Hierarchical Model Predictive Control of an Unmanned-Aerial-Vehicle Based Multitarget-Multisensor Data Fusion System

Peter William Sarunic
M.Eng., B.Eng., B.Sc.

Submitted in total fulfilment of the requirements
of the degree of Doctor of Philosophy

March 2012

Department of Electrical and Electronic Engineering

The University of Melbourne
Victoria 3010, Australia

To my wife Mirjana,

my daughters Adriana, Marina and Kristina,

and the memory of my parents, Venci and Neda.

Abstract

There has been rapidly growing interest in the development of increased automation in the military in recent years. In particular, the number of unmanned aircraft and ground vehicles being put into use is rapidly increasing. Concurrently, there has been an associated increase in the amount of research being performed to develop increased autonomy, moving from relatively simple remotely controlled devices to autonomous systems that are able to operate in a sense-think-act paradigm, i.e., robots.

An application of robotic technology that is of considerable military significance is that of detection and tracking of enemy assets. A key advantage of using autonomous vehicles in this application is that the locations and details of potential threats can be determined using relatively inexpensive unmanned vehicles with the operator of the system standing back at a safe distance. One example is the use of teams of unmanned aerial vehicles (UAVs) carrying passive direction-of-arrival sensors to detect and track enemy emitters such as radar-carrying platforms so as to enable reaction to the threats with other resources which could, say, include jammers or missile-carrying aircraft. In this thesis the problem of how to adaptively control the trajectories of UAVs in such an application in order to optimize performance in response to target measurements, while avoiding no-fly zones, is considered and a solution is developed.

Because of the complexity of situations that are encountered, a major issue is

how to formulate the problem in a manner which enables efficient computation of optimal behaviours for the platforms. In fact, an optimal solution cannot in practice be found by any physically implementable method. Hence, in this thesis an approach will be developed that enables implementation of a computationally feasible, albeit suboptimal solution, that takes into account both short-term and long-term goals. To this end, the problem will be addressed by developing a hierarchical control approach, incorporating an automated planner and a low-level (short-term) control algorithm.

A key aim is to use a consistent mathematical framework that can be generalized to a range of optimal control problems. As a result, all components of the controllers that are developed are based on concepts from estimation theory, dynamic programming and optimal control, giving a mathematically coherent and scalable solution.

To evaluate the effectiveness of the approach, first a controller is developed using an idealized UAV model and simulations are performed. Its performance is compared with a commonly used "myopic" control approach and found to give important improvements. Subsequently an improved planner is incorporated and tested, and then a version of the controller using a fixed-wing aircraft model for the UAVs is implemented. This version is also tested by simulation and found to perform successfully. Finally, a mathematical analysis of stability is commenced and significant headway made towards a stability proof.

Declaration

This is to certify that:

(i) the thesis comprises only my original work towards the PhD except where indicated,

(ii) due acknowledgement has been made in the text to all other material used,

(iii) the thesis is fewer than 100,000 words in length, exclusive of tables, bibliographies and appendices.

Peter William Sarunic

Acknowledgements

Some time ago, in the course of discussions with Prof. Bill Moran and Dr. Len Sciacca, the suggestion that I might study for a PhD arose. I was unsure at first, but after some discussions with my wife, Mirjana, who strongly encouraged me to take on the challenge, I decided to enrol in the course of study. Were it not for them, especially my wife, I might never have taken this path.

I would like to extend my sincere gratitude to my principal supervisor, Professor Rob Evans for his support and guidance. His broad knowledge and deep insight, as well as his helpful manner, were invaluable to my studies. I would also like to thank my co-supervisors Bill Moran and Stephen Howard; Bill in particular, who I believe, was the first to suggest I study for a PhD.

I am also very grateful for the vision and leadership of Dr. Len Sciacca, Chief of Electronic Warfare & Radar Division (EWRD), Defence Science & Technology Organization, in helping to initiate this work, and would like to thank Dr. Jackie Craig, subsequent Chief of EWRD, Dr. Alasdair McGinnes, Research Leader, Electronic Warfare Systems Integration Branch, and Dr. Damian Hall, Head, Distributed Electronic Warfare Group, for allowing me to continue on and complete this project.

I would like to thank my friends and colleagues for sharing thoughts, problems and ideas. In particular, I would like to thank Dr. Hatem Hmam, John Kitchen and Dr. Darren Bachmann.

Last, and by certainly all means most, I thank my wife, Mirjana, for her enduring love, patience and support.

Peter Sarunic,
Adelaide, South Australia,
March 2012

Publications

During the course of this project, the following publications and public presentations have been made, which are based on work presented in this thesis. They are listed for reference.

1. Peter W. Sarunic, Robin J. Evans and Bill Moran, "Control of Unmanned Aerial Vehicles for Passive Detection and Tracking of Multiple Emitters", *IEEE Symposium: Computational Intelligence for Security and Defence Applications*, 8-10 July 2009, Ottawa, Canada.

2. Peter W. Sarunic, Robin J. Evans, "Control of Unmanned Aerial Vehicles Performing Multiple Target Passive Detection and Tracking", *Fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 7-10 December 2009, Melbourne, Australia.

3. Peter W. Sarunic, Robin J. Evans, "Trajectory Control of Autonomous Fixed-Wing Aircraft Performing Multiple Target Passive Detection and Tracking", *Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 7-10 December 2010, Brisbane, Australia.

Contents

List of Figures

List of Algorithms

Abbreviations

AOA	Angle of Arrival
CE	Certainty Equivalent
CEC	Certainty Equivalent Control
CSN	Cooperative Sensor Network
DEW	Distributed Electronic Warfare
DEWSN	Distributed Electronic Warfare Sensor Network
DOA	Direction of Arrival
EA	Electronic Attack
EETS	Exhaustive Expansion Tree Search
ES	Electronic Support
ESA	Electronically Steerable Array
EW	Electronic Warfare
FIM	Fisher Information Matrix
GPS	Global Positioning System
HMM	Hidden Markov Model
JMLS	Jump Markov Linear System
MDP	Markov Decision Process
MILP	Mixed Integer Linear Programming
MPC	Model Predictive Control
OODA	Observe Orient Decide Act

PDF	Probability Density Function
POMDP	Partially Observed Markov Decision Process
SAR	Synthetic Aperture Radar
SPLAM	Simultaneous Planning Localization and Map Building
SQP	Sequential Quadratic Programming
TDOA	Time Difference of Arrival
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

In recent years there has been a surge of interest in the development of increased automation in the military, particularly in the United States. Large numbers of unmanned aircraft and ground vehicles are currently in production and their numbers are rapidly increasing. In addition to the increase in numbers of unmanned platforms there has been a large expansion in research being performed to develop increased autonomy, moving from relatively simple remotely controlled devices to autonomous systems that are able to operate in a sense-think-act paradigm, i.e., robots. These systems have sensors with which they gather information about their environment, computer processors to processes the information and make appropriate decisions, and effectors to perform appropriate physical actions on their environment. Increasingly powerful computer technology, the Global Positioning System (GPS), and a range of other technologies have made these robotic systems feasible and useful in the battlefield. In addition to the development of enabling technologies, their increased ability in recent years to observe, geolocate and then attack targets in hostile environments while keeping soldiers out of harm's way has been one of the key drivers for their proliferation. Not only are numbers of unmanned

systems rapidly growing but the range of types of systems is quickly expanding. Examples such as Predator and Global Hawk drones, Fire Scout unmanned helicopters, Protector robotic sentry motorboats and the TALON ground robot are just a sample of the wide range currently operating or in development. To acquire more of a flavour of developments in this area and their implications, the reader is referred to [51] for a discussion of the current state and likely future trends of robotics in the military. The article also goes on to describe the profound and far reaching implications on the method of execution of war that this technology is bringing about, as well as implications for future warfare and associated ethical issues.

An application of robotic technology that is of considerable military significance is that of detection and tracking of enemy assets. A key advantage of using autonomous vehicles in this application is that the locations and details of potential threats can be determined using relatively inexpensive pilotless vehicles with the operator of the system standing back at a safe distance. If in addition the sensors that are used are passive, the threat's location can be determined covertly with the enemy being none the wiser. One example is the use of teams of unmanned aerial vehicles (UAVs) carrying passive direction of arrival (DOA) sensors to detect and track enemy emitters such as radar-carrying platforms so as to enable reaction to the threats with other resources which might, for example, include jammers or missile-carrying aircraft. In this thesis, the problem of how to control the UAVs in such an application, so as to optimize performance in response to target measurements, is considered and a solution developed.

Because of the complexity of situations that are encountered and the resulting vast number of low-level options available to the platforms, a major issue is how to formulate the problem in a manner which enables computationally efficient determination of "optimal" behaviours for the platforms. Sensor

scheduling and control problems can be formulated and, in theory, be solved using dynamic programming; however, finding the optimal solution to these problems is in general computationally infeasible, so, in practice approximations and simplifications must be made to find a satisfactory suboptimal solution. In this thesis the problem will be addressed by developing a hierarchical approach where higher level and more long term measures of effectiveness are used by an automated planner to find a coarse-grained approximate solution that is then used by a low-level control algorithm as guidance in its determination of the fine-grained control inputs to the sensors. This approach is used primarily to enable a computationally feasible, albeit suboptimal, solution to a problem whose optimal solution cannot in practice be found by the direct dynamic programming method or for that matter any other implementable method.

The problem considered in this thesis, i.e., that of multisensor-multitarget platform steering, was chosen primarily because of a requirement for a solution to this problem by the author's employer. As it turns out, the problem is also appropriate for a PhD project because finding an efficient solution is a challenging task, requiring a substantial research effort for what is still an open area of study. In the following few paragraphs we will define the problem somewhat more precisely.

The specific problem that will be considered is that of a mission where multiple UAVs, which are carrying passive electronic support (ES) sensors, are performing the task of detecting and tracking multiple radar-carrying platforms which may be stationary or slowly moving. In an operational scenario the radar-carrying platforms could be either land or sea based. A scenario could for example involve a group of UAVs flying in formation in support of a sea based asset. The UAVs will be passively sensing the environment for radar emitters and tracking their locations. A centralized tracker, which could be based on a

land or sea based asset, or possibly one of the UAVs, will be performing the tracking, using data received from all the UAVs. As enemy radars are detected, the UAVs will be assigned to appropriate targets and will alter their trajectories to improve target position estimates in as short a time as possible. Any no-fly zones that may exist in the region will be taken into account in the determination of target assignment and trajectory optimization. The control/scheduling task will involve:

- detection of targets,
- tracking of detected targets,
- obstacle avoidance (e.g. no-fly zones, threats to sensor platforms),
- trajectory control of UAVs on the basis of measurements.

The emphasis of the work is on tactical response in military scenarios hence a key aim will be to develop efficient algorithms capable of interacting with the environment in real time. Another key aim is to use a consistent mathematical framework for the solution to the problem. To this end the solution will involve sensor fusion, planning, assignment and low-level control tied together using estimation theory, dynamic programming and optimal control to give a mathematically coherent result. One more point should be noted; the mathematical theory used in this thesis is quite general, hence much of what is learned in solving the problem considered here can be used for solving the other types of problems mentioned above as well as a range of other control/scheduling problems.

1.1 Thesis Outline

This chapter introduces the subject of research along with a summary of the context into which the research fits. It also gives a summary of the remainder of the thesis and summarizes the main contributions.

Chapter 2 reviews the current relevant literature and provides a description of the implications for the research that is the subject of this thesis. The review describes what has been achieved to date on the problem in question, what is lacking in current solutions, and also includes a more general component which is used to develop the approach that will be presented in detail in subsequent chapters.

Chapter 3 provides an outline of the background theory that is utilized to develop the algorithms in the subsequent chapters.

In Chapter 4, the problem to be addressed is more formally stated and split up into the subproblems of estimation and tracking, assignment, planning and control. The subproblems are then addressed and a solution to each developed. The algorithms developed are then coded and tested through simulation. In this chapter (i.e., Chapter 4), a highly simplified dynamics model for the UAVs is used in order to concentrate effort on the mathematical and algorithmic aspects of the subproblems mentioned, without delving into specifics relating to any particular type of UAV.

In Chapter 5, more detailed and realistic UAV dynamics are introduced and investigated. To do this, a specific class of UAV is considered, i.e., fixed-wing UAVs. Modifications and additions are then made to the algorithms that were developed in Chapter 4 in order to enable them to function effectively with the dynamics and constraints of the fixed-wing UAVs considered. The new modified algorithms are then tested by simulation to verify their effectiveness.

Chapter 6 addresses the issue of system stability from a theoretical point of view. The type of controller developed in this thesis is an extremely difficult one to analyse for stability, so the task has been broken into looking at components of the system, analysing them and inferring stability of the entire system from them. Both the initial idealized UAV model of Chapter 4 as well as that of Chapter 5 are analysed and results developed. At the time of writing, only a partial analysis has been done with further work required to fully analyse the control system.

In Chapter 7, the outcomes of this thesis are reviewed and discussed. This is then followed by a summary of future work to be done in order to further develop and analyse the algorithms.

1.2 Contributions

This thesis makes a number of novel contributions to the body of knowledge in the field of sensor scheduling and control; these are summarized below.

Firstly, a comprehensive review of the scientific literature on the problem being considered here and of related areas has been completed in Chapter 2. The review brings together battle-management concepts from the military and research on a number of related electronic warfare sensor management problems to demonstrate the motivation and context within which the problem being considered fits, what the state of current research is, and aid in the development of the solution approach. As a result of this review, it has been found that no other significant works have been presented in the literature that deal with the unique combination of issues dealt with in this thesis.

Chapter 3 is the result of a survey of more general literature in the area of optimal control. While presented as a background chapter, it is largely the

result of a broad survey of modern optimal control theory and techniques, with the aim of finding the most appropriate solution approach for the sensor management problem being considered. The chapter describes a range of optimal control algorithms, starting from the basic dynamic programming algorithm, which proves to be impractical for our problem, and ending with the hierarchical model predictive control algorithm that is developed later in this thesis. The primary contribution of this chapter is the concise bringing together of the various modern optimal control techniques that have relevance to our problem and demonstration of how they evolve from the basic theory.

In Chapter 4, a novel controller using a hierarchical variant of the model predictive control approach is developed to address the adaptive multisensor multitarget tracking problem summarized above. This consists of a fine-grained controller and a planner, organized in a manner that is new to this type of adaptive sensing problem. Also in Chapter 4, two novel path planning algorithms are presented and their performance assessed. Both algorithms are integrated into the hierarchical controller and the performance of the system tested by simulation and compared with a "myopic" controller similar to that commonly seen in the literature. Key features of the controller are the incorporation of long-term goals utilizing a dynamic programming based formulation and its usability in real-time systems.

The problem considered is then developed further in Chapter 5 with the dynamics of a specific type of sensor platform being incorporated, i.e., fixed-wing aircraft, and the requisite additions made to the control algorithm to enable robust performance. The controller is again tested by simulation and shown to perform effectively, further demonstrating the effectiveness of the approach and usability in a practical distributed sensing application.

An important property of any control system is stability. In Chapter 6 a

mathematical analysis of stability of the newly developed controller of Chapter 4 and the version with fixed-wing aircraft dynamics of Chapter 5 is commenced. It is found that stability analysis of the controllers is a highly challenging task; however significant inroads are made by focusing on components of the control systems. Together with the results of simulations in the chapters 4 and 5, the analysis gives further confidence that the controllers should be stable. This being said, further mathematical analysis is required to give a deeper understanding of the behaviour of the controllers and more confidence in their overall stability.

Chapter 2

Literature Review

A literature review was performed with two aims in mind: (1) to determine what research has been done in related areas and what the inadequacies of the current solutions are, in order to determine and address the specific problem to be considered here, and (2) to determine and gain a knowledge of relevant areas of mathematics as well as gain new ideas on how to approach the problem. With regard to the first aim, a survey of published papers was performed, primarily in the areas of optimal control and scheduling with a particular emphasis on application to distributed sensors and autonomous airborne vehicles. For the second aim, texts on relevant areas of mathematics and engineering were read and analysed in order to gain a solid foundation in theory that could be used to address the task in question.

2.1 Summary of Published Papers

The following subsections, in a step by step fashion, will first describe the general class of problems that the problem considered here falls within. The general motivations and the types of solution approaches that have been used to solve

these optimization problems will then be described. This will be followed by a grouping/classification of the problems to show how our problem fits within the general context, and finally the general approach that will be used for our problem will be proposed.

2.1.1 Distributed Electronic Warfare Sensor Networks

In the military context, the task that is being considered here falls within the class of problems associated with distributed electronic warfare sensor networks. A general description of some modern trends in distributed electronic warfare sensor networks (DEWSN) can be found in [48]. The most relevant parts of this paper will be summarized below in some detail as the system that is developed in this thesis is a type of DEWSN and looking at the bigger picture has significant bearing on the solution approach chosen here.

The management of electronic warfare (EW) sensor and jamming assets is a major challenge for modern military forces. To achieve dominance of the radio frequency spectrum with the aim of enabling control of the battlespace requires the integration of multiple EW and surveillance sensors, communication networks as well as electronic attack measures. Sensors are now multi-tasking with the ability to rapidly and automatically switch modes in order to adapt to mission requirements. For example, radars with electronically steerable arrays (ESA) can switch between search and track modes and change their array steer direction almost instantly to track known targets while searching for new ones. Similarly synthetic aperture radar (SAR) systems on unmanned aircraft can perform multiple tasks in real time. These systems are called multi-function systems with each function being associated with a *mode* of operation and the selection of which mode to operate in is referred to as *sensor mode scheduling*. Even within a mode, such parameters as transmit energy, transmit waveform

type and beam time on target can be allocated and varied to accommodate mission requirements.

Single sensor scheduling and adaptation, of which the two above cases are examples, is a mature technology, however, techniques for the optimal scheduling and control of systems across a network of sensors are not as well developed. A key difference is that the scheduling and control of resources is carried out across a communications network which generally suffers from more stringent constraints on data throughput as a result of limited communications bandwidth. Clearly in this type of distributed system the scheduling of the transmission of data between components of the system is an important component of the scheduling problem. Another difference is, in general, an increase in the number of parameters that can be controlled, with the possibility of having multiple disparate sensors whose relative geographic position can also be controlled, themselves each having multiple modes of operation. This clearly provides the potential for improved sensing performance, but with this added flexibility of course comes the price of a more challenging scheduling and control task. However, as with the single sensor case, the distributed sensor scheduling problem is essentially a constrained optimization problem and in that sense is similar to the single sensor problem, with similar techniques being applicable.

In [48] the concept of the cooperative sensor network (CSN) is introduced as a framework for the management of data, sensor modes and other associated sensor parameters in a network of EW sensors. The CSNs described in the paper have as their core a sensor scheduler around which architectures can be constructed to enable the building of a range of systems which can include radar systems, ground sensor networks and distributed EW systems. The concept is also flexible enough to be used for the determination of flight paths of teams of UAVs with on-board angle-only ES sensors, for optimal target acquisition;

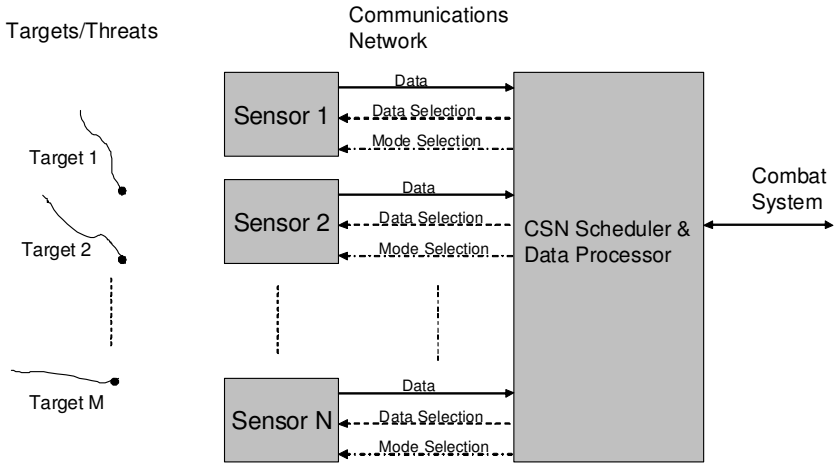

Figure 2.1: Basic Cooperative Sensor Network Concept

this of course is the subject of this thesis. The core concept of a CSN i.e., that of a sensor/data scheduler is shown diagrammatically in Figure 2.1. Note that the figure is a little different to that in [48]; this was done to emphasise the sensor mode/parameter selection aspect of the CSN which was missing in the corresponding figure in [48].

The basis of the CSN architecture is a sensor scheduler which optimizes system performance by computing an optimal sensor scheduling policy for the given target behaviour and subject to the prevailing constraints. Some key benefits of a CSN are:

- Tracking performance is improved in regions of sensor overlap. Tracks can be initiated sooner, be more reliable and of higher accuracy than those of the individual sensors.

- Sensor geometry can be optimized to yield different views of targets, aiding in the detection of low observables or targets whose view may be obstructed for individual sensors.
- Individual sensor performance requirements may be reduced through optimal placement of sensors.
- Individual sensors operating in different parts of the spectrum can provide a multi-spectral capability thus enabling detection of a wider range of targets.
- Tracks can be cued and handed over between sensors, and thus targets can be followed over wider geographic regions.

The concept of a CSN can also be extended to include cooperative ES and electronic attack (EA). Adding effectors (jammers) or sensor/effectors (combined sensors/jammers) to the list of sensors in Figure 2.1, with appropriate changes to the associated communication link with the scheduler, is easily accommodated in the CSN concept. EA is used to disrupt enemy surveillance functions and information gathering/sharing capability, but can also potentially interfere with friendly surveillance and communications assets. Hence coordination of distributed EW assets is paramount if friendly RF assets such as sensors and communication systems are not to be adversely affected. A CSN can be used to perform this coordination; additionally the CSN can optimize resource use so that the least number of assets are required to perform the coordinated sensing/jamming function.

