\partial H}{\partial \theta_n} = 0$:

$$\frac{\partial H}{\partial \theta_n} = -\mu_n^x.u_n(t).sin\ \theta_n(t) + \mu_n^y.u_n(t).cos\ \theta_n(t) = 0 \tag{3.16}$$

Which further gives $\theta_n^*(t) = \frac{\mu_n^y(t)}{\mu_n^x(t)}$. Since we know that $u_n^*(t) = 1, n = 1,, N$, the only variable left to calculate is $\theta_n^*(t)$. This can be done by modelling the problem as a two point boundary value problem which shall involve both forward and backward integrations of the co-state and state equations to find out the value of $\partial H/(\partial\theta_n)$ after each iteration and by applying gradient descent approach until the required objective function converges to a possibly global or local minimum. Clearly this is a non-convex and a computationally intensive problem which scales poorly with size of the mission space and the number of agents. It is also required that the mission time should be discretized which further adds complexity to the whole process.

Since now it is known that $u_n^*(t) = 1, n = 1,, N$, we can tell from (3.2) that $(\dot{s}_n^x)^2 + (\dot{s}_n^y)^2 = 1$, which gives from (3.6):

$$(\frac{\partial f(\sigma_n,\rho_n(t))}{\partial \rho_n(t)}\dot{\rho}_n(t))^2 + (\frac{\partial g(\sigma_n,\rho_n(t))}{\partial \rho_n(t)}\dot{\rho}_n(t))^2 = 1 \tag{3.17}$$

From the above equation, we obtain the dynamics of $\rho_n(t)$ as:

$$\dot{\rho}_n(t) = [(\frac{\partial f(\sigma_n,\rho_n(t))}{\partial \rho_n(t)})^2 + (\frac{\partial g(\sigma_n,\rho_n(t))}{\partial \rho_n(t)})^2]^{\frac{-1}{2}} \tag{3.18}$$

With the required initial conditions $\rho_n(0) = 0, n = 1, ..., N$ and also $\rho_n(t) \in [0,\ 2\pi)$. Therefore the final dynamics of $\rho_n(t)$ is:

$$\dot{\rho}_n(t) = \frac{1}{2\pi}[(f_n^x \sum_{\gamma=1}^{\gamma_n^x} a_{n,\gamma}^x \cos(2\pi\gamma f_n^x \rho_n(t) + \varphi_{n,\gamma}^x))^2 + (f_n^y \sum_{\gamma=1}^{\gamma_n^y} a_{n,\gamma}^y \cos(2\pi\gamma f_n^y \rho_n(t) + \varphi_{n,\gamma}^y))^2]^{\frac{-1}{2}} \tag{3.19}$$

3.4.1 IPA Equations

The cost function which was derived earlier (3.10) can also be written as:

$$\min_{\sigma_n,\ n=1,,N} J = \int_0^T \sum_{m=1}^{M} Q_m(\sigma_1, , \sigma_n, t)dt \tag{3.20}$$

Due to the dependence of the uncertainty on the parameters of the agent trajectory, the cost function is modified in the above form to perform necessary analysis. Considering our system as a hybrid system, we concentrate on all the events which are causing transitions in the dynamics of $Q_m(t)$. Initially if $Q_m(t) = 0$ and $A_m - BP_m \leq 0$, applying the IPA equations to $Q_m(t)$ and using (3.9), we get

$$\frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n} = 0 \tag{3.21}$$

When $Q_m(t) \leq 0$, we have

$$\frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n} = -B\frac{\partial p_n(w_m, s_n(t))}{\partial \sigma_n}\sum_{i\neq n}^{N}[1 - p_i(w_m, s_i(t)] \tag{3.22}$$

Noting that $p_n(w_m, s_n(t)) \equiv p_n(D(w_m, s_n(t)))$, we have

$$\frac{\partial p_n(w_m, s_n(t))}{\partial \sigma_n} = \frac{\partial p_n(w_m, s_n(t))}{\partial D(w_m, s_n(t))}.\frac{\partial D(w_m, s_n(t))}{\partial \sigma_n} \tag{3.23}$$

Where $D(w_m, s_n(t)) = [(s_n^x - a_m)^2 + (s_n^y - b_m)^2]^{\frac{1}{2}}$. Using partial differential equations and by representing $D(w_m, s_n(t))$ as D, we get:

$$\frac{\partial D}{\partial \sigma_n} = \frac{1}{2D}(\frac{\partial D}{\partial s_n^x}\frac{\partial s_n^x}{\partial \sigma_n} + \frac{\partial D}{\partial s_n^y}\frac{\partial s_n^y}{\partial \sigma_n}) \tag{3.24}$$

And from (3.7), $\frac{\partial p_n(w_m,s_n(t))}{\partial D(w_m,s_n(t))} = -1$ and hence the final equation of the uncertainty dynamics is:

$$\frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n} = \frac{B}{2D}(\frac{\partial D}{\partial s_n^x}\frac{\partial s_n^x}{\partial \sigma_n} + \frac{\partial D}{\partial s_n^y}\frac{\partial s_n^y}{\partial \sigma_n})\sum_{i\neq n}^{N}[1 - p_i(w_m, s_i(t)] \tag{3.25}$$

Where $\frac{\partial D(w_m,s_n(t))}{\partial s_n^x} = 2(s_n^x - a_m)$ and $\frac{\partial D(w_m,s_n(t))}{\partial s_n^y} = 2(s_n^y - b_m)$. The only differentials of the equation left to be calculated are $(\partial s_n^x)/(\partial \sigma_n) and (\partial s_n^y)/(\partial \sigma_n)$ which can be calculated by individually from (3.3).