An important point brought up in [48] (and earlier in [49]) relates to battlespace dynamics. A close coupling exists between the management of sensors in the battlespace and the ability to engage threats with available weapon systems. Clearly dynamic optimization of sensor parameters and modes can lead

to more rapid and appropriate target engagement. Figure 2.2 (extracted from [48]) is a representation of the cycle of battle as an Observe-Orient-Decide-Act (OODA) loop. The small rectangles added to the figure represent breaks in the automated components of the loop that would occur when non-networked sensor data is reported either manually or semi manually. The effect of the manual handling, and processing of data is that OODA blockages are introduced, resulting in an increase in time required to assimilate information, make time critical battle decisions and hence reduced warfighting tempo and hence effectiveness. A CSN achieves an increase in OODA loop tempo by removing the manual process of data assimilation as well as by optimizing the sensor parameters and modes in a way that would be infeasible to perform manually. Put concisely, *the purpose of a CSN is to increase the rate at which knowledge can be gained and exploited to increase the tempo of battle and thus overwhelm the enemy's ability to react to friendly actions.* Given this aim, the motivation for a considerable effort in solving as effectively as possible what is a complex and difficult set of constrained optimization problems is clear.

As possible further reading on the topic, the reader is also referred to [49] where a more general description of the concept of a cooperative sensor network (CSN) is provided. This paper, while also covering some DEW concepts looks at how CSNs can more generally aid in the battle management process and in particular shorten the time of execution of the OODA loop.

2.1.2 Electronic Warfare Sensor Management Solution Approaches

This section looks at the solution approaches that have been tried for various types of multimode sensors and CSNs, the aim being to help determine the

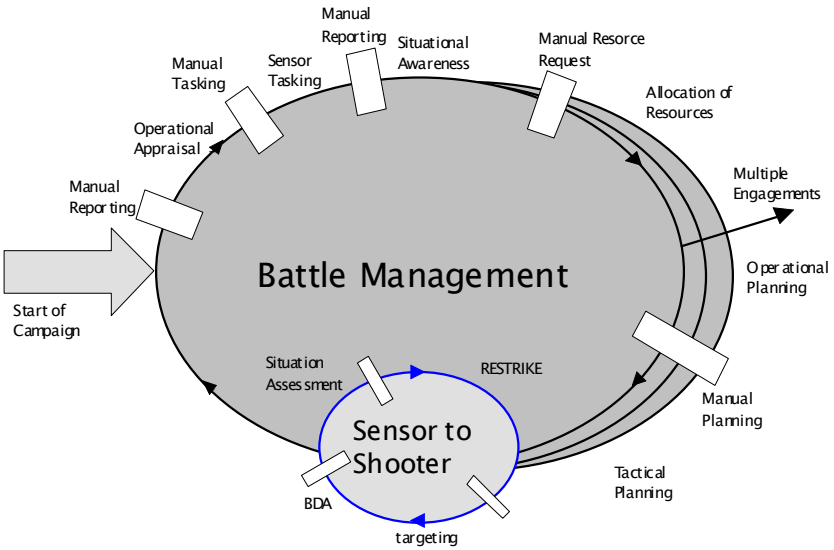

Figure 2.2: The "broken" OODA Loop

appropriate way ahead for the problem being considered in this thesis.

Modern military radars are increasingly using phased array antennas, have increased computing capability and more flexible transmitter hardware, giving them increased capability to adapt their beam steer direction and transmit waveforms to optimize performance. Unlike mechanically scanned radars which have to contend with antenna inertia, phased array radars can re-direct their beam in any direction almost instantaneously. One outcome of this is that phased array radars can switch between the tasks of tracking existing targets and acquiring new targets equally quickly. This enables the separation of the task of searching for new targets and that of updating existing tracks and thus the optimization of both separately and individually. In [34] the task of optimizing the radar transmit waveform for new target acquisition is considered and a solution approach presented. In that paper the adaptive waveform scheduling

problem for new target detection is posed as a partially observed Markov decision process (POMDP) and solved using a stochastic dynamic programming approach. POMDPs and stochastic dynamic programming will be described later in this thesis, but for now what is most relevant is that this is the technique that was used. Solutions to this type of problem are optimal control policies that optimize an objective (cost or reward) function. The adaptive waveform optimization problem for target detection thus becomes the selection of which sequence of waveforms to use to maximize the overall rewards of target detection. With appropriate choice of reward function, the time taken to detect new targets and the radar resources used can thus be optimized. In [34] a short optimization horizon was used, the justification being that only the detection of new targets problem was considered and the number of radar dwells used to confirm a new track is typically very small.

Another paper that looks at a multifunction phased array radar scheduling problem is [58]. In [58] an algorithm that optimizes on two criteria in order to schedule radar beam steer actions is presented. The algorithm has the dual aim of detecting new targets (surveillance mode) and maintaining tracks on known targets (track or revisit mode), treating the resulting sensor management problem as two separate subproblems, one for each radar mode. For the surveillance mode, the objective (reward) function for selecting the next beam steer direction is the expected number of new targets within the beam and for the revisit mode the objective function is the expected information gain. The problem is formulated as a stochastic control problem and a myopic (i.e., one step look-ahead), dynamic programming based scheduling scheme is developed. As part of the solution a tunable parameter which they refer to as the "revisit ratio" is defined and used to provide control of the distribution of resources (i.e., radar beam dwell time) between the two radar modes.

A paper by Krishnamurphy and Evans [32] also considers the electronically scanned radar beam scheduling problem. This paper presents a dynamic programming based solution in which the beam scheduling task is formulated as a multi-arm bandit problem involving a hidden Markov model (HMM) for the targets. The multi-arm bandit approximation of the system is implemented as a way of computing long-term optimal beam scanning schedules while avoiding the exponential complexity of more general POMDP models. The paper describes the algorithm in considerable detail including the assumptions made in using the multi-arm bandit formulation and the range of applicability of the solution.

When a network of distributed sensors is used, scheduling data transmission between components of the system via the network communication system can result in significant performance improvements. This is because in a distributed system there are generally constraints on data throughput as a result of limited communications bandwidth. Scheduling of the actual activation of the individual sensors may also have pay-offs, as there may be costs associated with power usage, interference with other sensors, or for example, in military applications some sensors may have higher likelihood of being detected by the enemy than others. One paper that has application to this problem is [31], which considers the problem of choosing which one of a number of sensors to select at each time instant to provide the next measurement of a target's state. The paper uses a HMM based formulation of the problem to develop dynamic programming based solutions. Both optimal and suboptimal scheduling algorithms are developed and numerical examples provided to demonstrate performance. While the problem considered involves the selection of only one sensor at a time, the solution approach can be generalized to the selection of multiple sensors at any time instant. The paper also suggests a generalization from the use of HMMs

for the targets to implementing jump Markov Linear Systems (JMLSs) in the target model; this generalization is described in [14] which will be summarized next.

A paper by Evans, Krishnamurthy, Nair and Sciacca [14] describes sensor and data rate control algorithms for tracking maneuvering targets. It first gives a short description of an algorithm for the scheduling of flexible multimode measurement sensors and then, in considerable detail, presents algorithms for scheduling data transfer across data links between multiple sensors and a tracking computer. Both problems are solved within the framework of a partially observed stochastic control problem. For the case of the multi-mode sensor problem the paper considers the example of beam scheduling in electronically scanned radars and represents the problem with a multi-armed bandit structure (as in [32], which was reviewed earlier, but now using a JMLS target model in place of a HMM). For the data transfer scheduling case, the problem of target tracking in a multiple radar network, with a limited-data-rate communication channel to the central tracking computer, is considered. For this problem, first the optimal sensor-data scheduling algorithm, which is developed using stochastic dynamic programming, is presented. This algorithm uses a finite horizon cost function, which is based on the outputs of an adaptive tracker, to optimally select which sensor's data is to be transmitted to the tracker at each tracker update. As the optimal algorithm is not practically implementable because of computational complexity, a practically implementable suboptimal algorithm which optimizes instantaneous cost is then developed. Simulations that demonstrate performance improvements resulting from data rate control are also presented.

An article by Sciacca et al. [50] considers the problem of the localisation of multiple targets using a CSN with communication bandwidth constraints.

It discusses two examples (i) a radar network and (ii) unattended ground sensors with angle-only measurements. The paper outlines a suboptimal scheduling algorithm for the unattended ground sensor problem and presents results of simulations of the system. The algorithm that is used is also based on a stochastic scheduling/control approach.

In [21], sensor scheduling in a multiple sensor network is investigated. The particular problem considered is that of selection of which sensor to activate over time, trading off sensing performance with sensor usage costs. The problem is formulated as a stochastic optimal control problem, and a suboptimal approximation is found to the long term optimal solution. Two key features of the approach used are that it takes into account both long-term and short-term costs and benefits, and secondly that it does not rely on analytical tractability of the system under consideration. Sophisticated models for sensor behaviour and target dynamics can be modelled because of the use of a Monte Carlo approach. The scheduling algorithm that is developed uses a particle filter to deal with the non-linear aspects of the problem, with the long term expectations of costs calculated by averaging over a large number of particles. Policy roll-out [5, p. 314] is used as the Q-value approximation method [5, p. 320]. As an example problem, a simulation of a radar network and a single target, with only one radar selected at a given time, is presented. The algorithm is shown to give improved performance over a simpler and commonly used "closest point of approach" algorithm. In its conclusions the paper suggests a future direction of work could be the more general sensor management problem, including sensor geometry control, sensor bandwidth allocation, sensor mode switching, as well as sensor scheduling. Other possible constraints into the POMDP formulation suggested are battery capacity, load balancing, and bandwidth limits.

2.1.3 Solution Approaches for Problems Involving Sensor Platform Trajectory Control

As UAV trajectory control for the optimization of sensor geometry in a CSN is the central theme of this thesis, let us now look at work that has been done that is specifically about controlling trajectories of UAVs for the purpose of sensing. This will then be related to the work on control of the parameters described above and be used as an aid in developing the algorithm that is the subject of this thesis.

A paper by Oshman and Davidson [45] considers the problem of bearings only target localization, i.e., the estimation of the location of a *fixed* target using a sequence of noisy bearings measurements. The paper considers a single sensor, single target system and presents three methods of computing optimal trajectories for the observer (i.e., sensor). The methods are direct optimal control methods, which are based on parameterizing the control history using a fixed set of constant parameters (the discrete-time control inputs), thereby converting an infinite dimensional control problem into a finite dimensional parameter optimization one. The methods presented take into account long term costs by considering a predetermined number of discrete-time measurements and optimizing on those measurements. The first two methods rely on gradient-based numerical techniques and the third method is formulated so as to be solvable using non-linear programming, and a gradient descent method is then used to solve the non-linear programming problem. The third method (i.e., the differential inclusion method) has the added feature that it can incorporate both state and control inequality constraints in the solution. The measure that is used as the reward in the optimization is the determinant of the Fisher Information Matrix (FIM). The reasoning behind using the determinant of the FIM

is as follows. According to the Cramér-Rao Lower Bound Theorem, if we have a nonrandom parameter $\mathbf{x}_0$ and an unbiased estimator of the parameter $\hat{\mathbf{x}}(\Theta)$ where Θ is the set of measurements, the estimation error covariance matrix P is bounded from below as follows:

$$P = E\left\{(\hat{\mathbf{x}}(\Theta) - \mathbf{x}_0)(\hat{\mathbf{x}}(\Theta) - \mathbf{x}_0)^T\right\} \geq M^{-1} \tag{2.1}$$

where M is the FIM and is given by

$$M = -E\left\{\frac{\partial^2}{\partial \mathbf{x}^2} \log p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})\right\}_{\mathbf{x}=\mathbf{x}_0} \tag{2.2}$$

and $p_{\Theta|\mathbf{x}}$ is the conditional probability density function. In the case of an efficient estimator, equality holds in Equation 2.1. The covariance matrix P is positive semidefinite and its associated quadratic form defines a hyperellipsoid whose semiaxis sizes are defined by the eigenvalues of P and semiaxis orientations are defined by the eigenvectors of P. For the two-dimensional case the one-sigma area of the resulting ellipse is

$$A_{1\sigma} = \pi\sqrt{\det P} \tag{2.3}$$

The optimal trajectory is defined in [45] to be the one that results in the minimum value of $A_{1\sigma}$ in Equation 2.3. Now, from Equation 2.1, for an efficient estimator the error covariance matrix and the FIM are inversely related, resulting in the one-sigma uncertainty region for the two dimensional case being expressible as

$$A_{1\sigma} = \frac{\pi}{\sqrt{\det(M)}}$$

Hence, maximizing $\det(M)$ results in the minimization of $A_{1\sigma}$ and thus optimization of the trajectory. Note that because the Fisher information matrix is dependent on the target position, an initial estimate of the position must be made, with the optimality of the calculated trajectory being dependent on the

validity and accuracy of this initial estimate. It should also be noted that the techniques presented in [45] cannot deal with manoeuvring targets (i.e., target tracking) and cannot incorporate prior information, in the form of a probability distribution, in target position. Additionally, all three techniques are suboptimal as they all rely on a gradient based solution process which cannot in general be guaranteed to converge to a global minimum.

A paper that covers optimization of multiple passive receiver trajectories for stationary target localization is [12]. This paper again considers a single stationary target, but instead of using angle-only sensors, it describes a technique called scan-based localization. For a radar with a constant scan rate (typically mechanically scanned radars with rotating antennas) the times at which the radar beam is intercepted by a pair of passive sensors can be used to calculate the angle subtended by the pair of sensors about the radar's position. With more than two sensors the collection of subtended angles can be used to calculate the radar's position, thus the term scan-based localization. As with [45], this paper presents an algorithm that is based on maximization of the determinant of the FIM; however, in this case there is no attempt at long term optimization. The algorithm presented simply performs one-step-ahead (myopic) optimization by using a simple gradient descent approach. Because of the use of a myopic cost function that relies entirely on local topology the trajectories that this technique produces can be quite non-optimal in the long term sense. To compensate for this, an ad-hoc factor is added to the cost function that makes the sensors move closer to the target more quickly than would be the case when the FIM maximization is simply used by itself.

The paper (i.e., [12]) also presents an ad-hoc method for avoidance of zones that the sensor platforms may be barred from entering. For example these zones may be regions where threats to the platforms may exist. The method it

proposes is to add a multiplicative factor for each "threat" zone centre to the cost function; this serves to steer the sensors away from the centre of that threat zone. The method, as presented, requires the threat zones to be circular. In order to stop the sensors from coming too close to the target, another addition is made to the algorithm; when an update is computed for the next sensor position, if the update would bring the sensor within a minimum radius around the target, the updated position is projected away from the target onto the circle representing the minimum radius, thus avoiding the possibility of coming too close to the target.

As was the case with [45], the approach presented in [12] cannot deal with manoeuvring targets (i.e., target tracking) and, as it relies on a gradient based solution technique with only a local cost function used at each update, it cannot in general be guaranteed to converge to a global minimum. The ad-hoc nature of the factor that is introduced to make the sensors move closer to the target more quickly and thus make the trajectories "more optimal" in the global sense is also very limiting from the point of view of adding any insight into the task of finding the long-term optimum trajectories and doesn't address the issue of avoidance of the sensors becoming trapped in local maxima.

In [56], research on coordinating multiple UAVs to search for, detect and locate mobile ground targets that are emitting radio frequency signals is described. Each UAV carries a passive sensor that collects relatively coarse angle-of-arrival measurements. A Kalman filter is used to compute target location estimates and an ad-hoc control algorithm is used to coordinate the motion of the UAVs so as to further improve the location estimates. To address the general problem considered, i.e., searching for, detecting and locating emitters, a distributed control architecture, using a state machine for each UAV to determine its operating state, is used. The paper gives a short description of the state machine which

has four states, i.e., Global Search, Approach Target, Locate Target, and Reacquire Target, and then goes on to describe in more detail the algorithms used in the Locate Target state. The algorithms that are described are not actual optimization or optimal control algorithms. Instead, Toussaint et al. prescribe predetermined desired behaviours for the UAVs and apply ad-hoc control laws to make the UAVs behave in the desired way; no-fly zones are not considered in the formulation.

In [17], Frew, Dixon, Argrow and Brown outline the development of a networked UAV communication, command and control architecture. Of particular interest here, their efforts to solve the radio localization problem, where one or more UAVs react cooperatively to determine the locations of radio emitters, are introduced. In their approach, emitter localization is cast as a stochastic distributed estimation problem and UAV mobility is used to optimize target position estimation accuracy by maximizing the FIM. In their problem formulation, a team of N UAVs and M radio emitters are considered. The motion of the UAVs, which are fixed-wing aircraft, is represented by a simple planar kinematic model involving vehicle speed (which is constant), heading and turn rate. The turn rate can take on discrete values up to the maximum turn rate $\omega_{\max}$. The radio emitters, which may be stationary or moving, are each assumed to be transmitting omnidirectionally and a propagation model is used to determine the power received by each sensor from each emitter. The receivers on each UAV can only measure received power; two cases are considered: (1) the receivers can only measure the total power received, and (2) the receivers can distinguish between transmission from different sources, and thus measure the individual signal powers received. The radio localization problem is then broken down into two subproblems, the first being the estimation of the state of all the transmitters, and the second being the determination of the control input (in this case turn rate)

for each UAV that optimizes some combination of performance and information criteria. At the time of writing in [17], the radio source localization estimator had not yet been developed, but it was stated that a maximum-likelihood estimator would be designed in future that fuses synchronous measurements from the multiple distributed UAVs and that multiple-model and hybrid estimation techniques would be incorporated when the number of emitters is also unknown. Their solution to the second subproblem, i.e., that of calculating the optimal control inputs, involves the use of a distributed receding horizon control policy. Receding horizon control will be described in more detail later in this thesis; however, for the purposes of understanding this paper a short summary is given here. Let us assume that the state of one of the UAVs is $\mathbf{x}(k)$ at time k, then the main steps in the receding horizon control policy at time k for that UAV are (1) calculating the optimal control sequence $\{u^*(k+1), ..., u^*(k+T_h)\}$ over a finite planning horizon $2 \leq T_h < \infty$ and (2) implementing the optimal control input(s) over some control horizon $k \leq t \leq k + T_c$ where T_h and T_c are integers satisfying $1 \leq T_c \leq T_h$; then at time $k + T_c$ steps 1 and 2 are repeated, and so on. In [17], the basic RHC policy is extended to multiple UAVs through a cooperative process in which distributed control is achieved by iterative consensus between the individual receding horizon controllers. The optimization presented in [17] uses an optimization that for each UAV minimizes a cost function that has three components, i.e., (1) a *control* cost which limits the control input, (2) an *uncertainty* cost which encourages the UAV to improve its knowledge of the state of the M targets, and (3) a *distance* cost which keeps the UAV some distance away from the emitter position estimates, both for safety and to avoid mathematical singularities in the estimation equations. In the calculation of the uncertainty cost, a version of the FIM that takes into account target process noise is used to compute the future covariance matrix for each emitter at each

time increment over the planning horizon; the *traces* of the covariance matrices are then summed up to arrive at the uncertainty cost. The reader is referred to equations (6) to (16) in [17] for the detailed cost computation equations.

The solution to the receding horizon optimization problem described above is obtained through an exhaustive search over some discretized input space. This method has the advantage that it results in the optimal solution for a given planning horizon, avoiding any issues relating to convergence to local minima. Unfortunately, as the planning horizon is increased, the amount of computation required grows extremely quickly (exponentially), necessitating very short horizons. In [17] the planning horizon was limited to 2-4 sample times. This very short planning horizon to a large extent negates most of the advantages of formulating the problem to include future costs as opposed to simply using a gradient descent technique. A point to note also is that no-fly-zones (using inequality constraints) are not considered in the formulation. While, in principal, they could be incorporated using the above approach, the short horizons that have to be used would make incorporation of no-fly-zones problematic. This is because, with short horizons, avoidance of UAV entrapment behind no-fly-zones (in local minima) cannot be guaranteed, and also the UAV dynamics, which are limited by maximum turn rates, may not be able to be allowed for, i.e., the UAVs could move into positions that they may not be able to manoeuvre out of without violating their dynamic constraints.

A paper by Leung et al. [38] describes work that applies the receding horizon control approach to the problem of trajectory control of autonomous robots gathering information. Note that in this paper the term model predictive control (MPC) is used for the control approach; receding horizon control and model predictive control are simply two alternate names for this type of control

algorithm. In this paper, two types of problem are considered (1) simultaneous planning, localization and map building (SPLAM) and (2) multi-robot geolocation. In both problems the information gathering task is formulated by estimating a state vector containing features of interest in the environment. In the SPLAM problem the estimation is performed by using an extended Kalman filter, whereas for the multi-robot geolocation problem an extended information filter is used. Because of its similarity to the problem considered in this thesis, we shall concentrate on the second problem, i.e., multi-robot geolocation; the summary below thus relates to that problem. It should be noted, however, that very similar approaches are used for both problems, the difference mainly being in the details.

The planning/control problem for the information gathering task is formulated as an optimal control problem and the MPC approach is used to find a suboptimal solution. The task which is optimized is the localization of targets as quickly as possible from a sequence of observations. This is done by applying controls so as to maximize the minimum eigenvalue of the information matrix, with constraints on the robots motion being incorporated into the optimization process. The features (i.e., the targets, in the multi-robot geolocation problem) are assumed to be stationary and it is assumed that the poses of the robots are obtained from an external source such as GPS. No-fly zones of the type considered in this thesis are not incorporated in the problem formulation, however, minimum distances from targets are enforced; they refer to these as no-go zones. In their simulations the UAVs were assumed to be equipped with cameras as the sensors, and the controls available to the controller were the roll angles of the UAVs.

Two optimization strategies are used, i.e., (1) Exhaustive Expansion Tree

Search (EETS) and Sequential Quadratic programming (SQP). The EETS performs an exhaustive search among a limited number of control sequences. As stated in the paper, if for each robot the number of possible control options is N_ω, the number of steps in the optimization is N and there are n robots, the information gain for $(N_\omega)^{nN}$ options needs to be evaluated at each update of the controller. The number of options to be evaluated hence rapidly becomes extremely large as N is increased (as N is a factor in the exponent); and as a result *the optimization horizon N is limited to small values.* They also use SQP together with the EETS, by first doing a coarse optimization using the EETS and then fine tuning their solution with SQP. The reasoning is that, while SQP is an efficient method for continuous optimization problems, using SQP with a random initial guess for the optimal solution will often lead to a local optimum being found. Using the EETS first is thus expected to provide a good initial guess that can then be refined using SQP.

To increase the planning horizon Leung et al. suggest a strategy where the controls are not changed at every step of the optimization calculations. For example, to double the planning horizon without increasing the amount of computation, they allow changes in control options at every second step. This of course increases the coarseness in the optimization and is quite limited in its capability to effectively increase the planning horizon. They also argue that they have shown in their SLAM simulations that there is a limit to the benefits gained by increasing the planning horizon, especially for systems with large uncertainties, where long-term rewards may not be realized. To some extent the argument holds for their problem; however, for the problem considered in this thesis, where there are no-fly zones that must be avoided, whose positions are known from the outset, and where the range of the sensors is much greater than the distance travelled by the UAVs over a small number of steps of the

controller, much longer horizons than are possible with the technique described in [38] are highly desirable.

In [54], Singh et al. present a sensor scheduling algorithm for the case when the state, observation and action (control) spaces are all continuous, one application clearly being the observer trajectory optimization problem. The major goal of the work presented is to solve this problem directly, without recourse to discretization of the state, observation or, in particular, the action space as is often done in other work presented in the literature. Additionally, no assumptions of linearity or Gaussianity are made, so as to make the solution approach as general a possible. To facilitate solution without these assumptions, they resort to a sequential monte-carlo approach, i.e., a particle filter. In their solution approach, the system is modelled as a continuous-state HMM, with the observations also assumed to be continuous. In order to allow the action space to be continuous, an iterative, stochastic gradient technique is implemented to optimize the action sequence. To develop the technique, a performance criterion is defined, whose gradient with respect to the action sequence is first derived, and then a method for computing low-variance estimates of the gradient is demonstrated. These estimates of the gradient are used in an iterative, simulation (i.e., particle filter) based optimization of the action sequence. Using this optimization, a local optimum of the performance criterion is computed.

As an example of a sensor scheduling problem, Singh et al. then consider the specific case of observer trajectory planning for the case of multiple observers and a single target. To demonstrate the utility of their algorithm for this application, they performed simulations for the one and two observer case. They used a relatively short optimization horizon of $N = 7$ in all their simulations. Results concerning convergence of the algorithm were derived mathematically as well as through demonstration in their simulations.

While an interesting and informative study from a theoretical point of view, for the problem being considered in this thesis, [54] has some serious shortcomings. Firstly the technique is computationally very intensive. For the simulation examples presented, Singh et al. cite simulation times of several hours on a 2.8 GHz Pentium 4 CPU. Importantly, this is for scenarios that are much simpler than those that will be dealt with in this thesis, and also for relatively short optimization horizons. Despite the high computational requirements, the algorithm only finds a local minimum of the cost function as a result of the use of the gradient based technique. When dealing with multiple observers and multiple targets, it is quite plausible that the algorithm could converge to a local minimum that is a very poor solution. Also, the problem that Singh et al. have solved does not include hard constraints for avoidance of obstacles such as no-fly zones, and the algorithm they use appears to not be particularly amenable to the addition of this type of constraint.

A paper by Hanselmann, Morelande, Moran and Sarunic [20] presents algorithms for scheduling passive sensors (mounted on UAVs), which are detecting and tracking radars. Two types of sensors are considered i.e., angle-of-arrival (AOA) sensors and time-difference-of-arrival (TDOA) sensors. The algorithms model the system as a POMDP in a stochastic dynamic programming formulation of the scheduling problem. An interesting aspect of the algorithms is the application of information theory through the use of Rényi information divergence between posterior and prior probability density functions (PDFs) as the reward function in the optimizations. The reasoning behind the use of this measure is that it has the advantage of permitting target detection and estimation to be handled in the same framework. One-step-ahead algorithms for scheduling angle of arrival vs. time difference of arrival measurements, as well as for trajectory control of UAVs are presented. Multi-step-ahead formulations are also

discussed. A significant aspect of the work is that a sophisticated tracker, which deals with some real-world issues in passive angle-only tracking, is used. The use of information theoretic concepts for making the reward functions more general could possibly be further developed to advantage, but care must be taken to ensure that the increase in "information" that is maximized accurately reflects performance parameters that are relevant to the application. A major drawback of the approach is the amount of computation required to find the optimal solutions. The one-step-ahead algorithms don't deal with long term costs of actions while still being quite computationally intensive; the multi-step-ahead version, although it can be formulated in theory, is even more severely limited by computational requirements, to the extent of not being practicably implementable for anything but quite short horizons. As long term optimization is in general very important for optimal trajectory control, this limitation is quite serious for this application.

Bellingham, Richards and How [3] present a trajectory optimization algorithm for autonomous fixed-wing UAVs. The case of a single UAV flying a trajectory to a known goal position in two dimensional space, with obstacles (no-fly zones) in the flight path, is considered. Note that the algorithm they present does not involve sensing, but is included in this survey because the way it performs trajectory control is relevant to this thesis. The most important feature of the approach that they present is that it computes a long-range dynamically constrained UAV trajectory, which avoids no-fly zones, from the UAV's starting point to the final goal. To achieve this, while maintaining the amount of computation required at a workable level, the authors of [3] have split the trajectory optimization problem into two sub-problems, producing an algorithm comprising a cost estimation phase, which in a coarse manner considers

the entire UAV path to the goal, and a trajectory design phase, which incorporates a receding horizon control strategy with relatively short term goals. In the cost estimation phase, a cost map of minimum time-to-go from a limited set of points to the final goal is produced. This is done by creating a visibility graph which is then searched using Dijkstra's single source shortest path algorithm [10]. The resulting cost map takes into account the requirement for the UAV to avoid the no-fly zones, which are assumed to be the sum of one or more rectangles. The cost estimation phase is performed once for a given goal position and obstacle field, but needs to be repeated if the environment changes. The cost map information is used in the terminal penalties of the receding horizon optimization i.e., the trajectory design phase, to design a series of short trajectory segments that are followed until the goal is reached.

The division of computation between the cost estimation and trajectory design components is performed in such a way that the trajectory optimization phase can be formulated using only linear and binary variables. Then in the trajectory design phase, mixed integer linear programming (MILP) is repeatedly applied in a receding horizon strategy to compute the UAV trajectory from starting point to the final goal. Note that by using MILP, difficulties associated with nonlinear programming such as having to choose a suitable initial guess for optimization because of the possibility of converging to a local minimum are avoided.

From the point of view of the aims of this thesis, the key feature of the approach used in [3] is that it has a planning horizon right up to the final goal, which enables it to deal with no-fly zones in a way that avoids entrapment of the UAV behind them. The trajectories it produced in simulations were also not far from optimal while still computationally realistic for the problem considered, which is a significant achievement in this type of problem. However, for the task

set in this thesis, the algorithm still falls short of the mark. While the algorithm manages to reduce the amount of computation significantly through the use of receding horizon control, it is still quite computationally intensive. Also, it only deals with single UAVs and doesn't consider target position estimation; the goal position (i.e., the endpoint) of the UAV's trajectory is deterministic and fixed. This is in fact a much simpler problem than that considered in this thesis. In the problem that is the subject of this thesis there are multiple UAVs and multiple targets whose position (and even existence) is not known initially and must be estimated "on the fly" as the trajectory control is performed. As a result the technique presented in [3] still requires too much computation to be useful for the problem of this thesis, and of course doesn't deal with the estimation aspects at all. It does, however, incorporate some useful ideas that have bearing on the current task.

2.1.4 Cooperative Sensor Network Problem Types

The articles summarized in Sections 2.1.2 and 2.1.3 offer various solutions to a representative range of CSN problems. The key CSN problems that are covered by the references are summarized below.

Sensor Beam Steering This problem involves multiple objects being observed by a sensor. An example of this type of problem involves a radar tracking multiple targets, where at each time step the controller must decide which target to observe, so as to minimize some cost function. Examples of research in this area from the work reviewed in Section 2.1.2 are [32], [14] and [58]. All three of these papers present solutions to this problem that use stochastic dynamic programming.

Sensor Waveform Selection An example of this type of problem is a radar with single or multiple targets. At each time step the controller decides which waveform mode to use with only one mode being able to be used at that time step. The choice of mode is based on the minimization of a cost function. The example considered in Section 2.1.2 on this topic is [34]. The solution approach used in [34] involves modelling the system as a POMDP and using stochastic dynamic programming to find the "optimal" waveform sequence.

Communication Bandwidth Limited Sensor Scheduling An example of this type of problem is a radar network with bandwidth limited communications between individual radars. At each time step the controller decides which data to transmit to the network's tracking computer on the basis of optimizing tracking performance within the communication bandwidth constraints. This type of problem may also include more general sensor usage costs in the optimization. Examples considered in Section 2.1.2 of research in this area are [14], [31] [50] and [21]. These papers describe solution approaches that use stochastic dynamic programming. Decisions on which sensor to activate over time are made on the basis of minimizing the sum of the estimation error and sensor usage costs.

Sensor Platform Steering The fourth type of problem involves typically multiple sensors whose combined state is controlled to optimize target state estimation. In this type of problem the positions of the sensors are controllable, while the targets are generally assumed to follow independent partially observed stochastic processes. The observation processes are also assumed to be independent and random. Stochastic control theory can form the basis of the solution to this problem as well. An important example of this type of problem structure

is that of UAVs carrying passive ES sensors, detecting EM radiation emanating from radar transmitters. Examples that are summarized in Section 2.1.3 of research on sensor platform steering are [45], [12], [56], [17], [38], [54] and [20]. Of these, [56] uses ad-hoc control laws to perform the UAV trajectory control, [12] uses straight forward gradient descent method, [45] uses an optimal control technique that does not involve dynamic programming, [17] uses a distributed receding horizon control technique that involves dynamic programming, [38] also uses a receding horizon control technique, [54] uses a gradient based technique and [20] uses a direct stochastic dynamic programming approach. While it does not involve sensing, the trajectory control technique described in [3] has relevance to the sensor platform steering problem we are interested in; it uses an optimal control technique that also does not explicitly involve dynamic programming, and has the added novelty of using a two level approach that achieves a near optimal solution with an optimization horizon at the final goal.

2.1.5 Proposed General Solution Approach

All four types of problem summarized in Section 2.1.4 are important in actual military applications, with typically more than one needing to be addressed simultaneously in a particular application. The upshot of this is that, all other things being equal, it would be advantageous to use the same general approach for all four of the problems. As can be seen from the references summarized in Sections 2.1.2 and 2.1.3, dynamic programming based approaches have been used in all four types of problem, so from this point of view alone it is desirable to seriously consider its use in the problem considered in this thesis. One important advantage of dynamic programming and one of the reasons for its use in all of these problems is its generality and as a result a wide range of optimal control and scheduling problems are amenable to its application. Using

a dynamic programming based approach for the CSN being developed in this thesis, if done properly, will impart the advantages of compatibility with other CSNs developed using the same approach as well as extendability thus enabling development of systems of increased complexity and a mix of optimization parameters. Another advantage of this approach is that it provides a method of taking into account long term cost, which is highly desirable in the above problems and this is particularly evident in the sensor platform steering problem. Dynamic programming is also well suited to all four types of problem for another important reason, i.e., uncertainty in the sensor information in these problems necessitates continual adaptation as more information is obtained, and dynamic programming is in essence an adaptive technique. There is one important disadvantage that dynamic programming has, and that is that exact solutions to real world problems which are computationally feasible are often not achievable using the technique. For this reason, gradient descent techniques which only perform local optimization are often used instead. However the solutions they produce are often far from optimal in the global sense and they are also susceptible to convergence to local minima. A middle ground is to develop approximate dynamic programming based solutions using ingenuity to achieve near optimality in a global sense and computational feasibility. The approach that will be developed in this thesis will use that approach. Before developing the solution, a considerable amount of further reading of the literature, now concentrating on more general theory in relevant areas of mathematics and engineering, was performed; this will be summarized in the following section.

2.2 Review of Relevant Theory

The second aim of the literature survey was to determine, and gain a knowledge of, relevant areas of mathematics as well as gain new ideas on how to approach the problem. As a result, the areas covered in this component of the literature survey were quite broad, the reasoning being that a wide range of background material should be looked at so as to enable the selection of appropriate techniques from as wide a range of options as possible. While some papers were studied, the primary references for this part of the survey were text books. The text books that have been fully or partially read in order to gain knowledge of the areas of mathematics and engineering that could have some relevance to the problem considered in this thesis are as follows. Philosophical aspects of Bayesian probability theory are covered in [25]. Differential geometry was considered to have general application to the problem; some basic theory in this field is covered in [39]. Dynamic programming will form the core of the approach that will be used in this project. For this the major reference was [5]. The broader area of optimal control is described in [33]. Planning, which is a closely related field, is covered in [36]. Estimation and tracking theory is another key relevant area for this work. For this, [2] has been referenced. The field of convex optimization has some relevance to this work. For this, [7] was studied. Information Theory is covered in [11] and [46]. The theory of differential games is covered in [26]. Bayesian networks are covered in [28]. Dynamical system stability theory is covered in [59]. Finally, a number of topics of relevance to advanced and intelligent control systems are covered in [1].

The relevant theory from the text books that had the most influence in developing the algorithms presented in this thesis will now be summarized in the following chapter. The chapter is presented as a background theory chapter,

and for the sake of conciseness is a summary of only the theory that is required in the subsequent chapters. Note that presentation of the theory of stability of dynamical systems is postponed until Chapter 6, where it is subsequently used for stability analysis of the solutions that have been developed in this project.

Chapter 3

Theoretical Background

3.1 Some Preliminaries

The solutions that will be presented in the following sections treat the states of the systems concerned as stochastic processes and contain an underlying assumption of the "Markov property"; the following definitions relate to this.

3.1.1 Stochastic Processes

A scalar random variable is a (real) number x determined by the outcome ω of a random experiment

$$x = x(\omega)$$

A (scalar) random process or *stochastic process* [2, p. 55] is a function of time determined by the outcome of a random experiment

$$x(t) = x(t, \omega)$$

This is a family or an ensemble of functions of time, in general different for each outcome of ω.

3.1.2 Markov Processes

A *Markov process* is a stochastic process [2, p. 59] that is defined by the following property (called the *Markov property*)

$$p\left[x\left(t\right)|x\left(\tau\right),\tau\leq t_1\right]=p\left[x\left(t\right)|x\left(t_1\right)\right]\qquad\forall t>t_1$$

In words, this property states that the past up to any time t_1 is fully characterized by the value of the process at t_1, or worded in another way, "the future is independent of the past if the present is known".

The state $\mathbf{x}\left(t\right)$ of a (possibly time-varying) dynamic system driven by white noise $\mathbf{n}\left(t\right)$ as defined by the following equation:

$$\dot{\mathbf{x}}\left(t\right)=f\left[t,\mathbf{x}\left(t\right),\mathbf{n}\left(t\right)\right]$$

is a Markov process, where, in general both $\mathbf{x}\left(t\right)$ and $\mathbf{n}\left(t\right)$ are vector-valued random processes.

3.1.3 Markov Sequences

A random sequence or a *discrete time stochastic process* is a time-indexed sequence of random variables

$$X^k=\left\{x\left(j\right)\right\}_{j=1}^{k}\qquad k=1,2,...$$

In a similar way to the continuous-time definition of the Markov property, a random sequence is a *Markov sequence* or *discrete time Markov process* [2, p. 62] if

$$p\left[x\left(k\right)|X^j\right]=p\left[x\left(k\right)|x\left(j\right)\right]\qquad\forall k>j$$

The (real-valued) zero-mean sequence $v\left(j\right),\ j=1,2,...$ is a *discrete time white noise* (or a white sequence) if

$$E\left[v\left(k\right)v\left(k\right)\right]=q\left(k\right)\delta_{kj}$$

where δ_{kj} is the Kronecker delta function and $q(k)$ is the variance of the sequence. If $q(k) = q$, i.e., the variance is time invariant, the sequence is *stationary.*

The state $\mathbf{x}(k)$ of a dynamic system excited by white noise $\mathbf{v}(k)$ as defined by the following equation

$$\mathbf{x}(k+1) = f[k, \mathbf{x}(k), \mathbf{v}(k)]$$

is a discrete time Markov process, or Markov sequence where, in general both $\mathbf{x}(k)$ and $\mathbf{v}(k)$ are vector-valued.

The state of a discrete time linear dynamic system excited by white Gaussian noise

$$\mathbf{x}(k+1) = F\mathbf{x}(k) + \mathbf{v}(k)$$

is a *Gauss-Markov process.*

3.1.4 A Remark on Notation for Discrete Time Processes

Much of the mathematics presented in the remainder of this thesis will involve representations of discrete time systems. Two types of notation will be used for the time indexing. The first involves simply placing the time index k in brackets as above, whilst the second has the time index as a subscript. So, for example, if we have a variable x that is a function of discrete time then

$$x(k) \triangleq x_k \triangleq x(t_k)$$

The choice of which is used will be based on which is most convenient for the particular situation.

3.1.5 Markov Decision Processes

A Markov decision process (MDP) is an extension on the concept of a Markov process (described in Sections 3.1.2 and 3.1.3). It is a natural way of formulating problems involving stochastic processes in which decisions are made incrementally as additional information is received. The basic formulation of an MDP [22, p. 1] consists of four objects (S, C, q, r) where

(a) S is the *state-space*. The elements x of S are called the *states*.

(b) C is the *action* (or *control*) space. To each state $x \in S$ we associate a non-empty subset $U(x)$ of C, whose elements u are *admissible actions* (or controls) when the system is in state x.

(c) q is the *transition law*.

(d) r is the *one-step reward function* (alternately one can use a *cost function*).

The above definition is interpreted as representing a controlled stochastic system which is observed at times t_k, $k = 0, 1, 2, \ldots$. Let the state and control at time t_k be denoted by x_k and u_k, respectively, then the MDP generates a sequence of states that evolves as follows. At time $k = 0$, the system starts at the initial state x_0, then for all $k \geq 0$, if the system is in state x_k at time t_k and the control u_k is chosen, a reward $r(x_k, u_k)$ is received and the system moves to a new state x_{k+1} according to the probability distribution $q(x_{k+1}|x_k, u_k)$.

A *partially observed Markov decision process* (POMDP) is a generalization of an MDP. In a POMDP the system dynamics are modelled as for an MDP; however, the underlying state cannot be directly observed. Instead, a probability distribution over all possible states x, known as the *belief state* is maintained. To account for this, the formulation of a POMDP consists of all the objects described above for an MDP, plus the following [22, p. 84], [21]

(e) An observation space Z. The elements z of Z are called the *measurements*.

(f) An observation map $L(z|x,u)$, which probabilistically maps states x and controls u into measurements z.

(g) An initial observation map $L_0(z|x)$, which probabilistically maps states x measurements z.

(h) An *a priori* initial probability distribution P_0 for the state x.

The sequence of states generated by a POMDP evolves as follows. At time $k = 0$, the system starts at the initial (unobservable) state x_0 which has a given *a priori* probability distribution $P_0(x_0)$, and the initial measurement z_0 is generated according to the initial observation map $L_0(z_0|x_0)$. Then for all $k \geq 0$, if the system is in state x_k at time t_k and the control u_k is applied, a reward $r(x_k, u_k)$ is received and the system moves to a new state x_{k+1} according to the probability distribution $q(x_{k+1}|x_k, u_k)$. The measurement z_{k+1} is generated according to the observation map $L(z_{k+1}|x_{k+1}, u_k)$.

The solution of problems with this structure (i.e., both MDPs and POMDPs) comes in the form of a policy, i.e., a rule that specifies which control one should apply if one arrives in a particular state (or belief state) at a particular time. To arrive at the policy, MDPs and POMDPs can be solved using dynamic programming; this will be described in Section 3.3.1.

3.2 Estimation and Tracking

Estimation and tracking are an important component of the problem being considered in this thesis. The generally accepted way of performing this function is to treat the system to be tracked as a dynamic stochastic process. Key

aspects of the approach, with emphasis on what will be used later in this thesis, are described in this section. Note that for the purposes of this thesis the targets that will be tracked will be treated as a linear system. This is only an approximation, as in reality the measurement process (using angle-only sensors), when considered in Cartesian coordinates, will be non-linear; more will be said about this later.

3.2.1 Continuous Time Linear Stochastic Dynamic Systems

The state space model of continuous time linear stochastic systems can be represented as [2, p. 183]

$$\dot{\mathbf{x}}(t) = A(t)\,\mathbf{x}(t) + B(t)\,\mathbf{u}(t) + D(t)\,\tilde{\mathbf{v}}(t) \tag{3.1}$$

where

$\mathbf{x}(t)$ is the n_x dimensional state vector of the system,

$\mathbf{u}(t)$ is the n_u dimensional (control) input vector,

$\tilde{\mathbf{v}}(t)$ is the continuous time zero-mean white n_v dimensional Gaussian process (plant) noise, with autocorrelation $E\left[\tilde{\mathbf{v}}(t)\,\tilde{\mathbf{v}}(\tau)^T\right] = V(t)\,\delta(t-\tau)$,

$A(t)$ is the system matrix,

$B(t)$ is the (continuous time) input gain,

$D(t)$ is the (continuous time) noise gain.

$A(t)$, $B(t)$, and $D(t)$ are known matrices of dimensions $n_x \times n_x$, $n_x \times n_u$, $n_x \times n_v$, respectively.

The output of the system can be represented as

$$\mathbf{z}(t) = C(t)\,\mathbf{x}(t) + \tilde{\mathbf{w}}(t) \tag{3.2}$$

where

$\tilde{\mathbf{w}}(t)$ is the output disturbance or measurement noise, and

$C(t)$ is the (known) $n_z \times n_x$ measurement matrix.

Equation 3.1 is known as the (continuous time) dynamic or *plant equation* and Equation 3.2 is referred to as the output or *measurement equation*. The noises $\tilde{\mathbf{v}}(t)$ and $\tilde{\mathbf{w}}(t)$ in Equations 3.1 and 3.2 are usually assumed to be zero mean, white and mutually independent.