The amplitude parameter vectors for each agent's position values s_n^x and s_n^y are $A_n^x = [a_{n,0}^x, a_{n,1}^x, , a_{n,\gamma_n^x}^x]$ and $A_n^y = [a_{n,0}^y, a_{n,1}^y, , a_{n,\gamma_n^y}^y]$. Their respective phase parameters are $\varphi_n^x = [\varphi_{n,0}^x, \varphi_{n,1}^x, , \varphi_{n,\gamma_n^x}^x]$ and $\varphi_n^y = [\varphi_{n,0}^y, \varphi_{n,1}^y, , \varphi_{n,\gamma_n^y}^y]$. Therefore the parameter vector is defined as $\sigma_n = [\ A_n^x\ A_n^y\ f_n^x\ \varphi_n^x\ \varphi_n^y\ ]$ and hence we can define the values of $(\frac{\partial s_n^x)}{(\partial \sigma_n)} and (\frac{\partial s_n^y)}{(\partial \sigma_n)}$ as:

$$\frac{\partial s_n^x}{\partial \sigma_n} = [\frac{\partial s_n^x}{\partial A_n^x}, \frac{\partial s_n^x}{\partial A_n^y}, \frac{\partial s_n^x}{\partial f_n^x}, \frac{\partial s_n^x}{\partial \varphi_n^x}, \frac{\partial s_n^x}{\partial \varphi_n^y}] \tag{3.26}$$

$$\frac{\partial s_n^y}{\partial \sigma_n} = [\frac{\partial s_n^y}{\partial A_n^x}, \frac{\partial s_n^y}{\partial A_n^y}, \frac{\partial s_n^y}{\partial f_n^x}, \frac{\partial s_n^y}{\partial \varphi_n^x}, \frac{\partial s_n^y}{\partial \varphi_n^y}] \tag{3.27}$$

From (3.3) , each value of the above vector are found out to be:

$$\frac{\partial s_n^x}{\partial A_n^x} = \{ \begin{array}{cc} 1 \qquad , & \gamma = 0 \\ \sin(2\pi\gamma f_n^x \rho_n(t) + \varphi_{n,\gamma}^x), & otherwise \end{array} \} \tag{3.28}$$

$$\frac{\partial s_n^y}{\partial A_n^x} = 0 \tag{3.29}$$

$$\frac{\partial s_n^y}{\partial A_n^y} = \{ \begin{array}{cc} 1 \qquad , & \gamma = 0 \\ \sin(2\pi\gamma f_n^y \rho_n(t) + \varphi_{n,\gamma}^y), & otherwise \end{array} \} \tag{3.30}$$

$$\frac{\partial s_n^x}{\partial A_n^y} = 0 \tag{3.31}$$

$$\frac{\partial s_n^x}{\partial f_n^x} = 2\pi\rho_n(t)\sum_{\gamma=1}^{\gamma_n^x} a_{n,\gamma}^x.\gamma.\cos(2\pi\gamma f_n^x \rho_n(t) + \{\varphi_{n,\gamma}^x) \tag{3.32}$$

$$\frac{\partial s_n^y}{\partial f_n^x} = 0 \tag{3.33}$$

$$\frac{\partial s_n^x}{\partial \varphi_n^x} = a_{n,\gamma}^x . \cos(2\pi\gamma f_n^x \rho_n(t) + \ \varphi_{n,\gamma}^x) \tag{3.34}$$

$$\frac{\partial s_n^y}{\partial \varphi_n^x} \ = 0 \tag{3.35}$$

$$\frac{\partial s_n^x}{\partial \varphi_n^y} \ = 0 \tag{3.36}$$

$$\frac{\partial s_n^y}{\partial \varphi_n^y} = \ a_{n,\gamma}^y . \cos\left(2\pi\gamma f_n^y \rho_n(t) + \{\varphi_{n,\gamma}^y\right) \tag{3.37}$$

Using all the above equations, we can finally find the value of $\frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n}$ and then use it to find out $\frac{\partial Q_m(t)}{\partial \sigma_n}$. As we have a hybrid behaviour in the dynamics of $Q_m(t)$, we separate the continuous integration into parts within which there are no changes in dynamics, which means that we break the whole time period into various small event times τ_k, which define the number of changes in the dynamics of $Q_m(t)$. The value of K' defines the total number of changes in dynamics and therefore we integrate within each range $[\tau_k, \tau_(k+1))\forall k = 0,, K$ and finally sum up the whole integral value. The gradient of the uncertainty is given by:

$$\frac{\partial Q_m(t)}{\partial \sigma_n} = \frac{\partial Q_m(\tau_k^+)}{\partial \sigma_n} + \{ \begin{array}{ll} 0 & , \ if \ Q_m(t) = 0, \quad A_m \leq BP_m(s(t)) \\ \int_{\tau_k}^{\tau_{k+1}} \frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n} \ dt & , \ otherwise \end{array} . \tag{3.38}$$

The only part of the equation left to be solved is $\frac{\partial Q_m(\tau_k^+)}{\partial \sigma_n}$ which is solved using the IPA equations from chapter 2. The gradient $\nabla\tau_k = \frac{\partial \tau_k}{\partial \sigma_n}$ for k = 1,. . . ,K,. Since the system is a hybrid, we have two cases where the dynamics shift and hence we have two values for$\frac{\partial Q_m(\tau_k^+)}{\partial \sigma_n}$.

Case 1: If at any τ_k, $\dot{Q}_m(t)$ switches from $\dot{Q}_m(t) = 0$ to $\dot{Q}_m(t) = A_m - BP_m(s(t))$, it is obvious that the dynamics are continuous and therefore we get:

$$\frac{\partial Q_m(\tau_k^+)}{\partial \sigma_n} = \frac{\partial Q_m(\tau_k^-)}{\partial \sigma_n}, \quad m = 1,, M \tag{3.39}$$

Case 2: If at any $\tau_k, \dot{Q}_m(t)$ switches from $\dot{Q}_m(t) = A_m - BP_m(s(t)) to \dot{Q}_m(t) = 0$, then we have $Q_m(\tau_k) = 0$. In this case, similar to [28], we determine the value of $\nabla\tau_k$ in order to determine $\frac{\partial Q_m(\tau_k^+)}{\partial\sigma_n}$. This event is endogenous and therefore from [28], we have

$$\nabla\tau_k = -\frac{\nabla Q_m(\tau_k^-)}{A_m - BP_m(s(\tau_k^-))} \tag{3.40}$$

$$\frac{\partial Q_m(\tau_k^+)}{\partial\sigma_n} = \frac{\partial Q_m(\tau_k^-)}{\partial\sigma_n} - \frac{[\ A_m - BP_m(s(\tau_k^-))]\frac{\partial Q_m(\tau_k^-)}{\partial\sigma_n}}{A_m - BP_m(s(\tau_k^-))} = 0 \tag{3.41}$$

Therefore from the above two equations, we conclude that $\frac{\partial Q_m(\tau_k^+)}{\partial\sigma_n}$ is always equal to 0 regardless of $\frac{\partial Q_m(\tau_k^-)}{\partial\sigma_n}$.