3.2.2 Discrete Time Linear Stochastic Dynamic Systems

In most practical situations it is convenient to discretize time and use a discrete time state space model instead [2, p. 192]. In the state space representation of a discrete time system it is assumed that the input is piecewise constant, i.e.,

$$\mathbf{u}(t) = \mathbf{u}(t_k) \qquad t_k \leq t \leq t_{k+1}$$

The state at sampling time t_{k+1} can be written in the following form

$$\mathbf{x}(t_{k+1}) = F(t_{k+1}, t_k)\,\mathbf{x}(t_k) + G(t_{k+1}, t_k)\,\mathbf{u}(t_k) + \mathbf{v}(t_k)$$

For a time-invariant continuous time system sampled at arbitrary times we then have

$$F(t_{k+1}, t_k) = F(t_{k+1} - t_k) = e^{(t_{k+1}-t_k)A} \triangleq F(k)$$

$$G(t_{k+1}, t_k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)A} B d\tau \triangleq G(k)$$

$$\mathbf{v}(t_k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)A} D\tilde{\mathbf{v}}(\tau)\, d\tau \triangleq \mathbf{v}(k)$$

Assuming $\tilde{\mathbf{v}}(t)$ to be zero-mean and white results in

$$E[\mathbf{v}(k)] = 0$$

and

$$E\left[\mathbf{v}(k)\,\mathbf{v}(k)^T\right] = Q(k)\,\delta_{kj}$$

where δ_{kj} is the Kronecker delta function, and the covariance $Q(k)$ of the discrete time process noise is

$$Q(k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)A} DV(\tau) D^T e^{(t_{k+1}-\tau)A^T} d\tau$$

Hence the system equation for a *discrete time* linear dynamic stochastic system can be described using index-only notation as

$$\mathbf{x}(k+1) = F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \mathbf{v}(k) \qquad k = 0, 1, ... \tag{3.3}$$

Using similar notation the discrete time measurement equation can be written as

$$\mathbf{z}(k) = H(k)\mathbf{x}(k) + \mathbf{w}(k) \qquad k = 1, 2, ... \tag{3.4}$$

where

$$E[\mathbf{w}(k)] = 0$$

and

$$E\left[\mathbf{w}(k)\mathbf{w}(k)^T\right] = R(k)\delta_{kj}$$

Here the measurement given by Equation 3.4 represents a "short-term" integration during which the state is assumed to be constant.

The definitions of the entries in Equations 3.3, and 3.4 for the discrete time system are

$\mathbf{x}(k)$ is the n_x dimensional state vector,

$\mathbf{u}(k)$ is an n_u dimensional known (control) input vector,

$\mathbf{v}(k), \quad k = 0, 1, ...$ is a sequence of zero-mean white n_v dimensional Gaussian process noise,

$\mathbf{z}(k)$ is the n_z dimensional measurement vector,

$\mathbf{w}(k)$, $k = 1, 2, ...$ is a sequence of zero-mean white n_z dimensional Gaussian measurement noise,

$F(k)$ is the state transition matrix,

$G(k)$ is the gain through which the (control) input enters the system, and

$H(k)$ is the measurement matrix.

One can alternatively define a direct discrete time model rather than a discretized version of a continuous time model. In that case the process noise, which is modelled *directly* as a sequence of zero-mean white n_v dimensional Gaussian noise is typically modelled as entering the system through a *noise gain* $\Gamma(k)$, resulting in the following system equation

$$\mathbf{x}(k+1) = F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k) \qquad k = 0, 1, ... \tag{3.5}$$

In this case the process noise covariance is also defined directly as

$$Q(k) \triangleq E\left[\mathbf{v}(k)\mathbf{v}(k)^T\right]$$

resulting in

$$E\left[(\Gamma(k)\mathbf{v}(k))(\Gamma(k)\mathbf{v}(k))^T\right] = \Gamma(k)Q(k)\Gamma(k)^T$$

and similarly the measurement noise covariance matrix is defined directly as

$$R(k) \triangleq E\left[\mathbf{w}(k)\mathbf{w}(k)^T\right]$$

Note that the matrices $F(k), G(k), \Gamma(k), H(k), Q(k)$, and $R(k)$ are all assumed known and may be time-varying, i.e., the system can be time varying with non-stationary process and measurement noise. Also note that the state $\mathbf{x}(k)$, $k = 0, 1, ...$ is a Gauss-Markov process.

3.2.3 The Kalman Filter

The Kalman filter is the primary method of solving the problem of state estimation for the discrete time linear dynamic system described in the previous subsection. Before summarizing the Kalman filter algorithm some definitions will first be given, as follows. Let

$$\hat{\mathbf{x}}(j|k) \triangleq E\left[\mathbf{x}(j)\,|\mathbf{Z}^k\right]$$

where

$$\mathbf{Z}^k \triangleq \{\mathbf{z}(j),\ j \leq k\}$$

denotes the sequence of observations available at time k, and

$$E\left[\mathbf{x}(j)\,|\mathbf{Z}^k\right]$$

is the conditional expectation of $\mathbf{x}(j)$ given $\mathbf{Z}^k$.

If $j = k$ then $\hat{\mathbf{x}}(j|k)$ is the *estimate* of the state (also called the *filtered* value), or, if $j = k+1$ then $\hat{\mathbf{x}}(j|k)$ is the *predicted* value (one-step) of the state. The state estimation error at time k is defined as

$$\tilde{\mathbf{x}}(k|k) \triangleq \mathbf{x}(k) - \hat{\mathbf{x}}(k|k)$$

The state prediction error at time k is defined as

$$\tilde{\mathbf{x}}(k+1|k) \triangleq \mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1|k)$$

The state estimation covariance matrix (i.e., the covariance associated with the estimate $\hat{\mathbf{x}}(k|k)$) at time k is

$$P(k|k) \triangleq E\left[\tilde{\mathbf{x}}(k|k)\,\tilde{\mathbf{x}}(k|k)^T\,|\mathbf{Z}^k\right]$$

The state prediction covariance matrix (i.e., the covariance associated with the prediction $\hat{\mathbf{x}}(k+1|k)$) at time k is

$$P(k+1|k) \triangleq E\left[\tilde{\mathbf{x}}(k+1|k)\,\tilde{\mathbf{x}}(k+1|k)^T\,|\mathbf{Z}^k\right]$$

The predicted measurement (one-step) is

$$\hat{\mathbf{z}}(k+1|k) \triangleq E\left[\mathbf{z}(k+1)|\mathbf{Z}^k\right]$$

The measurement prediction error (also called the innovation or residual) is defined as

$$\boldsymbol{\nu}(k+1) \triangleq \tilde{\mathbf{z}}(k+1|k) \triangleq \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k)$$

The measurement prediction covariance matrix or innovation covariance matrix is

$$S(k+1) \triangleq E\left[\tilde{\mathbf{z}}(k+1|k)\,\tilde{\mathbf{z}}(k+1|k)^T|\mathbf{Z}^k\right]$$

The Kalman filter gain is

$$W(k+1) \triangleq P(k+1|k)\,H(k+1)^T\,S(k+1)^{-1}$$

The Kalman filter is a recursive algorithm, starting with the initial estimate $\hat{\mathbf{x}}(0|0)$ of $\mathbf{x}(0)$ and the associated initial covariance $P(0|0)$, both assumed to be available. The steps involved in one cycle of the algorithm are shown in Figure 3.1

3.2.4 Multitarget Tracking

In the type of problem being considered here, the number of targets is generally unknown. There are also often false detections occurring and some detections are missed. To deal with this, algorithms for track initiation, data association (i.e., assignment of detections to appropriate targets), track maintenance (when there are missed detections), and track termination (when targets go out of view), are generally required. Depending on the circumstances and details of the sensors and emitters, the multitarget tracking and false detection aspects of the problem can vary from relatively simple to extremely challenging. There

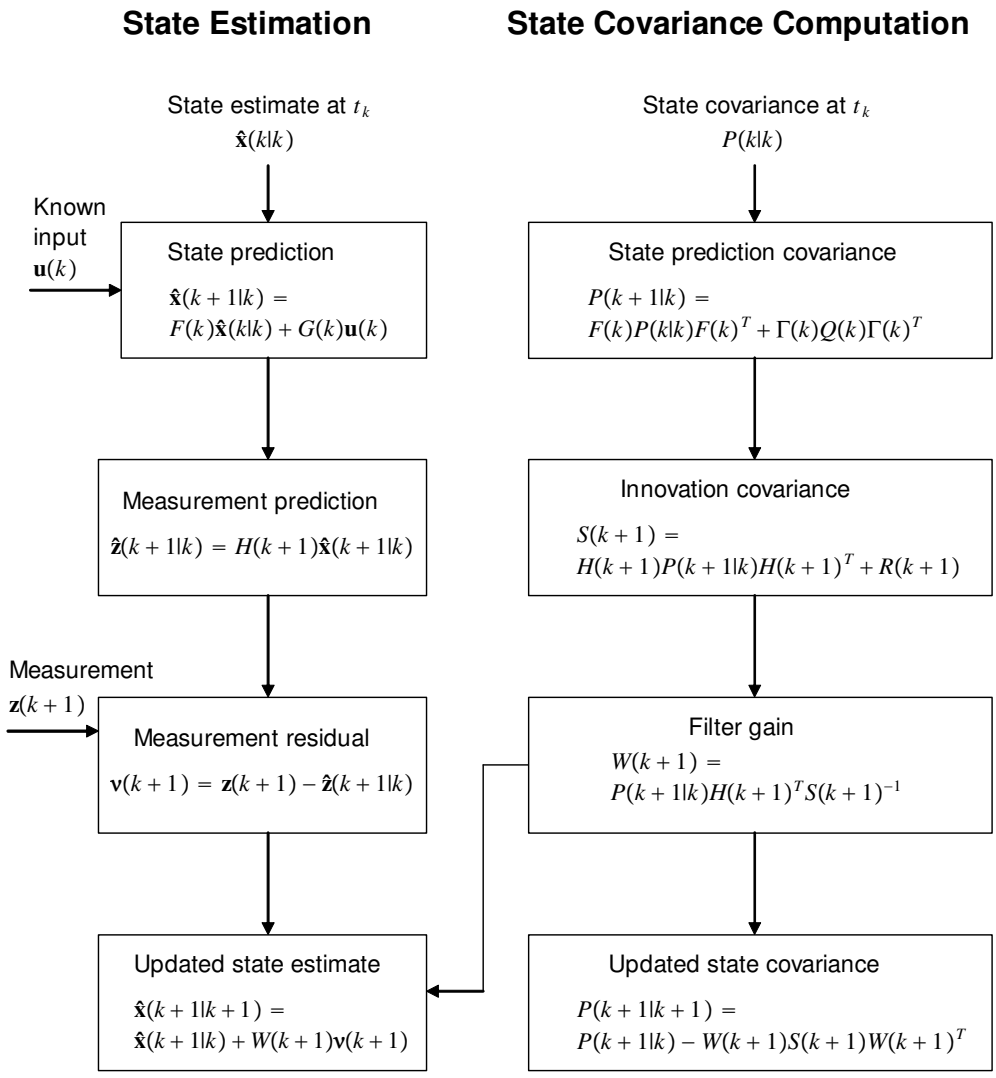

Figure 3.1: Cycle $k+1$ of the Kalman Filter

is considerable literature on the multitarget tracking problem covering a wide range of target and sensor types. One example of research for a situation very much like that dealt with in this thesis is [19]. As the central theme of this thesis is the *control* of UAVs, the details of how the multiple target tracking component would work are not considered here. Instead it is assumed that data association has been achieved successfully by unspecified algorithms and then (as part of the algorithms developed here) the correctly assigned measurements are applied as inputs to Kalman filters which perform the target state estimation process.

3.3 Optimal Control

The type of problem that we are considering has a sequential structure in which decisions must be made as information is being received and outcomes of the decisions are partly random and partly under the control of the decision maker. Dynamic programming provides a natural mathematical framework for modelling and solving this type of decision problem. It is used in a wide range of areas, including robotics, automated control as well as economics and manufacturing. We will now describe the dynamic programming algorithm, and following from that, show how other well known optimal control algorithms derive from it, finally finishing with the approach that we have developed for our problem.

3.3.1 Dynamic Programming

Consider the discrete-time dynamic system [5, p. 13]

$$x_{k+1} = f_k\left(x_k, u_k, v_k\right), \qquad k = 0, 1, ..., N-1$$

where the state x_k is an element of a space S_k, the control u_k is an element of a space C_k, and the random disturbance v_k is an element of a space D_k.

The control is constrained to take values in a predefined non-empty subset $U(x_k) \subset C_k$, which depends on the current state x_k, or in other words, $u_k \in U(x_k)$ for all $x_k \in S_k$ and k. The random disturbance v_k is characterized by a probability distribution $P_{v_k}(\cdot|x_k, u_k)$ that may depend explicitly on x_k and u_k but not on values of prior disturbances $v_{k-1}, ..., v_0$ (i.e., it satisfies the Markov property). Note that this system is an example of an MDP (summarized in Section 3.1.5).

Let us consider the class of policies (control laws) that consist of a sequence of functions

$$\pi = \left\{\mu_0, ..., \mu_{N-1}\right\}$$

where μ_k maps states x_k into controls $u_k = \mu(x_k)$ and $\mu(x_k) \in U_k(x_k)\ \forall x_k \in S_k$. Such policies will be called *admissible*. Given an initial state x_0 and an admissible policy $\pi = \left\{\mu_0, ..., \mu_{N-1}\right\}$, the states x_k and disturbances v_k have distributions defined by the system equation

$$x_{k+1} = f_k(x_k, \mu(x_k), v_k), \qquad k = 0, 1, ..., N-1$$

For given cost functions g_k, $k = 0, 1, ..., N$, the expected cost of the policy π starting at x_0 is

$$J_\pi(x_0) = E\left\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu(x_k), v_k)\right\}$$

where the expectation is taken over the random variables x_k and v_k. The optimal policy π^* is the one that minimizes the cost, i.e.,

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)$$

where Π is the set of all admissible policies.

The optimal cost depends on x_0 and will be denoted $J^*(x_0)$, i.e.,

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)$$

$J^*(x_0)$ will be called the *optimal cost function* or the *optimal value function.*

The basic idea that the dynamic programming algorithm is based on is the *principle of optimality* [5, p. 18] which can be stated as follows.

Let $\pi^* = \left\{\mu_0^*, ..., \mu_{N-1}^*\right\}$ be an optimal policy for the basic problem, and assume that when using π^*, a given state x_i occurs at time i with positive probability. Consider the subproblem whereby we are at x_i at time i and wish to minimize the "cost-to-go" from time i to time N, i.e.,

$$E\left\{g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu(x_k), v_k)\right\}$$

then the truncated policy $\left\{\mu_i^*, \mu_{i+1}^*..., \mu_{N-1}^*\right\}$ is optimal for this subproblem.

Using this principle leads us to the dynamic programming algorithm, which can be stated as follows [5, p. 23].

For every initial state x_0, the optimal cost $J^*(x_0)$ of the basic problem is equal to $J(x_0)$, given by the last step of the following algorithm, which proceeds backward in time from period $N-1$ to 0

$$J_N(x_N) = g_N(x_N),$$

$$\begin{aligned} J_k(x_k) &= \min_{u_k \in U_k(x_k)} \underset{v_k}{E}\left\{g_k(x_k, u_k, v_k) + J_{k+1}(f_k(x_k, u_k, v_k))\right\}, \qquad (3.6) \\ k &= 0, 1, ...N-1 \end{aligned}$$

where the expectation is taken with respect to the probability distribution of v_k which depends on x_k and u_k. Furthermore, if $u_k^* = \mu_k^*(x_k)$ minimizes the right side of Equation 3.6 for each x_k and k, the policy $\pi^* = \left\{\mu_0^*, ..., \mu_{N-1}^*\right\}$ is optimal.

The above algorithm assumes that the controller has access to the exact value of the current state (this is referred to as the *perfect state information* case); however, in the problem that is considered in this thesis, as with many other real-life problems, this is not the case. Some of the variables are not directly accessible; instead, their values are measured with relatively inaccurate sensors. We can model this type of problem by assuming that at each stage the controller receives some observations of the value of the current state, which are corrupted by stochastic uncertainty. Problems in which the controller uses observations of this type in place of the actual state are called problems of *imperfect state information*; they are also referred to as POMDPs (summarized in Section 3.1.5). Conceptually these types of problem can be reduced to the problem of perfect state information and hence solved using dynamic programming. One way of reducing the problem to the perfect information case is described below.

Problems with Imperfect State Information

Consider the problem where, instead of knowing x_k, we receive observations

$$z_0 = h_0(x_0, w_0), \; z_k = h_k(x_k, u_{k-1}, w_k), \; k = 1, 2, ..., N-1$$

where each observation z_k belongs to a given observation space Z_k, the random observation disturbance w_k belongs to a given space W_k. We will assume that the probability distribution of the observation disturbance w_{k+1} depends explicitly only on the immediately preceding state, control, and system disturbance x_k, u_k, v_k and not on $x_{k-1}, ..., x_0, u_{k-1}, ..., u_0, v_{k-1}, ..., v_0, w_{k-1}, ...w_0$.

The initial state x_0 is also random and characterized by a probability distribution P_{x_0}. The probability distribution $P_{v_k}(\cdot|x_k, u_k)$ of v_k is given and may depend explicitly on x_k and u_k but not on values of prior disturbances $v_{k-1}, ..., v_0$, $w_{k-1}, ..., w_0$. The control u_k is constrained to take values from a given nonempty

subset U_k of the control space C_k and it is assumed that this subset does not depend on x_k.

Let I_k be the information available to the controller at time k; we will call it the *information vector*. We have

$$\begin{aligned} I_k &= (z_0, ..., z_k, u_0, ..., u_{k-1}), \ k = 1, 2, ..., N-1, \\ I_0 &= z_0 \end{aligned}$$

We now consider the class of policies that consist of a sequence of functions

$$\pi = \left\{\mu_0, ..., \mu_{N-1}\right\}$$

where each function μ_k maps the information vector I_k into the control space C_k and $\mu(I_k) \in U_k$ for all I_k, $k = 0, 1, ..., N-1$. These policies will be called *admissible*. We now want to find an admissible policy $\pi = \left\{\mu_0, ..., \mu_{N-1}\right\}$, that minimizes the cost function

$$J_\pi = \underset{\substack{x_0, v_k, w_k \\ k=0,..,N-1}}{E} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu(I_k), v_k) \right\}$$

subject to the system equation

$$x_{k+1} = f_k(x_k, \mu(I_k), v_k), \qquad k = 0, 1, ..., N-1$$

and the measurement equation

$$z_0 = h_0(x_0, w_0), \ z_k = h_k\left(x_k, \mu_{k-1}(I_{k-1}), w_k\right), \ k = 1, 2, ..., N-1$$

Note that for the perfect information case we were looking for a rule that specifies the control u_k to be applied for each state x_k and time k, whereas now we are trying to find a rule that gives the control u_k to be applied for every possible information vector I_k and time k.

Now, with the assumptions made above, it can be shown [5, chapt. 5] that a *sufficient statistic* for the above problem is given by the conditional probability distribution $P_{x_k|I_k}$ of the state x_k, given the information vector I_k. In particular, we can write for all k

$$P_{x_{k+1}|I_{k+1}} = \Phi\left(P_{x_k|I_k}, u_k, z_{k+1}\right),$$

where Φ_k is some function that can be determined from the data associated with the problem, u_k is the control, and z_{k+1} plays the role of a random disturbance whose statistics are known and depend explicitly on $P_{x_k|I_k}$ and u_k, and not on $z_k, ..., z_0$.

The dynamic programming algorithm for the imperfect information case, can be thus stated as follows [5, p. 246] for all $k < N-1$

$$\bar{J}_k\left(P_{x_k|I_k}\right) = \min_{u_k \in U_k}\left[\underset{x_k, v_k, z_{k+1}}{E}\left\{g_k\left(x_k, u_k, v_k\right) + \bar{J}_{k+1}\left(\Phi\left(P_{x_k|I_k}, u_k, z_{k+1}\right)\right)|I_k, u_k\right\}\right]$$

and for the case where $k = N-1$

$$\bar{J}_{N-1}\left(P_{x_{N-1}|I_{N-1}}\right) = \min_{u_{N-1} \in U_{N-1}}\left[\underset{x_{N-1}, v_{N-1}}{E}\left\{\begin{array}{c} g_N\left(f_{N-1}\left(x_{N-1}, u_{N-1}, v_{N-1}\right)\right) + \\ g_{N-1}\left(x_{N-1}, u_{N-1}, v_{N-1}\right)|I_{N-1}, u_{N-1}\end{array}\right\}\right]$$

The above algorithm yields a policy of the form

$$\bar{\pi}^* = \left\{\bar{\mu}_0^*\left(P_{x_0|I_0}\right), ., \bar{\mu}_k^*\left(P_{x_k|I_k}\right), .., \bar{\mu}_{N-1}^*\left(P_{x_{N-1}|I_{N-1}}\right)\right\}$$

and the optimal cost is given by

$$J^* = \underset{z_0}{E}\left\{\bar{J}_0\left(P_{x_0|z_0}\right)\right\}$$

where J_0 is obtained from the last step of the of the algorithm, and the probability distribution of z_0 is obtained from the measurement equation $z_0 = h_0\left(x_0, w_0\right)$ and the statistics of x_0 and w_0.

The above algorithm can be viewed as a dynamic programming algorithm of a perfect state information problem whose state is $P_{x_k|I_k}$. Hence, in the absence of perfect knowledge of the state, the controller can be viewed as controlling the *probabilistic state* $P_{x_k|I_k}$ so as to minimize the expected cost-to-go conditioned on the information I_k. The representation of the optimal policy as a sequence of functions of the conditional probability $P_{x_k|I_k}$ is conceptually very useful in that it provides a decomposition of the optimal controller into two parts, i.e.,

- an *estimator*, which generates the probability distribution $P_{x_k|I_k}$ using the measurement z_k and the control u_{k-1},
- an *actuator*, which generates a control input to the system as a function of the probability distribution $P_{x_k|I_k}$.

While conceptually useful, the above algorithm, as it stands can only be applied in practice for relatively simple problems that have a solution space that is relatively small. The problem is that for most real-life applications the amount of computation required to obtain optimal policies is so huge that the algorithm simply is infeasible to apply. However, one can take this concept further. Consider the case of a linear system and a quadratic cost function; this type of system can be described by the following equations [5, chapt. 5.2]

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + v_k \qquad k = 0, 1, ..., N-1 \\ z_k &= C_k x_k + w_k \qquad k = 0, 1, ..., N-1 \end{aligned}$$

and its cost as

$$E\left\{ x_N^T Q_N x_N + \sum_{k=0}^{N-1} \left(x_k^T Q_k x_k + u_k^T R_k u_k \right) \right\}$$

where x_k, z_k and u_k are vectors of dimension n_x, n_z and n_u, respectively and the matrices A_k, B_k, C_k, Q_k and R_k are given and have the appropriate dimensions.

The matrices Q_k and R_k are assumed to be positive semidefinite symmetric and positive definite symmetric, respectively. The controls u_k are unconstrained, and the disturbances v_k are independent random vectors with given probability distributions that have zero mean and finite second moment and are also independent of x_k and u_k. The observation noise vectors w_k are independent and also independent of v_k and x_0.

It can be shown that for the above problem the optimal policy is the same as for the perfect information case, but with the state x_k replaced by its conditional expectation $E\{x_k|I_k\}$. So now we have the optimal controller decomposed into two parts as follows

- an *estimator*, which generates the *conditional expectation* $E\{x_k|I_k\}$ using the measurement z_k and the control u_{k-1},
- an *actuator*, which multiplies $E\{x_k|I_k\}$ by a gain matrix L_k and applies the control input $u_k = L_k$.