3.4.2 Gradient Evaluation

The cost function as derived earlier was:

$$\min_{\sigma_n,\ n=1,,N} J = \int_0^T \sum_{m=1}^{M} Q_m(\sigma_1, , \sigma_n, t)dt \tag{3.42}$$

According to optimal control approach, we have to minimise the gradient of the cost function, and therefore the gradient of the cost function is found by differentiating the cost function:

$$\nabla J = \sum_{m=1}^{M}\sum_{k=0}^{K}(\int_{\tau_k}^{\tau_{k+1}} \nabla Q_m(t)dt +\ Q_m(\tau_{k+1})\nabla\tau_{k+1} -\ Q_m(\tau_k)\nabla\tau_k) \tag{3.43}$$

We observe from prior calculations that all the terms of the form $Q_m(\tau_k)\nabla\tau_k$ are cancelled out for all k (with $\tau_0 = 0, \tau_(k+1) = T$ fixed) and therefore we finally obtain the following:

$$\nabla J = \sum_{m=1}^{M}\sum_{k=0}^{K}\int_{\tau_k}^{\tau_{k+1}} \nabla Q_m(t)dt \tag{3.44}$$

3.4.3 Parameter Optimization

Our main aim is to minimise $J(\sigma_1, , \sigma_n)$ by finding out an optimal parameter vector $[\sigma_1^*, , \sigma_n^*]$. Therefore after every iteration of the process, we update the parameter vector by a small portion from the gradient of the cost function which gives:

$$[\sigma_1^{l+1},,\sigma_n^{l+1}] = [\sigma_1^l,,\sigma_n^l] - \eta\nabla J(\sigma_1^l,,\sigma_n^l) \tag{3.45}$$

This process continues until when the value of ∇J is less than a particular threshold $'\epsilon'$ and also the value η is reduced after every iteration so that the settlement of the system at any optimum point is smooth.

The procedure in this approach is:

1. Initialize the parameter vector σ_n and the initial conditions of uncertainty $Q_m(0)$ and also the values of η, A and B.
2. Calculate the value of $\rho_n(t)$.
3. Calculate the initial agent trajectories $s_n^x(t)\ and\ s_n^y(t)$.
4. Using the above trajectories, find out the derivative of the gradient of the uncertainty $\frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n}$with respect to each point in the mission space.
5. Integrate the above derivative to find out the gradient of the uncertainty$\frac{\partial Q_m(t)}{\partial \sigma_n}$.
6. Using the gradient of uncertainty, find out the cost gradient

$$\nabla J = \sum_{m=1}^{M}\sum_{k=0}^{K}\int_{\tau_k}^{\tau_{k+1}} \nabla Q_m(t)dt$$

7. Finally, update the parameter vector $[\sigma_1^{l+1},,\sigma_n^{l+1}] = [\sigma_1^l,,\sigma_n^l] - \eta\nabla J(\sigma_1^l,,\sigma_n^l)$ and repeat the whole process until the cost gradient is less than a threshold $'\epsilon'$.
8. Apply stochastic continuous algorithm and repeat the above steps to obtain the best parameter vector.
9. Plot all the agent trajectories after every 10 iterations.

3.5 Nearest Point Optimization

We have seen the uncertainty based optimization for the present problem which includes lot of complex calculations and integrations which are highly non-convex in nature. Another approach to this problem would be to find out the closest the agent trajectories get to any point and then try to optimize that particular distance from the point. The idea is to find out a point in the whole trajectory which is closest to any fixed point of interest and then we try to minimize the distance between the two points by updating the parameters.

Let us assume that r_n is the sensing radius of any agent in consideration. Therefore from (3.5), the probability of any point $w_m = [a, b]$ to be detected by the agent is

$$p_n(w_m, s_n(t)) = \{ \begin{array}{cc} 1 - \frac{D(w_m, s_n(t))}{r_n}, & if\ D(w, s_n)\ \leq 1 \\ 0 \quad , & if\ D(w, s_n) > 1 \end{array} . \tag{3.46}$$

The cost function should try to minimize the distance between the point of interest and the closest point in its trajectory. Which means the cost is zero if the agent senses the points of interest and is an increasing function with respect to the distance if the agent is not sensing the point. Therefore the cost function is taken as:

$$J(\sigma_1,,\ \sigma_n) = \max_{m=1,,\ M} (0,\ (\frac{D(w_m, s_n(t_c))}{r_n})^2 - 0.5)^2 \tag{3.47}$$

The cost function is taken as a square function to help us find out a continuous and differentiable gradient for updating the parameters. Here, the term $s_n(t_c)$ represents the point in the trajectory of the 'nth' agent which is closest to the point of interest w_m. Applying optimal approach to the above cost function requires the gradient of the cost function which can be calculated by derivation of the cost function with respect to the parameter vector.

$$\frac{\partial J}{\partial \sigma_n} = \frac{\partial}{\partial \sigma_n}(\max_{m=1,,\ M} (0,\ (\frac{D(w_m, s_n(t_c))}{r_n})^2 - 0.5)^2) \tag{3.48}$$

$$\nabla J\ = 2\ (\max_{m=1,,\ M} (0,\ 1 -\ (\frac{D(w_m, s_n(t_c))}{r_n})^2)). \frac{2D(w_m, s_n(t_c))}{{r_n}^2}\ .\ \frac{\partial D(w_m, s_n(t_c))}{\partial \sigma_n} \tag{3.49}$$

Applying partial differentiation and from (3.24), we get the cost gradient as:

$$\nabla J = \frac{2}{{r_n}^2}\left(\frac{\partial D(w_m, s_n(t_c))}{\partial s_n^x}\frac{\partial s_n^x}{\partial \sigma_n} + \frac{\partial D(w_m, s_n(t_c))}{\partial s_n^y}\frac{\partial s_n^y}{\partial \sigma_n}\right) . \max_{m=1,, M}(0, \ (\frac{D}{r_n})^2 - 0.5) \tag{3.50}$$

Where $\frac{\partial D(w_m,s_n(t))}{\partial s_n^x} = 2(s_n^x - \ a_m)$and $\frac{\partial D(w_m,s_n(t))}{\partial s_n^y} = 2(s_n^y - \ b_m)$. The terms $\frac{\partial s_n^x}{\partial \sigma_n}$ and $\frac{\partial s_n^y}{\partial \sigma_n}$ can be found out similarly as in (3.26) - (3.34) .

With the help of all the equations above, we finally obtain the value of the cost gradient ∇J and use it to update the parameter vector.

$$[\sigma_1^{l+1}, , \sigma_n^{l+1}] = \ [\sigma_1^l, , \sigma_n^l] - \ \eta \nabla J(\sigma_1^l, , \sigma_n^l) \tag{3.51}$$

This process is continued until the gradient is below a fixed threshold or when all the points of interest are well observed and then we have an optimal solution of the problem which has an optimal trajectory of the agent within the mission space.

The procedure of this approach is:

1. Initialize the parameter vector σ_n and the initial conditions of uncertainty $Q_m(0)$ and also the values of η, A and B.

2. Calculate the value of $\rho_n(t)$.

3. Calculate the initial agent trajectories $s_n^x(t)$ and $s_n^y(t)$.

4. Using the agent trajectories, find out the cost gradient:

$$\nabla J = \frac{2}{{r_n}^2}\left(\frac{\partial D(w_m, s_n(t_c))}{\partial s_n^x}\frac{\partial s_n^x}{\partial \sigma_n} + \frac{\partial D(w_m, s_n(t_c))}{\partial s_n^y}\frac{\partial s_n^y}{\partial \sigma_n}\right) . \max_{m=1,, M}(0, \ (\frac{D}{r_n})^2 - 0.5) \tag{3.52}$$

5. Finally, update the parameter vector $[\sigma_1^{l+1}, , \sigma_n^{l+1}] = \ [\sigma_1^l, , \sigma_n^l] - \ \eta \nabla J(\sigma_1^l, , \sigma_n^l)$ and repeat the whole process until the cost gradient is less than a threshold $'\epsilon'$.

6. Apply stochastic continuous algorithm and repeat the above steps to obtain the best parameter vector.

7. Plot all the agent trajectories after every 10 iterations.

3.6 Obstacle Detection Problem

We have calculated and solved the optimal control problem by two different methods and now by taking it a step further, we try to omit some point in the mission space as an obstacle through which the agent trajectory shall not pass. To be able to attain a solution for this problem, the above two methods must be combined to create a new parameter updating function which will realise even the obstacle avoiding problem. The uncertainty analysis requires the sum of the total uncertainty to be minimised and therefore this method cannot be both used for surveillance as well as obstacle avoidance.

The cost function that we have come up with for obstacle avoidance is similar to (3.47) but instead of moving towards the point as in gradient descent, we use the positive gradient approach so as to move away from the point and maximise the distance of the closest point to the obstacle as much as possible. Therefore, the parameter updating equation from (3.45) and (3.47) will be:

$$[\sigma_1^{l+1}, , \sigma_n^{l+1}] = \ [\sigma_1^l, , \sigma_n^l] - \eta_3 \eta_1 \nabla J_1(\sigma_1^l, , \sigma_n^l) - \ (1-\eta_3)\ \eta_2 \nabla J_2(\sigma_1^l, , \sigma_n^l) \tag{3.53}$$

$$\nabla J_1 = \sum_{m=1}^{M} \sum_{k=0}^{K} \int_{\tau_k}^{\tau_{k+1}} \nabla Q_m(t) dt \quad or \tag{3.54}$$

$$\nabla J_1 = \frac{2}{{r_n}^2} \left(\frac{\partial D(w_m, s_n(t_c))}{\partial s_n^x} \frac{\partial s_n^x}{\partial \sigma_n} + \frac{\partial D(w_m, s_n(t_c))}{\partial s_n^y} \frac{\partial s_n^y}{\partial \sigma_n} \right) \max_{m=1,,\ M} (0,\ (\frac{D}{r_n})^2 - 0.5) \tag{3.55}$$

$$\nabla J_2 = -\ \frac{2}{{r_n}^2} \left(\frac{\partial D(w_o, s_n(t_c))}{\partial s_n^x} \frac{\partial s_n^x}{\partial \sigma_n} + \frac{\partial D(w_o, s_n(t_c))}{\partial s_n^y} \frac{\partial s_n^y}{\partial \sigma_n} \right) \max_{m=1,,\ M} (0,\ 1 -\ (\frac{D}{r_n})^2\ ) \tag{3.56}$$

Where $J_2 = max_{m=1,,\ M}(0,\ 1 - (\frac{D(w_m, s_n(t_c))}{r_n})^2)^2$. The cost is zero if the point is outside the sensing range and it is positive and increasing as the trajectory comes close to it within the sensing range r_n.

The procedure which is followed is:

1. Initialize the parameter vector σ_n and the initial conditions of uncertainty $Q_m(0)$ and also the values of η, A and B.
2. Calculate the value of $\rho_n(t)$.
3. Calculate the initial agent trajectories $s_n^x(t)$ and $s_n^y(t)$.
4. Using the above trajectories, find out the derivative of the gradient of the uncertainty
$$\frac{d}{dt}\frac{\partial Q_m(t)}{\partial \sigma_n}$$
with respect to each point in the mission space.
5. Integrate the above derivative to find out the gradient of the uncertainty $\frac{\partial Q_m(t)}{\partial \sigma_n}$.
6. Using the gradient of uncertainty, find out the cost gradient from (3.55) and (3.56).
7. Finally, update the parameter vector from (3.53) and repeat the whole process until both the cost gradients are less than a threshold $'\epsilon'$.
8. Apply stochastic continuous algorithm and repeat the above steps to obtain the best parameter vector.
9. Plot all the agent trajectories after every 10 iterations.