The gain matrix L_k is independent of the statistics of the problem and is the same as would be the case with the deterministic problem with v_k and x_0 fixed and equal to their expected values. Hence, *the estimator portion of the optimal controller is an optimal solution of the problem of estimating the state x_k assuming the control is not subject to choice, and the actuator portion is an optimal solution of the control problem assuming perfect state information.* This property is referred to as the *separation theorem for linear systems and quadratic cost* or the *certainty equivalence principle.*

If we go one step further and assume the disturbances v_k, w_k and the initial state x_0 are Gaussian random vectors, a computationally efficient implementation of the estimator of the conditional expectation $E\{x_k|I_k\}$ exists, i.e., the Kalman filter, which was described in Section 3.2.3.

The outcome of all this is a huge reduction in the amount of computation required while still retaining optimality of the solution. The above result for linear systems with quadratic cost is so useful that it is also used as the basis of various suboptimal control schemes for *nonlinear* systems. In fact, because of the challenging nature of optimal stochastic control of nonlinear systems, approximations are often the only recourse. One of the more straightforward methods of approximation is based on the idea that any sufficiently smooth nonlinear system is approximately linear in the neighbourhood of a reference sequence of states. The control scheme is then designed using the separation properties of linear quadratic systems, bearing in mind that the resulting control will be suboptimal; the resulting controller is referred to as a *certainty equivalent* (CE) controller [5, p. 271]. This approximation will, in fact, be used in the development of the algorithm that is the subject of this thesis. At this point, the reader is also referred to one of the seminal papers on separation theorems for discrete time stochastic control systems by H. S. Witsenhausen, i.e., [60]. The paper discusses the above separation theorem as well as other "softer" separation theorems based on less restrictive assumptions.

3.3.2 Certainty Equivalent Control

As mentioned above, the CE controller is a suboptimal control scheme that applies at each stage the control that would be optimal if some or all of the uncertain quantities were fixed at some "typical" values. To see how this works, consider the imperfect state information problem described above. Now assume we have an estimator that uses the information I_k to produce an estimate $\bar{x}_k(I_k) = E\{x_k|I_k\}$ of the state, and for every state control pair (x_k, u_k) an estimate of the disturbance $\bar{v}_k(x_k, u_k) = E\{v_k|x_k, u_k\}$. The control input $\tilde{\mu}_k(I_k)$ applied by the CE controller at time k is then determined by the following

algorithm:

1. Compute a state estimate $\bar{x}_k(I_k)$ given the current information vector I_k.

2. Find a suboptimal control sequence $\{\tilde{u}_k, \tilde{u}_{k+1}, ..., \tilde{u}_{N-1}\}$ that solves the deterministic problem of minimizing

$$g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \bar{v}_i(x_i, u_i))$$

subject to the initial condition $x_k = \bar{x}_k(I_k)$ and for $i = k, k+1, ..., N-1$

$$u_i \in U_i(x_i), \; x_{i+1} = f_i(x_i, u_i, \bar{v}_i(x_i, u_i))$$

3. Use the first element in the resulting control sequence as the control, i.e., $\tilde{\mu}_k(I_k) = \tilde{u}_k$.

The above optimization algorithm must be solved for each time k, hence a total of N such problems has to be solved by the CE controller to determine the entire sequence of control inputs $\tilde{\mu}_k(I_k)$, $k = 0, 1, ..., N-1$. Note, however, that each of the problems is deterministic.

Even though the *certainty equivalent control* (CEC) approach greatly reduces the amount of computation required, it still requires the solution of a deterministic optimal control problem at each stage. For many problems this still requires more computation than is feasible to implement. A method to reduce the computation further is to use some form of heuristic algorithm. One commonly used variant of the CEC approach is to use an easily implementable heuristic to find a suboptimal control sequence $\{\tilde{u}_k, \tilde{u}_{k+1}, ..., \tilde{u}_{N-1}\}$ for each stage of the above minimization problem. The way this is generally done is to use minimization over the first control u_k with the cost of the remaining stages

$k+1, ..., N-1$ being approximated using the heuristic. Hence, at each time k, a control $\tilde{u}_k$ that minimizes the expression

$$g_k\left(x_k, u_k, \bar{v}_k\left(x_k, u_k\right)\right) + H_{k+1}\left(f_k\left(x_k, u_k, \bar{v}_k\left(x_k, u_k\right)\right)\right),$$

over $u_k \in U_k\left(x_k\right)$ is applied, where H_{k+1} is the *cost-to-go* function corresponding to the heuristic. Note that $H_{k+1}\left(f_k\left(x_k, u_k, \bar{v}_k\left(x_k, u_k\right)\right)\right) = H_{k+1}\left(x_{k+1}\right)$ is the estimate of the cost incurred over the remaining stages $k+1, ..., N-1$, starting from state x_{k+1}, based on the heuristic and the assumption that the future disturbances are equal to their typical values $\bar{v}_i\left(x_i, u_i\right)$. In a later chapter of this dissertation, a particular and commonly used method of approximating the cost-to-go called *policy rollout* will be described in some detail and used to solve part of the optimization problem that we are concerned with.

Using the above heuristic based approach increases the range of problems that can be dealt with; however, many practical problems are sufficiently complex that, by itself, the approach does not simplify the computations sufficiently to provide the required solution. This is, in fact, the case for the problem being dealt with in this thesis. Two approaches that can further aid us in dealing with complexity are hierarchical control and model predictive control. These will be summarized in the following two sections.

3.3.3 Hierarchical Control

The advantage of developing hierarchical solutions for the optimization of dynamical systems is that, although dynamic programming can be used in principle, as the dimension and number of variables is increased the direct solutions to the control problems rapidly become intractable; this dilemma was first referred to by Bellman [4] as the "*curse of dimensionality*". Breaking up the problems

into a hierarchy of smaller control problems can make otherwise infeasible optimization problems solvable.

For our purposes, a hierarchy can be defined as having the following properties [52, chapt. 1]:

1. A hierarchy consists of decision making units arranged in a pyramid structure, where at each level, one or more units operate in parallel.

2. Hierarchical structures exist in systems which have an overall goal, and the goals of all the decision making units are in harmony.

3. There is an iterative information exchange between the decision making units on the various levels of the hierarchy, with a precedence for information going down and being treated as commands by the lower levels which try to obey where possible.

4. The time horizon of interest increases as one goes up the hierarchy.

A well known method of breaking up an optimization problem into a hierarchy is to form the *Lagrangian dual* of the original optimization problem and solve the dual problem by decomposing it into N optimization problems, one for each subsystem, and then using a two level iterative calculation structure to find the optimal solution. The reader is referred to [35], [52] and [53] for a detailed description of this hierarchical solution approach. This approach, however, is not feasible for the problem that we are dealing with because, even if all other potential issues could be dealt with, the method does not sufficiently reduce the computation requirements for this problem to enable it to be implemented in a realizable real-time system, which is a primary requirement of any solution that is to be developed. Instead a hierarchical form of the *model predictive control* approach was used. Model predictive control is described in the next section.

3.3.4 Model Predictive Control

Model predictive control (MPC), otherwise known as *receding horizon control* or *moving horizon control* is a form of CEC that originated in the nineteen seventies and has found widespread use particularly in petro-chemical process control and related industries. With the increase in computational power of modern computers it is also now finding more widespread use in areas such as robotics. The term model predictive control defines a range of control methods that have a common general strategy [8]. The features that are common to the MPC strategy are

- explicit use of a model to predict the system output at future time instants up to a finite horizon and calculation of an optimal control sequence by minimizing a cost function,
- a receding horizon strategy where at each update of the controller the horizon is displaced toward the future.

In its simplest form the MPC algorithm is as follows. At each sampling time k, $k = 0, 1, 2, ...$ of the system

1. solve on-line, a finite horizon open-loop optimal control problem using the state of the system at time k as the initial state, and the information available at time k. This optimization results in an optimal control sequence $\left\{u_k^*, u_{k+1}^*, ..., u_{k+N_H-1}^*\right\}$ where N_H is the prediction horizon,
2. then apply the *first* control in this sequence, i.e., u_k^* to the plant.

MPC offers a range of advantages over other methods, the most important being

- It is one of the few methods that can deal effectively with hard constraints.
- It can be used to control a large variety of processes from those with quite simple dynamics to more complex ones.
- It has the ability to handle control problems where off-line computation of a control law is difficult or impossible.
- It is able to effectively deal with multi-variable systems.
- It is an open methodology based on certain basic principles which allows for future extensions.

There is a considerable literature on MPC which attests to the importance and usefulness of this approach. The reader is referred to the surveys of the state of the art and literature which can be found in [42] and [43].

The strategy described above can also be generalized somewhat as follows. Let N_P be the *planning horizon* where $2 \leq N_P < \infty$ and N_C be the *control horizon* where $1 \leq N_C \leq N_P$. Then the strategy is as follows:

1. at times $k = 0, N_C, 2N_C, 3N_C, ...$calculate and store the optimal control sequence $\left\{u_k^*, u_{k+1}^*, ..., u_{k+N_P-1}^*\right\}$ based on the information available at time k,
2. at each sampling time k, $k = 0, 1, 2, ...$ of the system, apply the optimal control action u_k^* for time k that was (most recently) calculated in step 1.

This version of MPC reduces the number of times the open loop optimizations are performed while still applying the most recently computed optimal control at each sampling time k of the plant. The main advantage of this generalization is a reduction in the amount of computation required, but with

the cost, of course, of some degradation in performance as N_C increases; with this approach the designer needs to balance performance against computation requirements for the particular application being considered.

3.3.5 Hierarchical Model Predictive Control

One can generalize the MPC concept still further; in particular there has been considerable research in distributed MPC and increasingly in recent years in *hierarchical* MPC with the aim of dealing with more complex and large scale systems. A variety of solutions to various types of problems have been presented in the literature; the reader is referred to the survey paper [47] by Scattolini for a survey and classification of distributed and hierarchical MPC solution strategies for various types of problems.

In this thesis we take the general idea of hierarchical MPC and develop a form that enables us to solve the distributed sensing problem that we are dealing with. The details will be described in the following chapter; however, the general concept involves the implementation of a planner which computes coarse-grained preliminary trajectories for the UAVs with a *planning horizon* N_P that is right up to the current estimate of the goal state. Instead of these trajectories being passed down to the individual UAV controllers, individual cost functions for each UAV, which are based on the trajectories computed by the planner, are notionally passed down and added together with local cost functions to compute and implement the individual UAVs control actions. The local computation and application of the control actions is performed at each system sampling time k with a *local optimization horizon* of 1. The *control horizon* N_C (as defined above) corresponds in length to a single leg of the most recently computed preliminary trajectory produced by the planner. Figure 3.2 shows the concept diagrammatically; in the figure the cost functions produced

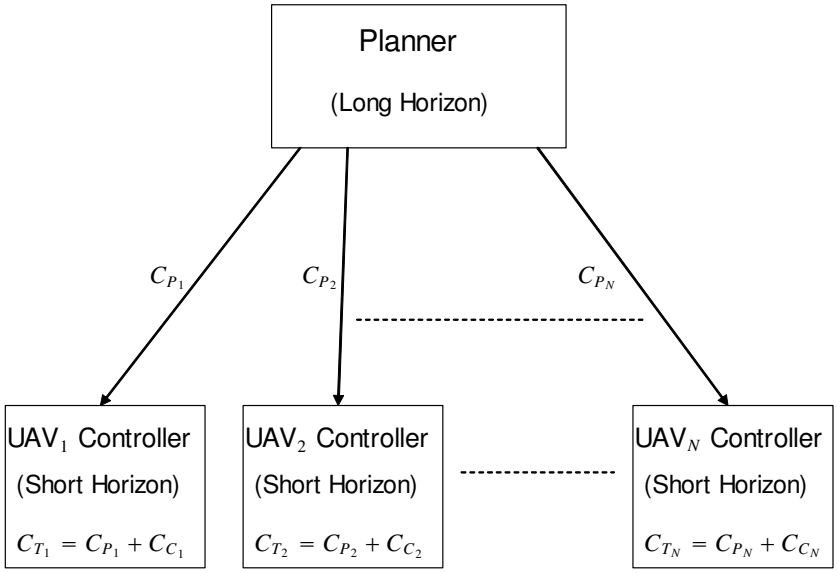

Figure 3.2: Proposed Hierarchical MPC Concept

by the planner are labelled C_{P_i}, the local cost functions are labelled C_{C_i} and the combined cost functions for each UAV are labelled C_{T_i} where $i = 1, .., N$ and N is the number of UAVs. Note that this control structure has the properties that were used to define a hierarchy in Section 3.3.3.

At this point now we have summarized the background theory that will be used for the formulation of the solution to the adaptive multisensor multitarget tracking problem being considered. The following chapter will now develop the hierarchical controller to address this problem.

Chapter 4

Hierarchical Model Predictive Control of UAV Trajectories

4.1 Problem Statement

The problem to be considered in this project is that of multiple UAVs carrying passive ES sensors, detecting, tracking and reacting to multiple stationary or slowly moving radar-carrying platforms. The tracking will be performed cooperatively with a centralized tracker collecting detections from the UAVs, performing data association and forming tracks. Upon detection of enemy radars, a centralized controller will assign UAVs to appropriate targets, based on target distance taking into account no-fly zones that must be avoided. The UAVs will respond to the threats by altering their trajectories to improve target position estimates in as short a time as possible, so as to enable reaction to threats with other resources which could, for example, include jammers or missile-carrying aircraft. Stochastic optimal control theory is used to form the basis of the solution to this problem.

Let us consider key components of the problem in some detail now.

Detection of targets Target detection will be by ES sensors which will provide angle-only information about radar emitters to a specified accuracy, which is generally quite coarse. The emitters will be assumed to be scanning radars with high gain narrow main beams. Detections by the ES sensors will only occur when they are illuminated by the main beam of a radar. The detections will hence be asynchronous.

Tracking of detected targets In this type of problem a multi-target Kalman filter based tracker, incorporating a data association algorithm, is generally used. For this paper a simplified tracker will be used which will include key features relevant to the scheduling problem being considered here.

Obstacle avoidance There may be no-fly zones in the area of coverage or areas to be avoided because of threats of UAVs being detected by the enemy. These obstacles need to be taken into account when UAV to target assignment is done and in the computation of UAV trajectories. For this to be done, planning algorithms need to be applied which optimize on the basis of long term goals while simultaneously avoiding the obstacles.

Optimal trajectory control Taking into consideration all of the above, the trajectories of the UAVs must be controlled in such a way as to optimize performance, the goal being to achieve required target position estimation accuracy within the shortest possible time so as to enable some operational goal to be achieved, e.g., destruction of enemy asset.

In the following sections the UAV trajectory optimization algorithm addressing the above problem is developed. The problem is approached by formulating it as a POMDP and developing a hierarchical model predictive control (Section 3.3.5) solution taking into account short and long term costs. The effectiveness

of the approach is evaluated by performing simulations involving multiple UAVs and targets.

4.2 Mathematical Framework and Proposed Control Algorithm Concept

As mentioned in Section 4.1, the system is modelled as a POMDP (Section 3.1.5); the optimization problem can then be formulated as a discrete time dynamic system of the following general form

$$x_{k+1} = f\left(x_k, u_k, v_k\right), \quad k = 0, 1, ..., N-1, \tag{4.1}$$

where k indexes discrete time, x_k is the state of the system at time k, u_k is the control or decision variable to be selected at time k, v_k is a random parameter (also called disturbance or noise), N is the horizon or number of times control is applied.

The controller has access to observations z_k of the form

$$z_0 = h_0\left(x_0, w_0\right),\ z_k = h_k\left(x_k, u_{k-1}, w_k\right),\ k = 1, 2, ..., N-1 \tag{4.2}$$

where w_k is the random observation disturbance.

The initial state x_0 is random and characterized by a given probability distribution P_{x_0}. The probability distribution $P_{v_k}\left(\cdot|x_k, u_k\right)$ of v_k is given, and it may depend explicitly on x_k and u_k but not on the prior disturbances $v_0, ..., v_{k-1}$, $w_0, ..., w_{k-1}$. The control u_k is constrained to take values from a given non-empty subset U_k of the control space C_k. Let us denote by I_k the information available to the controller at time k and call it the *information vector*. We have $I_0 = z_0$, $I_k = \left(I_{k-1}, z_k, u_{k-1}\right)$, $k = 1, 2, ..., N-1$. Furthermore, we have

$P(z_{k+1}|I_k, u_k) = P(z_{k+1}|I_k, u_k, z_0, z_1, ..., z_k)$ since $z_0, z_1, ..., z_k$ are part of the information vector I_k.

Consider the class of policies consisting of the sequence of functions $\pi = \{\mu_0, ..., \mu_{N-1}\}$, where each function μ_k maps the information vector I_k into the control space C_k and $u_k = \mu_k(I_k) \in U_k$, for all I_k, $k = 0, 1, ..., N-1$. Such policies are called *admissible*. Let $g_k(x_k, u_k, v_k)$ be the cost function associated with selection of control u_k at time k, $k = 0, 1, ..., N-1$. The optimal solution is then the admissible policy $\pi = \{\mu_0, ..., \mu_{N-1}\}$ that minimises the cost

$$J_\pi = \underset{\substack{x_0, v_k, w_k \\ k=0,..,N-1}}{E} \left\{ \sum_{k=0}^{N-1} g_k(x_k, u_k, v_k|I_k, u_k) \right\} \tag{4.3}$$

subject to Equations 4.1 and 4.2.

In principle the optimal solution can be found using dynamic programming, however, in practice computing the solution is infeasible because of the excessive amount of computation required. Instead, an approximate dynamic programming approach is considered here involving a form of CEC by substituting x_k with its state estimate $\bar{x}_k(I_k) = E\{x_k|I_k\}$, and similarly letting $\bar{v}_k(x_k, u_k) = E\{v_k|x_k, u_k\}$ and also applying heuristics to further reduce the amount of computation required. With these substitutions an approximate solution can be formulated in which a minimisation is performed on the following expression at each time step k

$$\tilde{J}_k = g_k(\bar{x}_k, u_k, \bar{v}_k) + H_{k+1}(f_k(\bar{x}_k, u_k, \bar{v}_k)), \tag{4.4}$$

where H_{k+1} is the cost-to-go function corresponding to the heuristic, to compute a suboptimal control sequence $\{\tilde{u}_k, \tilde{u}_{k+1}, ..., \tilde{u}_{N-1}\}$. Unfortunately, while much less computationally intensive than the optimal solution, this formulation still results in a very large amount of computation for the current problem unless H_{k+1} is set to zero (as is commonly done) or some trivial heuristic.

To further simplify the calculation of a suboptimal control sequence for the UAV control problem a hierarchical variant of the model predictive control approach is proposed. The solution involves splitting the control algorithm into two components. First, a planner which uses a heuristic algorithm, motivated by Equation 4.4, to find a coarse-grained solution. The planner uses a cost function which involves only distance travelled to target, to find the best assignment of UAVs and their best trajectories on the basis of the lowest overall cost. The reasoning is that a key aspect of the problem is to reliably bring the UAVs within proximity of their assigned targets, while avoiding no-fly zones and in particular avoiding being trapped in local minima, which is a problem that often occurs with myopic optimization algorithms. This optimization is performed repeatedly at relatively long time steps every T_P seconds (currently T_P is set to 60 seconds). Two types of planner will be developed, a heuristic algorithm and one that uses a policy rollout approach [6]; their performance will then be compared.

The second component of the control algorithm is a fine-grained one-step-ahead controller (i.e., as per Equation 4.4 but with $H_{k+1}\left(f_k\left(\bar{x}_k, u_k, \bar{v}_k\right)\right) = 0$). The one-step-ahead controller is executed at short time steps (at every measurement update in the algorithm presented in this chapter - this will, however, be modified in a subsequent chapter as the algorithm is further developed). The fine-grained controller has two components in its cost function, one being based on the one-step-ahead estimation error of the system of sensors and the other being based on the cost of divergence from the path computed by the planner.

In addition to the two components of the control algorithm described above, an estimation algorithm is required in order to be able to compute the state estimates $\bar{x}_k\left(I_k\right) = E\left\{x_k | I_k\right\}$. So in summary, the control system consists of a *state estimation algorithm* and a *controller*, where the controller has two

components, a *planner* and a *one-step-ahead controller*. The following sections will describe each component in detail.

Before beginning the next section, it should be noted that with the type of the problem being considered, all the key aspects can be modelled as two dimensional scenarios while maintaining the essential features of the problem. Attempting to model the problem in three dimensional space would hugely complicate the problem without adding anything of significance to the solution. Hence, in the remainder of the thesis, the problem is presented and solved in two dimensional Cartesian space. As is apparent from the references in Chapter 2, this is consistent with the approach generally taken in the literature for similar problems.

4.3 State Estimation Algorithm

A Kalman filter is used for the estimation component of the control algorithm. Section 3.2.3 describes the general Kalman filter in detail. Here it will be quickly summarized and then a filter be developed specifically for our application.

The Kalman filter is an optimal estimator for a discrete time linear dynamic system which can be described by the following general system and measurement equations respectively

$$\mathbf{x}(k+1) = F(k)\,\mathbf{x}(k) + G(k)\,\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k) \qquad k = 0, 1, ...$$

$$\mathbf{z}(k) = H(k)\,\mathbf{x}(k) + \mathbf{w}(k) \qquad k = 1, 2, ...$$

where $\mathbf{x}(k)$ is the state vector of the system, $\mathbf{u}(k)$ is a known (control) input vector, $\mathbf{v}(k)$, $k = 0, 1, ...$ is a sequence of zero-mean white Gaussian noise, $\mathbf{z}(k)$ is the measurement vector, $\mathbf{w}(k)$, $k = 1, 2, ...$ is a sequence of zero-mean white

Gaussian measurement noise, $F(k)$ is the state transition matrix, $G(k)$ is the gain through which the (control) input enters the system, $\Gamma(k)$ is the noise gain and $H(k)$ is the measurement matrix. $F(k), G(k), \Gamma(k), H(k)$ are assumed known.