The parameter which is very crucial for updating the parameter vector is the value of η. There are various methods and procedure to choose the value of η some of which are tried and tested out in this thesis. As the problem is very complex and non-convex, we still cannot assure which one is the best. The general cost gradient sums up to be a very huge number due to three step vector integration. The methods of deciding the value of η which are tested are:

• Initializing the value of η as a small number 0.1 and then multiplying it by 0.1 after every iteration because the cost gradient is around four to five times the power of ten.

• Normalizing the power of tens in the cost gradient and then taking a specific value of η 0.1 to 0.2.

• Providing an individual η to the cost gradient value for every point of interest with respect to the individual cost, so as to achieve better control.

$$\nabla J = \sum_{i=1}^{M} \eta_m \nabla J_m \tag{3.57}$$

Where $\nabla J_m = \sum_{k=0}^{K} \int_{\tau_k}^{\tau_{k+1}} \nabla Q_m(t) dt$.

Chapter 4

Towards A Globally Optimal Solution

As discussed in chapter 2, this stochastic comparison algorithm is necessary to obtain global optimum in systems where there are no specific points where the gradient of the cost function is equal to zero, which is definitely the case here. Due to the highly non-convex behaviour of the system, it is difficult to conclude whether the solution obtained is globally optimum because this approach requires assumption of many initial conditions and has three step integration which makes it very complex. In this thesis, we use the Stochastic comparison algorithm in [30] to find the global optimum.

The cost function $min_{\sigma_n,\ n=1,,N} J(\sigma_1, , \sigma_n) = \int_0^T \sum_{m=1}^M Q_m(\sigma_1, , \sigma_n, t)dt$ is continuous in σ_n and hence the stochastic comparison algorithm for a general continuous optimization problem can be applied to it. The parameter vector σ_n is considered to be a controllable vector and the stochastic analysis consists of the following steps:

- Take two sets of parametric values $\psi_y = \sigma_{n,y}{}^0, , \sigma_{n,y}{}^R]$ and $\psi_z = [\sigma_{n,z}{}^0, , \sigma_{n,z}{}^R]$

- Initialize $\sigma_n{}^0 = [\sigma_1^0, , \sigma_n^0] \in \psi_y$ and $i = 0$.

- **While** $i < R$, do the following steps:

- Take a sample value $\sigma_(n, y)^i = \psi_y(i)$.

- **Repeat**

```
• Compute the value of $s_n(t) = [s_n^x(t), s_n^y(t)]$ using $\psi_y(i)$.
• Compute the gradient of cost function $\nabla J(\sigma_(n, y)^i)$.
• Update the parameter vector $\sigma(n, y)^i$.
• **Until** $\nabla J(\sigma_(n, y)^i) < \epsilon$.
• Take another sample value $\sigma_(n, z)^i = \psi_z(i)$.
• **Repeat**
• Compute the value of $s_n(t) = [s_n^x(t), s_n^y(t)] using \psi_z(i)$ .
• Compute the gradient of cost function $\nabla J(\sigma_(n, z)^i)$ .
• Update the parameter vector $\sigma(n, z)^i$.
• **Until** $\nabla J(\sigma_(n, z)^i) < \epsilon$.
• Set the new parameter by the following rule:
```

$$\sigma_n{}^{i+1} = \{ \begin{array}{cc} \sigma_{n,z}{}^i \ , & \textit{with a probability } p^i \\ \sigma_{n,y}{}^i \ , & \textit{with a probability } 1 - \ p^i \end{array} . \tag{4.1}$$

Where $p^i = P[\ J(\sigma_{n,z}{}^i) < J(\sigma_{n,y}{}^i)]$

```
• Replace $i$ by $i + 1$.
• End **While**.
• Set $\sigma_n^* = \sigma_n^i$.
```

The above algorithm defines the method of stochastic continuous algorithm in a pseudo code manner. The first step is to take two sample sets of the initial values of the parameter vectors. We then take one value from each of those sets and then compute the whole optimal control problem as shown in previous chapter. Finally we obtain two solutions from the two initial conditions after updating the parameter vectors many number of times. The cost functions of both optimal solutions are compared and the final optimal parameter vector is found from (4.1).

The probability can be determined in two ways, the first will be updating the probability $p^i = 1$, if $J(\sigma_{n,z}^i) < J(\sigma_{n,y}^i)$ every time or else $p^i = 0$. Another method will be to observe the whole process and record the number of times $J(\sigma_(n,z)^i)$is less than $J(\sigma_(n,y)^i)$ and the final probability p^i will be:

$$p^i = \frac{no.\ of\ times\ J(\sigma_{n,z}{}^i) < J(\sigma_{n,y}{}^i)}{R\ (total\ iterations)} \tag{4.2}$$

This approach leads the solution slowly towards the global optimum but still does not ensure any such thing due to the high non-convexity of the problem.

Chapter 5

Results

5.1 Single agent, first order

There were various approaches taken and many different methods were performed on the optimal control problem to obtain the solution. The first step was to just find out the solution of a single agent problem with the order of the Fourier series to be one. Therefore a single agent is considered in the mission space which has to reduce the overall cost function of the system. We begin by assuming the initial values of the parameter vector. The initial agent trajectory is found out and from the previous equations we find out the cost gradient. We update the parameter vector until the cost gradient is less than a particular threshold.

The results obtained after every 10 iterations are shown as below:

The first figure shows us the initial trajectory of the agent with respect to the initial values of the parameter vector. We see a gradual change in the agent trajectory and the next images show the agent trajectories after every 10 iterations. The final image shows us the optimal solution obtained by a single agent which shows us clearly that the agent visits every point at least once within its sensing range of 1 unit. The cost to the number of iterations is compared with the other approaches in the end. This result gives us the basic idea how the agent optimizes its own trajectory with respect to all the points over the mission space. The time as well as the length of the trajectory is

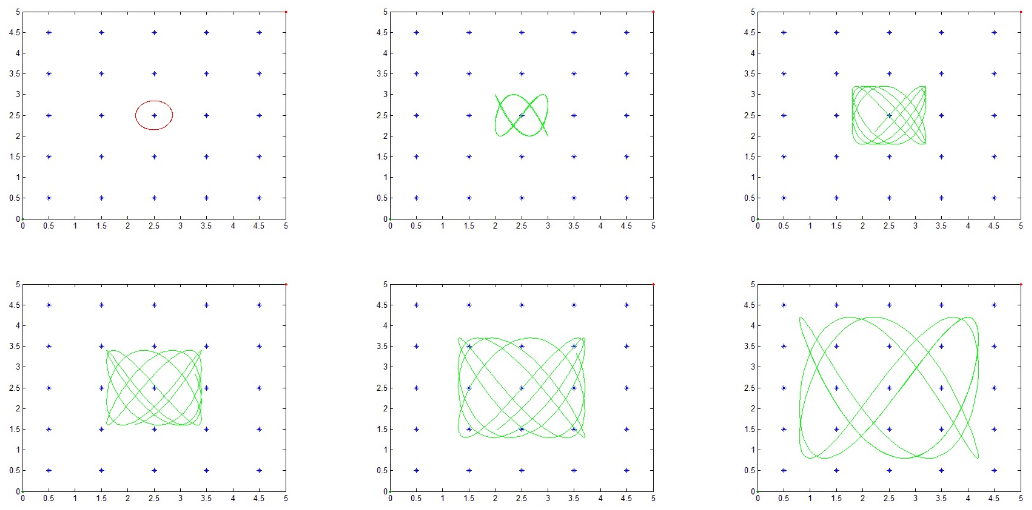

FIGURE 5.1: Agent trajectories after every 10 iterations

optimized automatically. The amount of time taken to compute the solution is also very less.

5.2 Single agent, first order, point to point cost function:

We try to slightly change the parameter updating process by updating just with respect to one point at a time and then calculating the agent trajectories and parameters again and then updating the parameter with respect to the second point and so on, i.e., instead of calculating the cost gradient as sum of the uncertainty of all the points together, we considered it to be the uncertainty of a single point at a time. Which means that the parameters are being updated with respect to each point at a time which shall provide better results. The procedure is same as previous until step no. 5 till which the uncertainty gradient $(\partial Q_m(t))/(\partial \sigma_n)$ is found out. The only difference in his and the previous is the value of the cost gradient. Here the cost gradient is $\nabla J = \sum_{m=1}^{M} \sum_{k=0}^{K} \left(\int_{\tau_k}^{\tau_{k+1}} \nabla Q_m(t)\, dt + Q_m(\tau_{k+1}) \nabla \tau_{k+1} - Q_m(\tau_k) \nabla \tau_k \right)$

Where $i = 1, , M$, keeps on increasing after every iteration, which means the cost gradient and the parameter vector are updated with respect to a single point each time and therefore in this case it requires 25 iterations just to complete one iteration of the previous approach taking into account all the points. And the process is repeated with all the points over and over.

Following are the images of the initial and final trajectories:

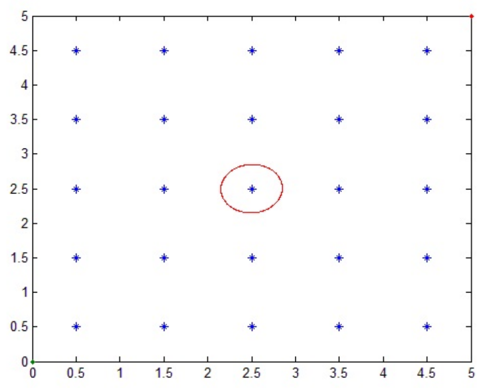
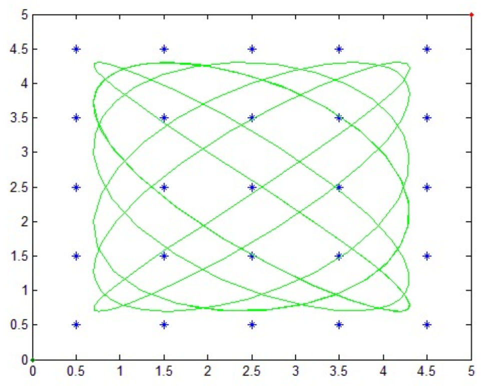

FIGURE 5.2: Initial and Final Trajectory for a point to point cost function analysis

From the above images we observe that the path of the agent slightly changes and it disperses more towards the points in the mission space. The cost versus iteration graph shows us that the approach takes more iterations to reach the lowest cost but the lowest cost is far more lower than the previous approach.

5.3 Multi agent Analysis:

Moving on forward with the number of agents we considered two mobile agents and the order of the Fourier series to be 2. The computational code is far more complex but the results are better than a single agent. In this approach the total mission space is automatically divided between the two mobile agents for the persistent surveillance of the mission space. The second order of the Furious series gives the agent for more flexibility to manoeuvre within the region space and optimize the solution. The procedure is quite similar to the single agent approach from before, the only differences are the extra agent and the order of the series. The following images shows us the mobile agents initial trajectories and the final trajectories:

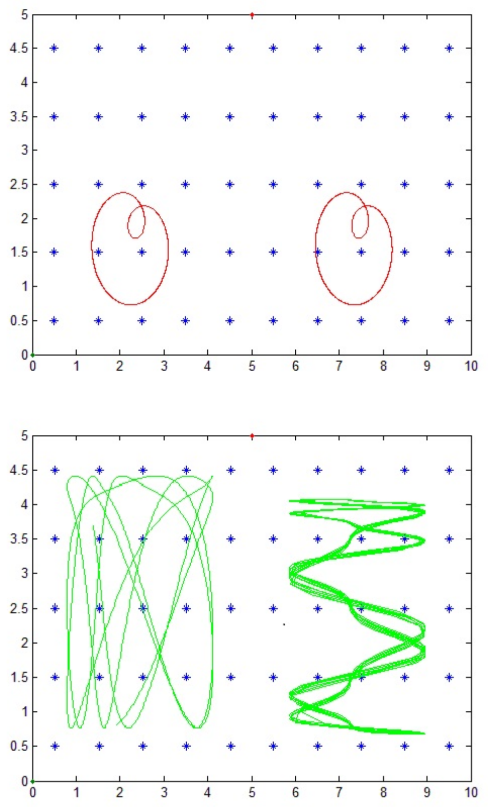

FIGURE 5.3: Initial and Final Trajectory for a multi-agent analysis

By taking a look at the above images it is clear that the agents are able to take more complex paths due to the order of Fourier series. The final optimal trajectories show us that the mobile agents have divided the whole mission space into two parts and that any single point is just observed by a single agent. This divides the total work and makes it much more time optimal.