Consider now the case of a target whose motion is modelled by a piecewise constant white noise acceleration model in a Cartesian coordinate system [2]. Also assume that the x and y components of the target's acceleration noise are independent, with equal standard deviation σ_a, then the following system equation can be easily derived

$$\mathbf{x}\,(k+1) = F(k)\mathbf{x}(k) + \Gamma(k)\mathbf{a}_r(k)$$

where

$$F(k) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Gamma(k) = \begin{bmatrix} 0.5T^2 & 0 \\ T & 0 \\ 0 & 0.5T^2 \\ 0 & T \end{bmatrix}$$

$$\mathbf{x}(k) = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}_k^T$$

$$\mathbf{a}_r(k) = \begin{bmatrix} a_{r_x} & a_{r_y} \end{bmatrix}_k^T$$

and the acceleration noise covariance matrix is

$$Q_a(k) = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}$$

The measurement equation is

$$\mathbf{z}(k) = H(k)\mathbf{x}(k) + \mathbf{w}(k)$$

where

$$H(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{z}(k) = \begin{bmatrix} z_x & z_y \end{bmatrix}_k^T$$

$$\mathbf{w}(k) = \begin{bmatrix} w_x & w_y \end{bmatrix}_k^T$$

In the application being considered the measurements are of angle only, with the possibility of some range information being deduced from signal strength. Since the measurements are actually in polar coordinates, with measurement errors being in range and azimuth, the measurement error covariance matrix, in Cartesian coordinates is [15]:

$$R(k) = \begin{bmatrix} \sigma_{w_r}^2 \cos^2\theta + r^2\sigma_{w_\theta}^2 \sin^2\theta & \frac{1}{2}\left[\sigma_{w_r}^2 - \sigma_{w_\theta}^2\right]\sin 2\theta \\ \frac{1}{2}\left[\sigma_{w_r}^2 - \sigma_{w_\theta}^2\right]\sin 2\theta & \sigma_{w_r}^2 \sin^2\theta + r^2\sigma_{w_\theta}^2 \cos^2\theta \end{bmatrix}_k$$

where $r(k)$ and $\theta(k)$ are the range and azimuth of the target, respectively, $\sigma_{w_r}(k)$ and $\sigma_{w_\theta}(k)$ are the standard deviations of the range and azimuth components of the measurement noise, respectively.

For the tracker simulator used in this work, a separate Kalman filter is run for each target using the measurements from all sensors. There are considerable challenges in multi-target multi-sensor angle-only tracking, leading to quite complex tracking algorithms being required to obtain good performance. The emphasis of this work is on control aspects, rather than the development of a sophisticated tracker, hence only a simple tracker simulator and measurement model are considered. The "actual" measurements that are simulated also follow this model, but retain the key properties of much better measurement accuracy

in angle than in range, and cross-range error that increases linearly with target range, thus avoiding tedious details that are not particularly relevant to the central theme of this work. Data association, i.e., the correct assignment of measurements to targets is also a challenging component of the tracking algorithm. For this work the data association is simulated using known correct assignments of measurements and for now assumed perfect. In real situations, data association would generally not be perfect, with its quality being a function of algorithms used and the specific scenario in question. For a description of a tracker that does address the complexities of multisensor angle-only tracking the reader is referred to [19].

4.4 Planning Algorithm

The planning algorithm is composed of a shortest path calculation algorithm and an assignment algorithm. Two shortest path computation algorithms have been developed; the first is a heuristic algorithm and the second uses a technique called policy rollout. Section 4.4.1 describes the assignment algorithm, and then Sections 4.4.2 and 4.4.3 describe the heuristic and rollout based shortest-path algorithms respectively. Section 4.4.4 then demonstrates and compares the performance of the planner when using heuristic and rollout based shortest-path algorithms. The algorithms were implemented in Matlab for this demonstration.

4.4.1 Assignment Algorithm

A key aim of the control is to position the UAVs as quickly as possible to satisfy some estimation accuracy criterion for all known targets. For direction of arrival sensing, two sensors for a particular target is the minimum number that can give good position estimates quickly. Hence a design choice was made

to assign a pair of UAVs to each known target and optimize each pair's control on performance for their assigned target. Excess available UAVs are not assigned to any particular target and their control is optimized for overall performance on all targets. Note that in some situations one may wish to assign single UAVs or even 3 or more UAVs to a particular target. This would still be workable within the algorithmic approach that is to be proposed. However, for the sake of demonstrating the approach without involving unnecessary complexity but still without loss of generality, we chose to demonstrate the approach with what is likely to be the most commonly used number of UAVs, i.e., two.

The first task once at least one or more targets are detected is hence to assign a pair or pairs of UAVs to known targets. The assignment is based on the shortest distance to target for each UAV taking into account no-fly zones which must be avoided. The no-fly zones are modelled as the sum of one or more circular regions and are assumed to be stationary. In order to perform the assignment, the shortest distance to target is first calculated for each UAV-target combination using one of the two shortest path algorithms that will be described in Sections 4.4.2 and 4.4.3. The assignment is then performed as follows.

Let d_{ik} be the shortest distance calculated for UAV i and target k using the selected shortest path algorithm. Then, once d_{ik} for all possible values of i and k are computed, the next step is to compute the cost for each pair of UAVs to target combination. Let C_{ijk} represent the cost for UAVs i and j to travel to target k. The algorithm then defines the cost to be:

$$C_{ijk} = \max\left(d_{ik}, d_{jk}\right)$$

Without limiting generality, let us now consider the case where the number of UAVs is even and equal to or greater than twice the number of known targets;

the following can then be relatively easily derived. The number of ways pairs of UAVs can be assigned to all the targets is:

$$N_A = \prod_{k=1}^{N_T} C_2^{N_U - 2(k-1)}$$

where N_U is the number of UAVs, N_T is the number of (known) targets and

$$C_k^N = \frac{N!}{k!\,(N-k)!}$$

Then for every possible assignment of pairs of UAVs to all targets, A_m, $m = 1, .., N_A$, compute the sum of the costs for each UAV pair to target combination

$$C_{A_m} = \sum C_{ijk}$$

The optimal assignment, which has the minimum value of C_{A_m} associated with it, is then found. This assignment is performed every time the planner is run and is stored for use by the one-step-ahead controller until the next run of the planner. The first legs of the UAV paths associated with this assignment are also stored.

4.4.2 Heuristic Shortest Path Algorithm

The problem of path planning in an environment with obstacles has in the past been performed with a number of algorithms. For example, the Distance Transform [27], a combination of the Distance Transform and a Map Segmentation Algorithm [40], Visibility Graph [44], [3], Rapidly-exploring Random Trees [37] and a Potential Field Based Approach [30]. In this thesis two new algorithms, one a heuristic algorithm and the other based on an approximate dynamic programming technique called policy rollout [6] are developed as alternate path planning algorithms suitable for real time applications. Let us now consider the heuristic algorithm.

The shortest distance to target is calculated for each UAV-target combination according to the recursive algorithm described by the three functions in Algorithm 4.1, starting with the function *StraightToTarget*. Application of this algorithm usually results in multiple possible paths (e.g., 2 paths when 1 no-fly zone is encountered, 4 paths when 2 no-fly zones are encountered, and so on). The best of these (i.e., the shortest path) is then picked based on the path length of each of the paths. Figure 4.1 shows an example of the case when two no-fly zones are encountered. The initial position (in metres) of the UAV is $x_{UAV} = (19000, 1000)$ and the position of its target is $x_T = (25000, 50000)$. The first no-fly-zone is the union of two circular regions with centres at $x_{c1}^{NFZ1} = (21500, 15000)$, $x_{c2}^{NFZ1} = (28500, 15000)$ and radii $r_1^{NFZ1} = 6500$, $r_2^{NFZ1} = 6500$, respectively, and the second no-fly zone is a circular region with its centre at $x_c^{NFZ2} = (25000, 35000)$ and radius $r^{NFZ2} = 8000$. For this case, four paths are computed, with path 1 being selected as the shortest path.

Some comments regarding computational complexity of Algorithm 4.1 are worthwhile. In computer science the term *time complexity* is generally used to quantify the time taken for an algorithm to run as a function of the size of the input to the problem. The time complexity is commonly expressed using a capital O to represent the order of the function. Using this terminology, the time complexity of Algorithm 4.1 can easily be shown to be $O\left(N_S.2^{N_{NFZ}}\right)$ where N_S is the number of steps traversed (equivalent to distance travelled) from the UAV's initial position to its final position, and N_{NFZ} is the number of no-fly zones that are encountered by the UAV. The important points to note are that the time complexity is a linear function of N_S, the number of steps traversed, and an exponential function of N_{NFZ}, the number of no-fly zones encountered. Since, in general, for the type of problem considered in this thesis, although N_S

Algorithm 4.1 Heuristic Shortest Path Algorithm

Function StraightToTarget:
loop
  Move one step straight toward target
  If reached target then exit loop
  If current step is in obstacle then
    BypassAnticlockwise
    BypassClockwise
    exit loop
end loop

Function BypassAnticlockwise:
loop
  Move one step at smallest possible angle
   anticlockwise from direction of target
  If reached target then exit loop
  If current step is not in obstacle for a step
   that is directly towards target then
    StraightToTarget
    exit loop
end loop

Function BypassClockwise:
loop
  Move one step at smallest possible angle
   clockwise from direction of target
  If reached target then exit loop
  If current step is not in obstacle for a step
   that is directly towards target then
    StraightToTarget
    exit loop
end loop

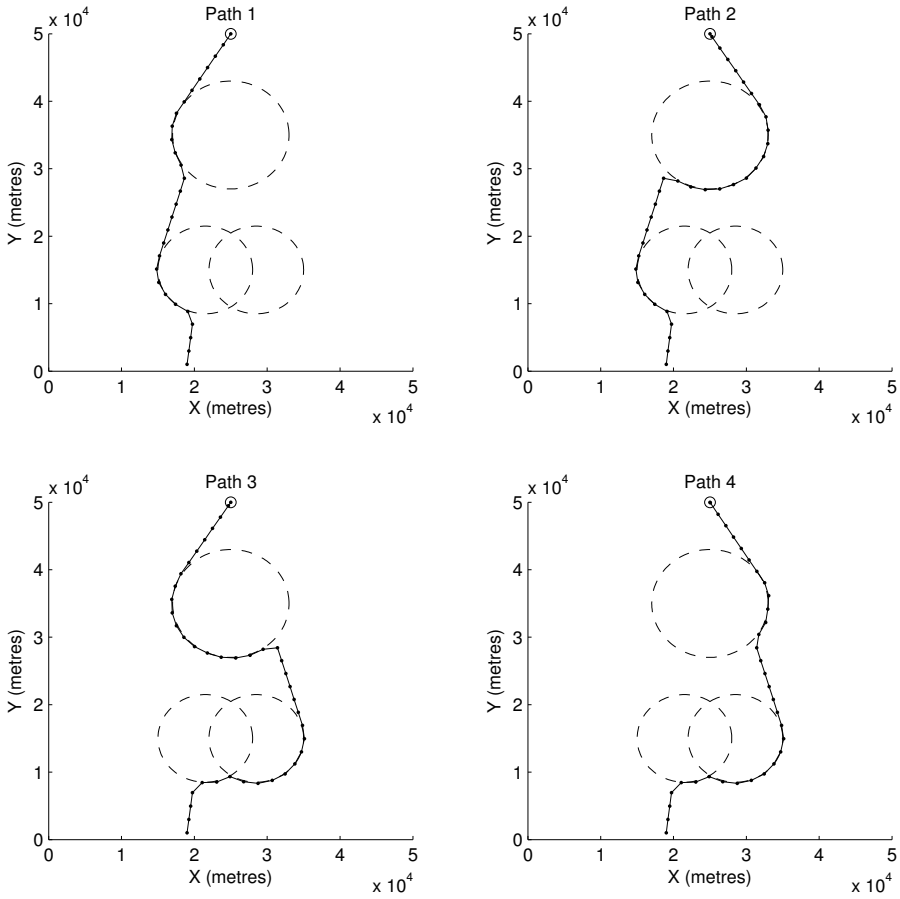

Figure 4.1: Paths Produced by Heuristic Shortest Path Algorithm for Case of Two No-fly Zones

will usually be large, N_{NFZ} will be small, making Algorithm 4.1 well suited to this type of problem.

Another point to note is that, while the shortest path produced by the heuristic algorithm is not optimal, it will generally be quite a reasonable approximation and, importantly, it can be computed very quickly. The algorithm will also be used as a component of the algorithm described in Section 4.4.3, leading to an algorithm which produces paths that are closer to optimal, but at the expense of extra computation.

4.4.3 Shortest Path Algorithm using Policy Rollout

The policy rollout approach [5, p. 314], [6] that has been developed for computing the shortest distance to target is as follows. Consider a single UAV and target; the shortest path from the current UAV position to the target can be estimated using Equation 4.4, where the equation is applied to the simplified problem of determining the shortest distance to target. To describe the rollout approach as used here, first, for all $u_k \in U_k(\bar{x}_k)$, let

$$Q_k(\bar{x}_k, u_k) = g_k(\bar{x}_k, u_k, \bar{v}_k) + H_{k+1}(f_k(\bar{x}_k, u_k, \bar{v}_k)) \tag{4.5}$$

where $Q_k(\bar{x}_k, u_k)$ is known as the *Q-factor* of $(\bar{x}_k, u_k)$ at time k. Consider, the second component of Equation 4.5, i.e., the cost-to-go $H_{k+1}(f_k(\bar{x}_k, u_k, \bar{v}_k))$. To approximate H_{k+1} for each possible next state, $f_k(\bar{x}_k, u_k, \bar{v}_k)$, a heuristic (suboptimal) base policy $\pi = \{\mu_k, ..., \mu_{N-1}\}$, where $\mu_k = \mu_k(\bar{x}_k)$, is applied, which results in the sequence of controls $\{u_k, ..., u_{N-1}\}$ and states $\{\bar{x}_{k+1}, ..., \bar{x}_N\}$. The base policy is the heuristic algorithm that was described in Section 4.4.2.

The Q-factors $Q_k(\bar{x}_k, u_k)$ for all $u_k \in U_k(\bar{x}_k)$ are then computed using Equation 4.5, where $g_k(\bar{x}_k, u_k, \bar{v}_k)$ is the distance travelled from $\bar{x}_k$ to $\bar{x}_{k+1}$ with control u_k applied, and H_{k+1} is the sum of the distances travelled from $\bar{x}_{k+1}$ to

$\bar{x}_N$ for the sequence of states $\{\bar{x}_{k+1}, ..., \bar{x}_N\}$, as computed from the algorithm above, starting from the next state, $f_k(\bar{x}_k, u_k, \bar{v}_k)$ resulting from control u_k, i.e.,

$$H_{k+1} = \sum_{j=k+1}^{N-1} |\bar{x}_{j+1} - \bar{x}_j| \tag{4.6}$$

The rollout control $\tilde{\mu}(\bar{x}_k)$ is then obtained by finding the minimum Q-factor, i.e.,

$$\tilde{\mu}(\bar{x}_k) = \arg \min_{u_k \in U_k(\bar{x}_k)} Q_k(\bar{x}_k, u_k) \tag{4.7}$$

This process is then repeated at each time step k, $k = 0, .., N_R - 1$, where N_R is the number of steps in the rollout policy, to find a (suboptimal) solution for the shortest path. The time complexity of the policy rollout algorithm, for a single UAV and target, is $O\left(N_R.N_S.2^{N_{NFZ}}\right)$, where N_S and $2^{N_{NFZ}}$ are defined in Section 4.4.2.

Note that there are some options in the way that the base policy could be used when performing the policy roll-out algorithm and subsequent assignment. The way that was chosen is as per the following procedure:

1. Perform the policy rollout algorithm for each single target to single UAV combination, with H_{k+1} in Equation 4.6 determined by applying the heuristic algorithm (Algorithm 4.1). As mentioned in Section 4.4.2, application of the heuristic algorithm usually results in multiple possible paths for each UAV-target pair. The best of these, i.e., *the shortest path*, is picked for use in the policy rollout.

2. Using the assignment algorithm (Section 4.4.1) and costs for each UAV-target pair produced by rollout algorithm, perform the assignment.

Two other ways that the policy rollout could have been performed are also worth mentioning. In the first, the policy rollout in step 1 would be performed

on all of the paths produced by the heuristic algorithm, and then selection of the best path, for use by the assignment algorithm, made on the basis of the costs of the rolled out paths. This is more accurate because the rolled out paths, which are in general closer to optimal than those produced by the heuristic algorithm, are now used to select the shortest path for use by the assignment algorithm. This method, however, requires more computation because the rollout algorithm, which requires considerably more computation than the heuristic algorithm, is now executed more times. Usually this approach would probably only produce a small improvement in accuracy, but occasionally (say for highly concave no-fly zones) it may result in a large improvement. Another option is to wait until the assignment is done before performing rollout, i.e., perform the assignment using just the heuristic algorithm and then apply rollout just on the paths corresponding to the final assignments of UAVs to targets. This is a little less accurate but possibly much less computation is required. The choice as to which is best is a compromise between accuracy and computation requirements and depends on:

a. The number of UAVs and targets - for large numbers, the last approach described above may be necessary.

b. The number of no-fly zones in the path of the UAVs - many no-fly zones results in many possible paths in step 1 above, making the first or last approach more attractive. For small numbers of UAVs, targets and no-fly zones the second approach is the best.

4.4.4 Comparison of Heuristic and Rollout Based Shortest Path Algorithms

To demonstrate and compare the performance of the heuristic and rollout algorithms, consider now a scenario with four UAVs, two targets and two no-fly-zones. The initial positions (in metres) of the UAVs are $x_{UAV1} = (16000, 5000)$, $x_{UAV2} = (17000, 5000)$, $x_{UAV3} = (18000, 5000)$, $x_{UAV4} = (19,000, 5000)$, and the positions of the targets are $x_{T1} = (10000, 45000)$, $x_{T2} = (35000, 45000)$. The first no-fly zone consists of the union of two circular regions with centres at $x_{c1}^{NFZ1} = (13000, 17000)$, $x_{c2}^{NFZ1} = (20000, 17000)$ and radii $r_1^{NFZ1} = 5000$, $r_2^{NFZ1} = 5000$, respectively. The second no-fly zone is a circular region with its centre at $x_c^{NFZ2} = (8500, 28000)$ and radius $r^{NFZ2} = 4000$. The step size in the planning algorithm is $|\bar{x}_{j+1} - \bar{x}_j| = 2000$ for all but the final step, and the control options are changes in heading in fixed increments of 5 degrees. The rollout algorithm and the base policy (heuristic) algorithm described in Sections 4.4.3 and 4.4.2 were run for this scenario producing the UAV paths shown in Figure 4.2 in Figure 4.3 respectively. Note that the rollout algorithm produces paths that are a substantial improvement on the paths produced by the heuristic base algorithm. The execution times for the rollout and heuristic algorithm were 4.55s and 0.21s, respectively, on a PC with a 2.4GHz Intel Core Duo CPU. The rollout algorithm clearly involves considerably more computation than simply using the heuristic algorithm, but for small numbers of UAVs, targets and no-fly-zones the computational requirements are not prohibitive. The rollout algorithm resulted in path lengths for UAVs 1 to 4 of 44975m, 44844m, 43789m and 43304m, respectively, whereas the heuristic algorithm gave path lengths of 46288m, 47772m, 44115m and 43445m. The rollout algorithm hence produced a reduction in path lengths of 2.84%, 6.13%, 0.74% and 0.32% respectively for

UAVs 1 to 4, for this particular scenario. Note that the path produced by the rollout algorithm is still not quite optimal with some "kinks" in the paths. Some preliminary experimentation has indicated that this effect is related to the size of the steps in the heuristic algorithm that is used as the base policy for the policy rollout algorithm. Making the size of the last step in the heuristic algorithm just prior to hitting a no-fly zone variable, i.e., making this step just reach the no-fly-zone without a change in heading, is expected to reduce this effect; however, this has not been tested experimentally as yet.