5.4 Nearest point algorithm

From Chapter 3 a different approach of finding out the cost gradient by minimizing the closest distance to every point was discussed. Here we try to simulate the condition and observe the differences with respect to the uncertainty analysis. Here again a single agent is considered with its Fourier series order to be one.

The following results were obtained:

From the above image of the final trajectory we can easily conclude that the agent tries to pass through all the points which is obvious due to the nature of the cost function. We also observe a simplicity in calculation and much better time optimal performance but

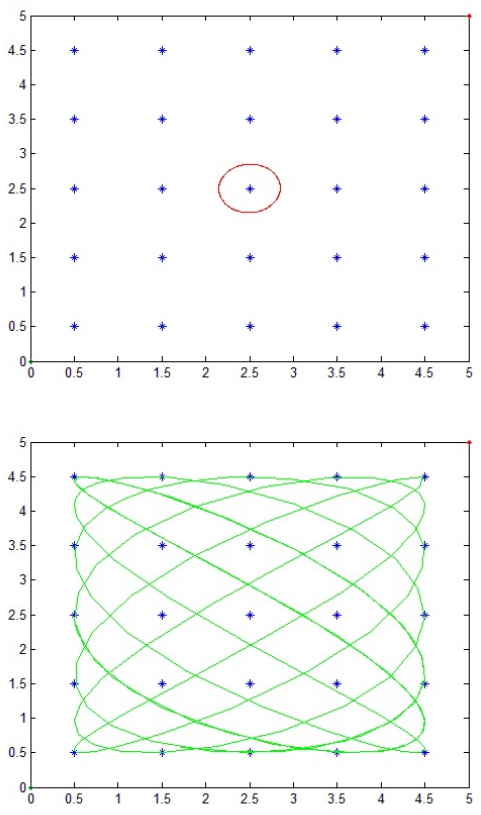

FIGURE 5.4: Initial and Final Trajectory for the nearest point analysis

it is still unsure whether the solution is globally optimal with respect to cost. Therefore we also try to plot the cost versus number of iterations to compare it with the previous solutions.

5.5 Obstacle avoidance

We now take a further step and try to put an obstacle within the mission space to observe whether the agent is able to detect obstacle and avoid it. Therefore we try to simulate a condition explained in chapter 3 which has two cost gradients, one determines the obstacles and the other ensures that all the points in the mission space are being observed respectively. Again here a single agent is considered with the order of Fourier series as one.

The following images shows the final trajectory after optimization.

The above graph shows us that the results are acceptable and the obstacle is avoided successfully. All of these approaches shows us that there are countless possibilities for the design of this specific problem and hence countless solutions.

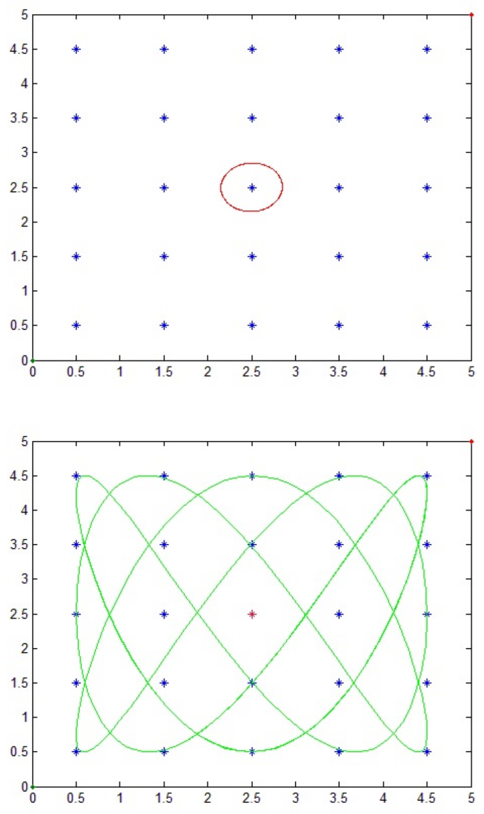

FIGURE 5.5: Obstacle avoidance initial and final trajectory

Chapter 6

Conclusions and future work

In this thesis, the multi agent persistent surveillance problem in two dimensional spaces is observed and many different approaches and solutions were obtained. Using the theories in chapter 2, we derived the solutions for the given problem. Fourier series were used to determine the agents trajectories and IPA algorithm was used by a Hamiltonian approach using gradient descent and stochastic comparison algorithm to attain the optimal solution. Also, different types of cost functions were analysed and their respective final agent trajectories and costs were compared. There are still some limitations which could not be solved by this thesis: First limitation would be deciding the value of η. It is unimaginable how the value of η affects the solution in a very major way. The optimal solution completely depends on the value of η and very slight changes in this parameter leads to huge changes in the solution. Many different approaches were tried in this thesis to decide the value of η but unfortunately it cannot be declared that any approach is the optimum because of its non-derivability and randomness. The only procedure is to select the value of η based on various experiments by hit and trial. It would be really better if a procedure is found out to select the optimum value of η.

The second limitation will be deciding the initial values of the parameter vector which determines the entire path to optimization of the trajectories. We definitely have used the stochastic continuous algorithm to select the best possible initial conditions for the parameter vector but even the sample matrix from which the initial values are taken are defined by the user himself which also leans towards a trial and error process which shows us the inability of the solution to select its own initial optimum conditions.