At this point it is worth noting that the choice that was made on how to represent the no-fly zones, i.e., by using the union of circular regions, is somewhat arbitrary and should not be considered as limiting. This method of representation was chosen simply for convenience and ease of implementation. With regard to the algorithms for circumventing the no-fly zones, there are other representations that would be equally valid and would not affect the efficacy of the algorithms, and may even be more appropriate in an actual operational implementation. The no-fly zone shapes can, in fact, be very general in nature; for example they could be represented by the union of convex polygons or even a discrete map, thus allowing any shape at all. For both of these representations, an efficient algorithm for determining whether a UAV has moved into a no-fly zone can easily be implemented. For example, with regard to the union of convex polygons representation, one could test whether a point is within a convex polygon by simply treating the line segments in the perimeter as vectors all touching tip to tail. Let us assume that they are joined this way in an anticlockwise fashion. Now, for each one of these perimeter vectors, form a second vector which starts at the start of the perimeter vector and finishes at the point being tested, then take the cross product of the two vectors. If, for all the perimeter vectors, the cross product is positive, the point being tested

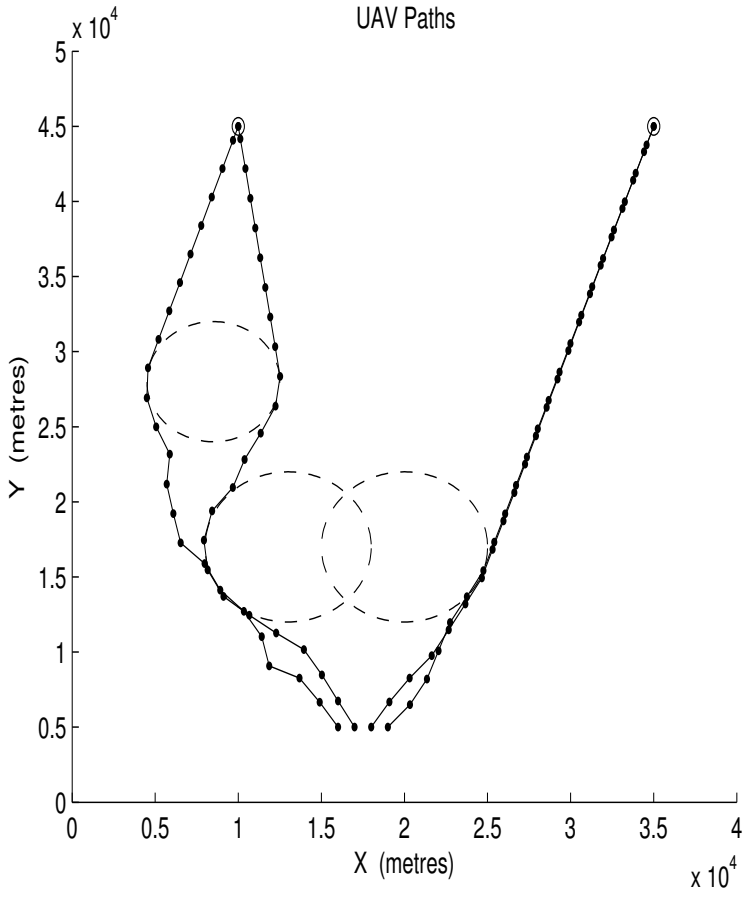

Figure 4.2: UAV Paths Produced by Rollout Based Planner

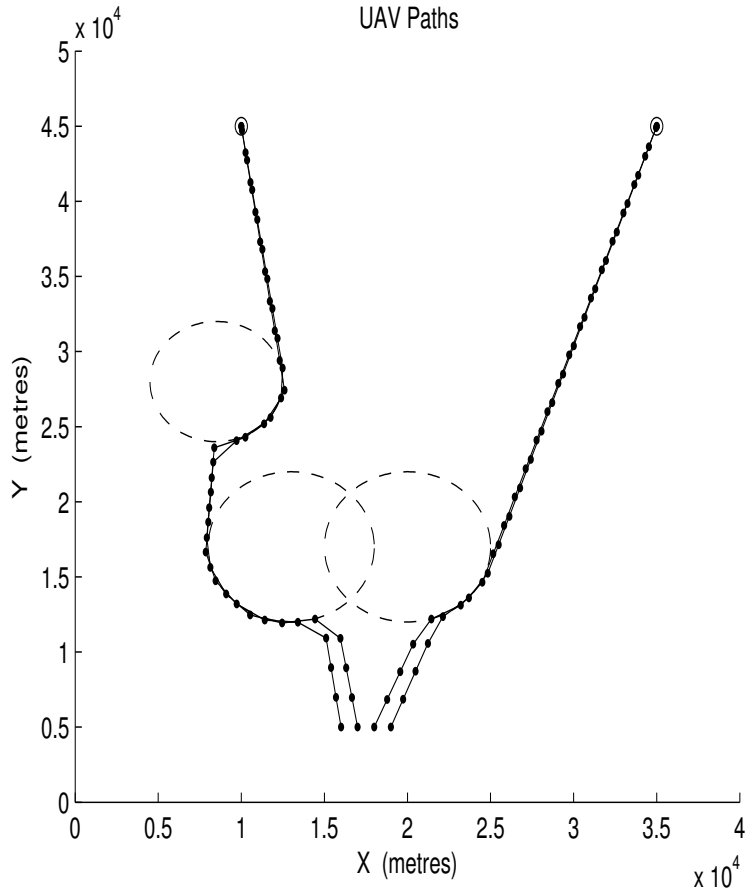

Figure 4.3: UAV Paths Produced by Heuristic Based Planner

is within the convex polygon. For a composite polygon, i.e., a polygon that is the union of convex polygons, if the point is not within any of the constituent polygons, it is obviously not in the composite polygon. The test for the discrete map case is even more obvious, so won't be described here.

4.5 One-Step-Ahead Controller

Control is implemented by changing the direction of travel of the UAVs which travel at a constant pre-set speed, or by setting speed to zero. The controller works on a one-step-ahead basis, where the step length is the time between reception of consecutive measurements. A key point to note is that long term costs are taken into account through a component of the cost function which uses information passed down by the planner.

Let us now consider cycle k of the controller. The first step is to calculate the target state predictions $\hat{x}_j(k+1|k)$ and their associated covariances $P_j(k+1|k)$, $j = 1, .., N_T$ at a predetermined time in the future. This is done by performing the prediction stage of the Kalman filter for each track from its last update time to the time in the future that we are interested in. For this work the time in the future is assumed to be the same for all tracks, set to $t + 2.5$ seconds, where t is the current time. In reality the tracks for the targets will not be updated simultaneously, but a quite complicated (and possibly not very reliable) algorithm would be required to estimate future update times for each track based on past measurements given that each UAV may receive measurements from multiple sources. The approximation made here is good as long as the targets are moving slowly and with small accelerations.

Next, all possible control options for each UAV u_{pi}, $p = 1, .., N_C$, $i = 1, .., N_U$, where N_C is the number of control options, are performed while holding all

the other UAVs states fixed. The control options are changes in heading in fixed predetermined increments, or the selection of zero speed. For each control u_{pi} the expected Kalman filter estimate for each target $\hat{x}_{pij}(k+1|k+1)$ and its covariance $P_{pij}(k+1|k+1)$ is computed using the prediction $\hat{x}_j(k+1|k)$, its associated covariance $P_j(k+1|k)$, and then the measurement prediction $\hat{z}_{pij}(k+1|k)$ and its covariance $S_{pij}(k+1|k)$ for all targets whose state estimates are in range of UAV i. Note that not all possible combinations are tried since to do this all possible choices for a particular UAV must be performed, for all possible combinations of the other UAV control options. Hence, what is done here is only an approximation to one-step-ahead dynamic programming. This is done to speed up computation, and is a good approximation if the distance travelled by each UAV during the prediction period is small compared to the distances between the UAVs. Note also that the control options are tested to see if they would take the UAV into a no-fly-zone or bring the UAV closer than allowable to any of the estimated target positions and are not used if they do so.

Let us define two types of cost now, a cost associated with estimation error C_E and a cost C_P associated with the divergence of the UAV from the path calculated by the path planner for the control option in question.

Consider first C_E. Given the expected estimate covariances for each target $P_{pij}(k+1|k+1)$ consider only the position components giving the covariance matrix for the position estimates

$$P^{pos}_{pij}(k+1|k+1) = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}_{pij}$$

From a practical point of view a useful measure of cost is the length of the major axis of the error ellipse which encloses the target position with probability

P_e. Given the above covariance matrix the length of the major axis is [55]

$$l_{\max} = 2\sqrt{\kappa\lambda_{\max}}$$

where λ_1 is the largest eigenvalue of the covariance matrix and is given by

$$\lambda_{\max} = \frac{1}{2}\left[\sigma_1^2 + \sigma_2^2 + \sqrt{\left(\sigma_1^2 - \sigma_2^2\right)^2 + 4\sigma_{12}^2}\right]$$

and

$$\kappa = -2\ln\left(1 - P_e\right)$$

Now let the cost $C_{E_{pij}}$ be $C_{E_{pij}} = k_\lambda \lambda_{\max_{pij}}$ where k_λ is a positive constant and note that $C_{E_{pij}} \propto l^2_{\max_{pij}}$.

Now consider C_P. Let $C_P = C_d + C_v$ where C_d is the cost associated with the distance d of the UAV from the path calculated by the path planner for that UAV and C_v is the cost associated with v_{diff}, the magnitude of the difference between the component of the UAV velocity vector in the direction of the (directed) line corresponding to the current leg of the path calculated by the planner and the speed of the UAV.

Let the current leg of the path for a particular UAV be represented by the line segment joined by (x_1, y_1) and (x_2, y_2) and the current UAV position be (x, y). Let $\mathbf{a} = (x_2, y_2) - (x_1, y_1)$ and $\mathbf{b} = (x, y) - (x_1, y_1)$ and θ be the angle between $\mathbf{a}$ and $\mathbf{b}$. It can easily be shown that the distance d is

$$d = \frac{\left|\left(x_2 - x_1\right)\left(y - y_1\right) - \left(y_2 - y_1\right)\left(x - x_1\right)\right|}{\sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}}$$

Now let the cost of using control option p for UAV i be $C_{d_{pi}} = k_d d_{pi}^2$ where k_d is a positive constant.

Consider now C_v. Let $\mathbf{v}$ be the (vector) velocity of a particular UAV and $\mathbf{w}$ be the orthogonal projection of $\mathbf{v}$ on the vector $\mathbf{a}$. Let $\mathbf{z}$ be a vector with

magnitude v_M, i.e., the maximum possible speed of the UAV, but with the same direction as $\mathbf{a}$. It can easily be shown that

$$v_{diff} \triangleq |\mathbf{z} - \mathbf{w}| = |\mathbf{z}| - \frac{\mathbf{v} \cdot \mathbf{a}}{|\mathbf{a}|}$$

Let the components of $\mathbf{v}$ be (v_x, v_y), and note that $|\mathbf{z}| = v_M$, then

$$v_{diff} = v_M - \frac{[v_x (x_2 - x_1) + v_y (y_2 - y_1)]}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

Now let the cost using control option p for UAV i be $C_{v_{pi}} = k_v v^2_{diff_{pi}}$, where k_v is a positive constant. The total cost of using control option p for UAV i is then

$$C_{pi} = C_{d_{pi}} + C_{v_{pi}} + \sum_{j \in S} C_{E_{pij}}$$

If the UAV being considered has not been assigned to a target, the set S is the set of all targets that are in range of the UAV. If the UAV has been assigned to a target then the set S is only the assigned target, assuming it's in range. If the UAV being considered has not been assigned to any particular target and is not within range of any of the targets the above cost function has the same value for every possible option, hence is not useful. For that situation it was decided (somewhat arbitrarily) that the UAV in question should continue on a path as near as practicable to its current path while still avoiding obstacles.

4.6 Simulations and Results

Simulation software has been written to assess the approach and some preliminary assessment performed with simulation experiments. Two sets of simulations were performed; in the first the new hierarchical MPC algorithm using the heuristic shortest path algorithm is compared with a "myopic" one-step-ahead

algorithm, and in the second, the two versions of the MPC algorithm are compared. The simulations and associated results are described in the following two subsections.

4.6.1 Comparison of the Hierarchical MPC Algorithm with a Myopic Controller

In this set of simulations, two algorithms were simulated, one being the new MPC algorithm (using the heuristic based shortest path algorithm) and the other being a myopic one-step-ahead control algorithm which was used for comparison. The myopic control algorithm was based on Equation 4.4 with H_{k+1} set to zero (and with no guidance from a planner). The scenario that was considered consisted of two slowly moving targets, travelling at 5 m/s, and four UAVs, travelling at a speed of 30 m/s with direction of travel being the controlled variable, with a no-fly-zone consisting of two intersecting circles. The simulation corresponded to a scenario of 25 minutes duration. The targets were mechanically scanned radars with a 4 deg. beamwidth and a scan rate of 72 deg./s. The UAVs each had an ES sensor with a detection range of 40 km for the radars in question, with a bearing measurement error standard deviation of 5 deg. The tracker Kalman filters and the controllers assumed an acceleration process noise standard deviation of 1 m/s^2. The planner update interval was 60 sec. Figures 4.4 and 4.5 show the UAV trajectories and tracker position estimate errors respectively for the myopic controller. The UAV trajectories and tracker position estimate errors respectively for the model predictive controller are shown in Figures 4.6 and 4.7.

The execution times for the myopic and model predictive controllers were 273s and 313s, respectively, on a PC with a 2.4GHz Intel Core Duo CPU. The

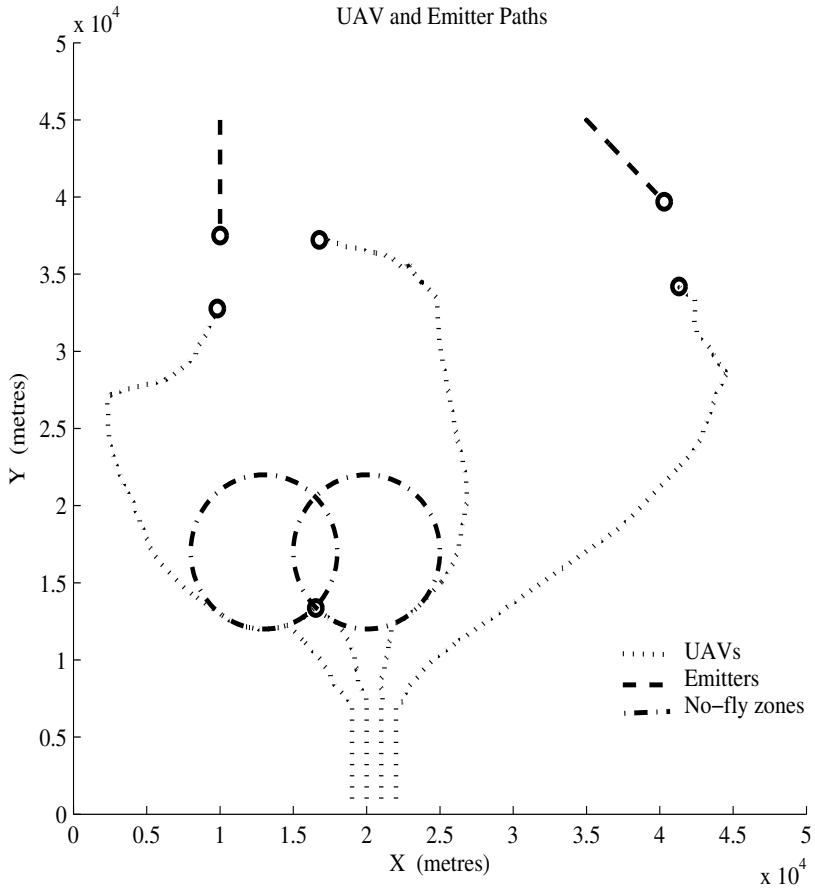

Figure 4.4: UAV and Emitter Trajectories Produced by Myopic Controller

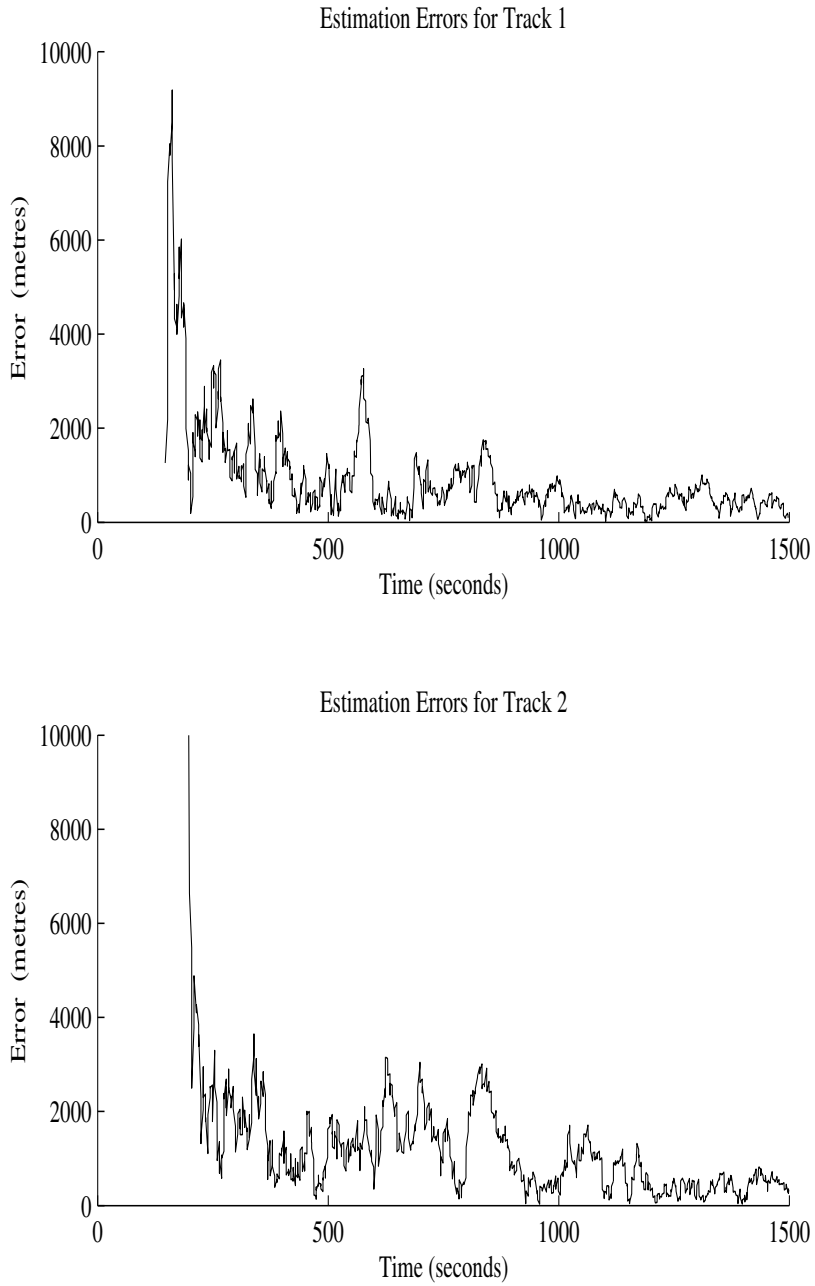

Figure 4.5: Emitter Position Estimation Errors for Myopic Controller

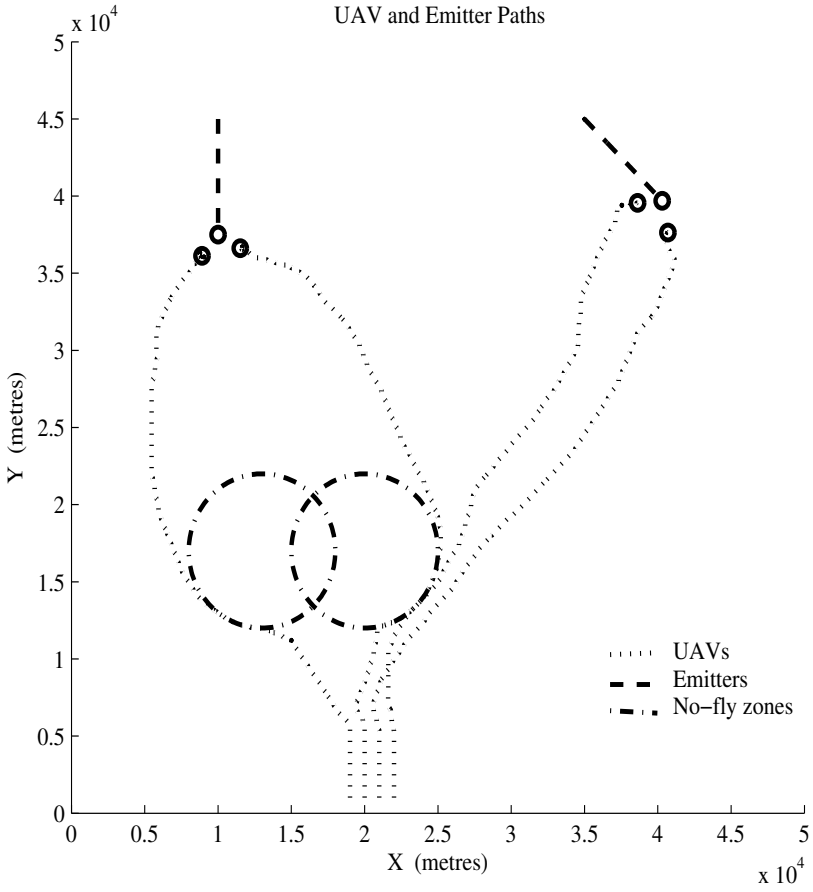

Figure 4.6: UAV and Emitter Trajectories Produced by Model Predictive Controller

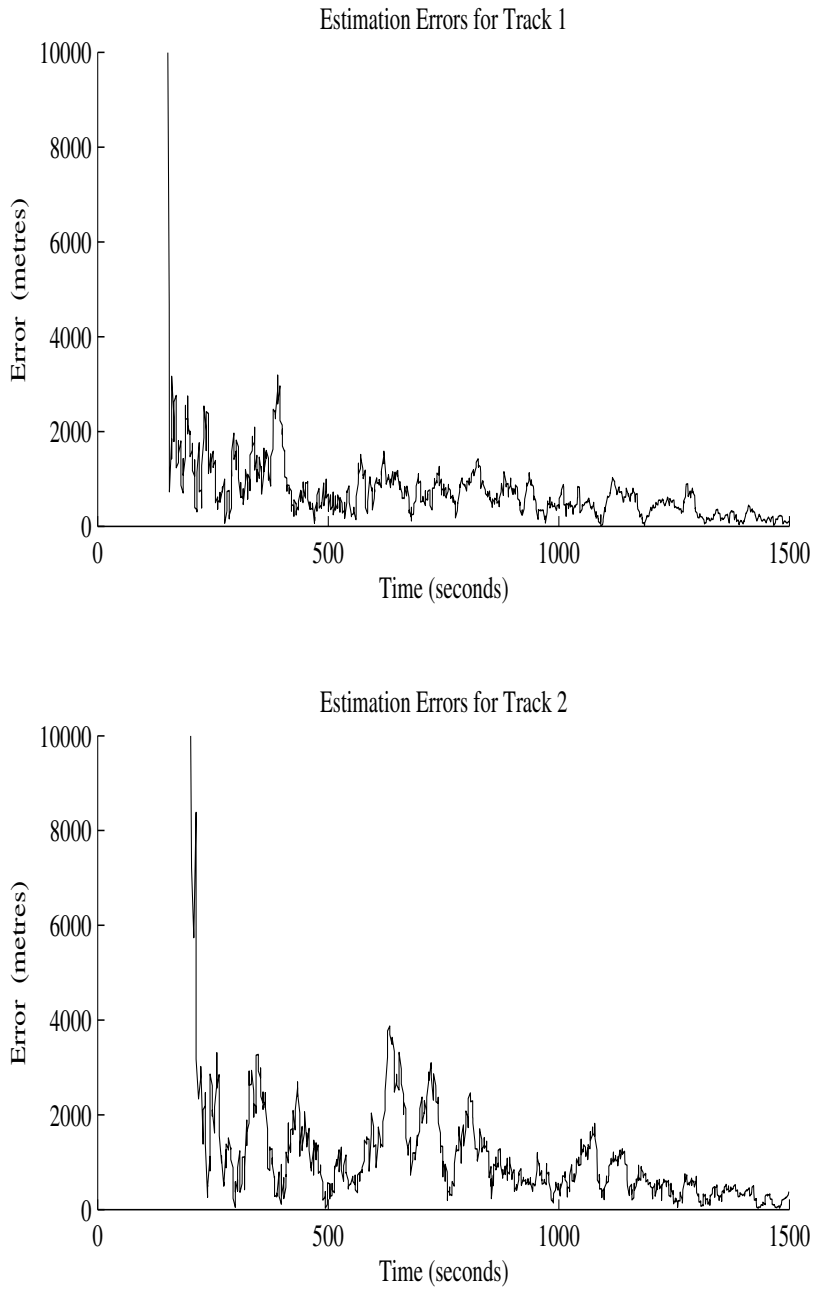

Figure 4.7: Emitter Position Estimation Errors for Model Predictive Controller

execution time for the model predictive controller is hence approximately 15% more than for the myopic algorithm, a relatively small penalty for taking into account long term cost. The most obvious performance difference is that for the myopic controller one of the UAVs becomes trapped by the no-fly-zone whereas with the model predictive controller all the UAVs successfully bypass the no-fly-zone. Another difference is that with the model predictive controller the UAVs come into close proximity of the targets more quickly. With regard to tracker estimate errors the model predictive controller appears to give somewhat smaller errors in the latter part of the scenario. To confirm this and to determine the extent of improvement a statistical analysis involving a large number of runs would be required.