The next limitation is the inability to declare the global optimum solution due to the sole non-convex behaviour of the system. Even if we obtain a better solution every time than the previous, it is unsure that any solution is the global optimum solution for the system.

The future work will surely be to find out an algorithm to select the optimum initial conditions for the system without any trial and error. An interesting task will be to find out a better cost function which shall make the system less non-convex so that we may have a chance to attain global optimality. Other work may include the objects of interest moving in a pre-defined motion themselves within a specific space. Also, future work may include the agent to perform more complex tasks rather than just surveillance.

Bibliography

[1] M. Zhong and C. G. Cassandras, "Distributed coverage control and data collection with mobile sensor networks," in *IEEE Transactions on Automatic Control*, vol. 56, no. 10. IEEE, 2011, pp. 2445–2455.

[2] J. Cortes and S. Martinez, "Coverage control for mobile sensing networks," in *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2. IEEE, 2004, pp. 243–255.

[3] N. Nigam and I. Kroo, "Persistent surveillance using multiple unmanned air vehicles," *Aerospace Conference*, pp. 1–14, 2008.

[4] P. F. Hokayem, Stipanovic, and M. W. Spong, "On persistent coverage control," *46th IEEE Conference on Decision and Control*, pp. 6130–6135, 2007.

[5] S. L. Smith, M. Schwager, and D. Rus, "Persistent monitorring of changing environments with limited range sensing," *IEEE International Conference on Robotics and Automation*, pp. 5448–5454, 2011.

[6] J. A. Miller, "A discrete adaptive guidance for a roving vehicle," *Guidance and Control Section, Jet propulsion Laboratory*, pp. 1–33, 1977.

[7] L. Fogarty and R. Howe, "Trajectory optimization by a direct descent process," in *Simulation*, vol. 11. IEEE, 1968, pp. 145–155.

[8] R. G. Brusch, "Trajectory optimization for the atlas/centaur launch vehicle," in *Decision and Control including the 15th Symposium on Adaptive Processes, 1976 IEEE Conference on.* IEEE, 1976, pp. 492–500.

[9] J. Snyman, *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms.* Springer Science & Business Media, 2005, vol. 97.

[10] B. Grocholsky and J. Keller, “Cooperative air and ground surveillance,” *IEEE Robotics and Automation Magazine*, pp. 16–26, 2006.

[11] P. Dames, M. Schwager, and V. Kumar, “A decentralized control policy for adaptive information gathering in hazardous environments,” *51st IEEE Conference on Decision and Control*, pp. 2807–2813, 2012.

[12] B. J. Julian, M. Angermann, and D. Rus, “Non-parametric inference and coordination for distributed robotics,” *51st IEEE Conference on Decision and Control*, pp. 2787–2794, 2012.

[13] K. Sugimoto, T. Hatanaka, M. Fujita, and N. Huebel, “Experimental study on persistent coverage control with information decay,” *SICE Annual Conference*, pp. 164–169, 2015.

[14] C. G. Cassandras, X. Lin, and X. Ding, “An optimal control approach to the multi-agent persistent problem,” in *IEEE Transactions on Automatic Control*, vol. 58, no. 4. IEEE, 2013, pp. 947–961.

[15] X. Lan and M. Schwager, “Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields,” *IEEE International Conference on Robotics and Automation*, pp. 2415–2420, 2013.

[16] K. Kawabata, L. Ma, J. Xue, S. Yokota, Y. Mitsukura, and N. Zheng, “Iterative polynomial-based trajectory extension for mobile robot,” in *Advanced Intelligent Mechatronics (AIM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 255–260.

[17] J. K. Arora and D. A. Pierre, “Optimal trajectories for multidimensional nonlinear processes by iterated dynamic programming,” *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 85–91, 1973.

[18] Y. Yao, P. Zhang, H. H. T. Liu, and F. He, “Optimall sweep-based persistent surveillance using multiple unmanned aerial vehicles with level of interest,” *World Congress on Intelligent Control and Automation*, pp. 2441–2446, 2012.

[19] E. C. Suicmez and A. T. Kutay, “Optimal path tracking control of a quadrotor uav,” in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 115–125.

[20] S. X. Yang and M. Meng, "Real-time collision-free path planning of robot manipulators using neural network approaches," *Autonomous Robots*, vol. 9, no. 1, pp. 27–39, 2000.

[21] J. Luh and C. Campbell, "Collision-free path planning for industrial robots," in *Decision and Control, 1982 21st IEEE Conference on.* IEEE, 1982, pp. 84–88.

[22] M. S. Branicky, "Studies in hybrid systems: Modeling, analysis, and control," DTIC Document, Tech. Rep., 1995.

[23] S. Engell, "Modelling and analysis of hybrid systems," *Mathematics and computers in simulation*, vol. 46, no. 5, pp. 445–464, 1998.

[24] K. M. Passino and Ü. Özgüner, "Modeling and analysis of hybrid systems: examples," in *Intelligent Control, 1991., Proceedings of the 1991 IEEE International Symposium on.* IEEE, 1991, pp. 251–256.

[25] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," in *Computer aided verification.* Springer, 1997, pp. 460–463.

[26] L. Wu and D. W. Ho, "Sliding mode control of singular stochastic hybrid systems," *Automatica*, vol. 46, no. 4, pp. 779–783, 2010.

[27] C. G. Cassandras and X. Lin, "An optimal control approach to the multi-agent persistent monitoring problem in two- dimensional spaces," in *IEEE Transactions on Automatic Control*, vol. 60, no. 6. IEEE, 2015, pp. 1659–1664.

[28] C. G. Cassandras, Y. Wardi, and C. Yao, "Perturbation analysis and optimization of stochastic hybrid systems," *European Journal of Control*, pp. 1–33, 2009.

[29] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[30] C. G. Cassandras and G. Bao, "Stochastic comparision algorithm for continuous optimization with estimation," in *Journal of Optimization Theory and Applications*, vol. 91, no. 3. JOTA, 1996, pp. 585–615.