4.6.2 Comparison of the Two Versions of the Hierarchical MPC Algorithm

In this set of simulations the two versions of the hierarchical MPC algorithm are implemented, i.e., using the heuristic planner and then the rollout based planner, and their performance compared. The scenario that was considered consisted of two slowly moving targets travelling at 5 m/s, four UAVs travelling at a speed of 30 m/s with direction of travel being the controlled variable, and two no-fly zones obstructing the paths of the UAVs. The simulation corresponded to a scenario of 25 minutes duration. The targets were mechanically scanned radars with a 4 degree beamwidth and a scan rate of 72 deg./s. The UAVs each had an ES sensor with a detection range of 40 km for the radars in question, with a bearing measurement error standard deviation of 5 deg. The tracker Kalman filters and the controllers assumed an acceleration process noise standard deviation of 1 m/s^2. The planner update interval was 60 sec. The UAV trajectories for the

controller using heuristic path planning are shown in Figure 4.8, and Figure 4.9 shows the UAV trajectories for the controller using rollout path planning. The execution times for the controller using heuristic path planning and the controller using rollout path planning were 259s and 316s, respectively, on a PC with a 2.4GHz Intel Core Duo CPU. For this scenario, the performance differences are essentially a small reduction in distance travelled to target by the UAVs when the rollout path planner is used, but at the cost of a small increase in computation time. The relative performances of the two planning algorithms are expected to vary considerably with changes in scenario; however, in general the heuristic algorithm is expected to be faster, but at the expense of less optimal UAV trajectories.

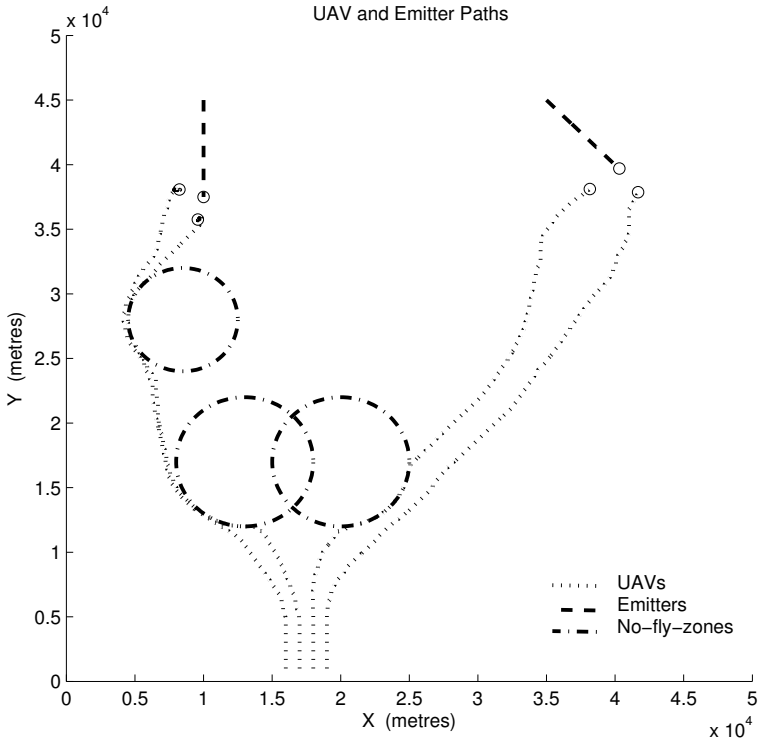

Figure 4.8: UAV and Emitter Trajectories for Controller with Heuristic Path Planning

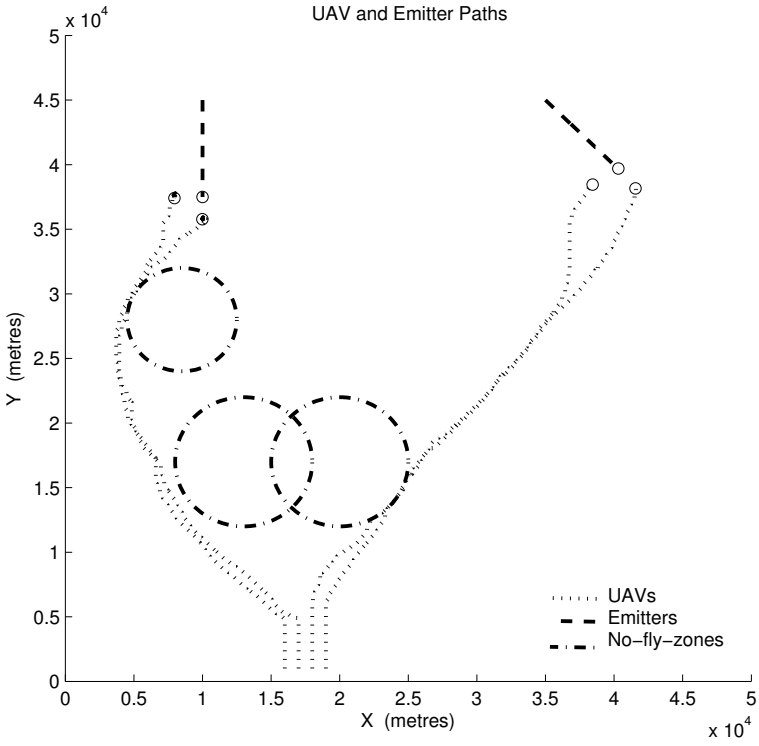

Figure 4.9: UAV and Emitter Trajectories for Controller with Rollout Path Planning

Chapter 5

Introduction of a Fixed-Wing UAV Dynamics Model

In Chapter 4, algorithms for the control of UAVs which used a very simple model of UAV dynamics were presented. The reasoning behind this approach was to first develop the general algorithms for controlling UAVs in the type of application being considered without the distraction of details regarding specific UAV types. The algorithms could then later be tailored to the specific system that is to be implemented. In this chapter, one class of UAV is investigated. For the purpose of example, fixed-wing UAVs are considered and an appropriate dynamics model developed. This choice of UAV type was made for two reasons:

- Optimal control of this type of UAV within the constraints of the problem being considered is quite challenging because of the limited range of dynamics available, for example a fixed-wing UAV cannot simply stop and instantaneously change direction to avoid a no-fly zone.
- The work presented in this thesis will contribute to a research and development program that incorporates Aerosonde UAVs [24], [16]. The

dynamics model used in this chapter is loosely based on this type of UAV so as to allow easy transfer of algorithms and knowledge gained to that application.

In addition to developing the fixed-wing dynamics model, some additional algorithm modifications needed to be made in order to enable the controller to perform successfully with the new dynamics. Sections 5.1 to 5.4 describe the dynamics model that is used and the other modifications that were required. Section 5.5 then presents the results of simulations of the new controller.

5.1 Fixed-Wing Dynamics Model

Consider a somewhat idealised model of a fixed-wing UAV which is limited to travelling at a fixed speed but can turn in the horizontal plane over a range of turn rates corresponding with transverse accelerations up to $\pm a_{t_{\max}}$. In any individual turn the magnitude of the transverse acceleration a_t is constant and its direction is toward a fixed point, the centre of the turn. This type of turn will be referred to as a coordinated turn. Using the coordinated turn model, the transition from the UAV state $\mathbf{x}(k)$ at time t_k to $\mathbf{x}(k+1)$ at time t_{k+1} is described by the following matrix equation [2, p. 187]

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & \frac{\sin\omega_k T}{\omega_k} & 0 & -\frac{(1-\cos\omega_k T)}{\omega_k} \\ 0 & \cos\omega_k T & 0 & -\sin\omega_k T \\ 0 & \frac{(1-\cos\omega_k T)}{\omega_k} & 1 & \frac{\sin\omega_k T}{\omega_k} \\ 0 & \sin\omega_k T & 0 & \cos\omega_k T \end{bmatrix} \mathbf{x}(k) \tag{5.1}$$

where

$$\mathbf{x}(k) = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k \end{bmatrix}^T$$

and

$$\omega_k = \frac{a_t(k)}{v} \tag{5.2}$$

In the above, $a_t(k)$ is the transverse acceleration at time $t_k \leq t < t_{k+1}$, v is the UAV's speed, ω_k is the angular velocity of the UAV at time $t_k \leq t < t_{k+1}$, and T is the time interval $T = t_{k+1} - t_k$. When the UAV is not turning, the state transition equation is simply

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k)$$

Using this model leads us to a new one-step-ahead controller as follows. Instead of control being implemented by (instantaneously) changing direction or speed as was the case previously, control is now implemented by applying an appropriate transverse acceleration at each controller update. The choice of possible control options that could be used must now be made. As there is no clear best choice, the following was chosen using judgement regarding what should work well. Firstly the maximum transverse acceleration must be determined. The acceleration must be within the capabilities of a realistic unmanned fixed-wing aircraft and additionally not result in manoeuvres that would severely affect performance of the sensors on the UAVs. With regard to the sensors, at this stage, details of sensor implementation are not known so limitations resulting from sensor choice and mounting details cannot be considered, although it is possible that certain aircraft banking angles or indeed aircraft headings could be problematic. This consideration is hence left for a later date; for now the assumption will be that transverse accelerations of up to 5 m/s^2 (approximately 0.5 G's) are allowable; this transverse acceleration should be easily achievable

in terms of aircraft dynamics. The next detail to determine is the number of possible controls that are to be considered and the way they will span the range of -5.0 to $+5.0$ m/s^2 transverse accelerations that the aircraft will be allowed to perform. The choice that was made is a compromise between computational load and granularity in possible controls allowed. The control options are as follows. Eleven possible controls (for each UAV) are considered, the controls being $-5, -5/2, -5/4, -5/8, -5/16, 0, 5/16, 5/8, 5/4, 5/2$, and 5 m/s^2. Note that the controls are not equally spaced, incorporating a zero turn rate, and then for both left and right turns, acceleration options that are twice the next smaller option forming a binary sequence up to the maximum allowed acceleration.

5.2 Avoidance of No-fly Zones

Incorporating the fixed-wing UAV dynamics model brings with it some difficulties not encountered with the simple model considered previously. These issues arise because now the UAV cannot instantaneously change its velocity. One problem is that of avoidance of no-fly zones; this will be considered now. With the control algorithm described in Chapter 4 the UAV can come to positions where, within the dynamical constraints now being considered, it cannot manoeuvre around a no-fly zone. The problem is that the planner only produces approximate flight paths which are used as a guide by the controller and not followed exactly. Hence the UAV cannot simply rely on the one-step-ahead controller, as presented in Chapter 4, for avoidance of no-fly zone intrusion. One solution to this is to incorporate some additional look-ahead in the one-step-ahead controller.

Before developing the look-ahead algorithms, a modification was made to

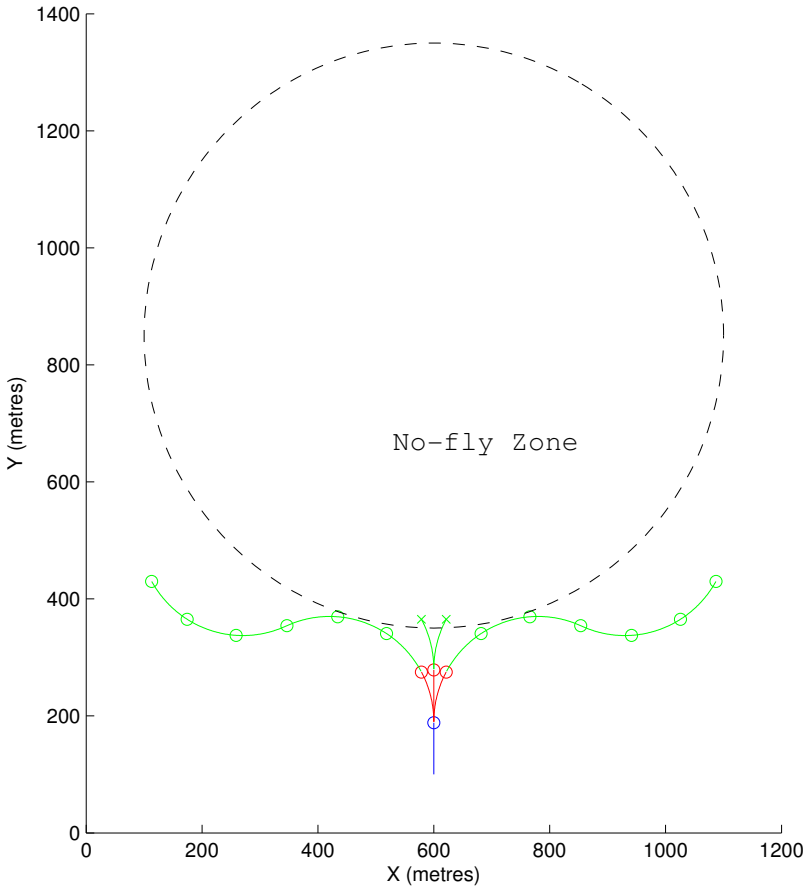

Figure 5.1: Multi-step Look-ahead Algorithm Example

the original low-level (i.e., one-step-ahead) controller algorithm so that the controller updates now occur at (fixed) 3 second intervals instead of at the receipt of each new measurement. This leads to a more predictable algorithm in terms of stability and with regard to the additional look-ahead that will be now be described. Now with a fixed update interval, the number of steps of look-ahead that is required can be calculated. Noting that we have set a maximum transverse acceleration of $a_{t_{\max}} = 5$ m/s^2, and referring to Equation 5.2 the maximum turn rate for a UAV travelling at 30 m/s is $\omega_{\max} = 5/30 = 0.1667$ rad/s or 9.55 °/s. To perform a 180 degree turn would take $180/9.55 = 18.85$ seconds which corresponds to 6.28 update intervals. Performing a look-ahead of at least 7 update intervals would thus guarantee that the UAV would be able to avoid a wide range of possible obstacles. Let us hence make the requirement that a look-ahead of 7 update intervals is to be performed at each update of the low-level controller. The most obvious way of doing this is to perform multi-step-ahead dynamic programming at each update; however, this approach results in a prohibitively large computational requirement as is demonstrated by the following. Consider the number of sequences of controls that need to be dealt with in a multi-step dynamic programming algorithm for the case of N_s steps, N_u UAVs and N_c controls per UAV at each step. It can be easily shown that the number of sequences of controls that need to be considered is

$$\mathcal{N}_{N_s}^{N_c,N_u} = \left[(N_c)^{N_u}\right]^{N_s}$$

Consider the case of $N_c = 11$, $N_u = 2$, $N_s = 7$, then using the above equation we find that approximately 3.8×10^{14} sequences of controls must be considered, i.e., the corresponding UAV states computed, whether the states correspond to no-fly zones determined and if they don't, then their costs calculated and the lowest cost sequence of controls determined. Realistically, doing this at

every update of the controller is not a viable approach. Clearly some form of simplification must be made to arrive at a usable algorithm for performing the multi-step look-ahead. The algorithm developed to achieve this is described in the following paragraphs. In essence the algorithm performs an approximation to one-step-ahead dynamic programming, but with multi-step-ahead testing of avoidance of no-fly zones for a subset of possible controls; the results of the avoidance of no-fly zones tests are then used to constrain the controls that are tried in the first step i.e., the one-step-ahead dynamic programming step.

Consider first a description of the algorithm by example. Referring to Figure 5.1 consider a UAV in the position shown by the blue circle heading in the positive y direction. At this point a decision must be made as to which control to apply for the next 3 seconds (until the next update of the low-level controller). The figure shows the results of three possible controls in red. Choosing a zero turn rate would take the UAV to the middle red circle after 3 seconds, performing a 5 m/s^2 right turn would take it to the rightmost red circle and performing a 5 m/s^2 left turn would take it to the leftmost red circle. Without loss of generality, let us assume that only these three controls are available; then with the algorithm described in Chapter 4, the cost of the three controls would be calculated and the lowest cost control selected and applied until the next controller update. If this happens to be the control associated with zero turn rate, on subsequent updates there will be no controls available that would enable avoidance of the no-fly zone, hence the UAV would fail to avoid the no-fly zone. With the new algorithm, prior to calculating costs, each possible control is first tested to see if for the subsequent 6 updates at least one path can be found that avoids all no-fly zones. The controls that satisfy the test then have their costs calculated and the control with lowest cost is subsequently applied. In Figure 5.1 both the left and right turns pass the test but the zero turn rate does not,

hence the zero turn rate will not be chosen irrespective of its cost. Of the two remaining controls, the one with the lower cost will then be applied for the next 3 seconds. This algorithm is repeated for each UAV at each update.

Now to the test to see if after 3 seconds a control will take the UAV to a position where it is still possible to find a path that avoids all no-fly zones. An algorithm to achieve this is outlined as pseudo-code in Algorithm 5.1. Note that this algorithm is performed for all N_c controls.

Algorithm 5.1 Depth-first Search Algorithm for Avoidance of No-fly Zones

```
use control to update UAV_State;
CanAvoidNoFlyZones = NoFlyZoneTest(UAV_State);

function CanAvoidNoFlyZones = NoFlyZoneTest(UAV_State):
set CanAvoidNoFlyZones = false;
test if current UAV position is in a no-fly zone;
if UAV in a no-fly zone
   return to calling function (with CanAvoidNoFlyZones = false);
end;
CanAvoidNoFlyZones = RecursiveNoFlyZoneTest(UAV_State);

function CanAvoidNoFlyZones = RecursiveNoFlyZoneTest(UAV_State):
Depth = Depth + 1;
if Depth > N_s
   CanAvoidNoFlyZones = true;
   return to calling function;
end;
set CanAvoidNoFlyZones = false;
for selected subset of controls (i.e., turn rates)
   update UAV_State;
   test to see if UAV position is inside of a no-fly zone;
   if UAV is not in no-fly zone
      CanAvoidNoFlyZones = RecursiveNoFlyZoneTest(UAV_State);
      if CanAvoidNoFlyZones = true
        return to calling function;
     end;
  end;
end;
```

Algorithm 5.1 recursively performs a depth-first search [10] to find a path that over the next N_s steps avoids all the no-fly zones. As soon as one path is found the algorithm returns *CanAvoidNoFlyZones* = true. If no path can be found the algorithm returns *CanAvoidNoFlyZones* = false.

Let us now determine the computational requirements of Algorithm 5.1. For step 1 (i.e., the approximation to one-step-ahead dynamic programming) each UAV is considered individually, hence for each UAV there is N_c controls, giving a total of $N_c \times N_u$ controls in total to be considered for all the UAVs. For steps 2 to N_s, each UAV is again considered individually; however, now only a subset of controls is considered. Let the number of controls considered be N_c'. Consider first, step 2. For a single UAV, for each control from step 1 there is N_c' controls at step 2, hence, $N_c \times N_c'$ sequences of controls. In total, for all the UAVs there is $N_c \times N_c' \times N_u$. By induction, the number of possible sequences at step N_s is then

$$\mathcal{M}_{N_s}^{N_c,N_c',N_u} = N_c \times (N_c')^{N_s-1} \times N_u$$

Let $N_c = 11$, $N_c' = 3$, $N_u = 2$, and $N_s = 7$, then, using the above equation, the maximum possible number of control sequences that need to be considered is approximately 1.6×10^4, which is 10 orders of magnitude less than for the full dynamic programming algorithm above. Note also that this is the *maximum* number of possible sequences to consider. Because the algorithm involves a depth-first search and needs to only find one sequence that avoids no-fly zones for each control in step 1, the number of sequences considered is usually much less than the maximum. In the case when there are no no-fly zones near any of the UAVs the number of sequences is simply N_c.

Note that $N_c' = 3$ was used in the example above even though the full set of controls available is $N_c = 11$; this can be justified by the fact that in steps 2 to N_s only a search for the existence of *one possible path* is being performed. No

costs are calculated and the path found is not necessarily the one that will be used in subsequent updates. In the current implementation of the algorithm, the three controls that are used are $a_{t_1} = 0$ m/s^2, $a_{t_2} = -5$ m/s^2 and $a_{t_3} = 5$ m/s^2, i.e., straight ahead, hard left and hard right turns. Using $N_c' = 3$ instead of the full set of 11 controls simply introduces some coarseness to the search for a possible path, which is not a significant issue since the updates are only 3 seconds apart. Note that the first step of the algorithm uses the *full set of controls* and the controller calculates the cost of all of them subject, of course, to satisfaction first of no-fly zone avoidance.

Note that an example of an obstacle that could still "trap" the above multi-step look-ahead algorithm is an object with a deep but very narrow concavity of width less than twice the turn circumference of the UAV. Putting a limitation on the no-fly zones to not have such concavities is a simple and not very restrictive way of dealing with this. Note also that there is a very simple and computationally efficient variant of the above algorithm that simply involves testing whether, for each possible control in the first update, either a hard left turn or a hard right turn avoids all no-fly zones for the subsequent 6 updates. With this algorithm the maximum number of possible sequences is simply $2N_c$. If the no-fly zones are spaced more than one turn diameter apart this algorithm would be expected to work quite well and doesn't have the disadvantage described above. It does have a disadvantage in that if distinct no-fly zones are very close to one another the algorithm will disallow paths between the no-fly zones; this, however is not considered a significant restriction in the current application.

5.3 Improved Approximation to One-Step-Ahead Dynamic Programming

Using the set of controls described in Section 5.1, in particular, making the controls differ only in transverse acceleration up to a maximum value, and also having the relatively short update interval of 3 seconds, enables another change to be made which improves on the approximation to one-step-ahead dynamic programming that was used earlier in Chapter 4. Previously, at each update k, instead of considering every possible combination of controls leading to all possible states at update $k+1$, an approximation was used where all possible controls for each UAV were considered *while keeping the other UAV states fixed at their update k values.* This led to a reduction in the number of controls to be considered for N_u UAVs from $(N_c)^{N_u}$ controls to $N_c N_u$ controls, at the cost of some decrease in the accuracy of the control cost calculation. Now, instead of keeping the other UAV states at their k^{th} update value, the states are extrapolated to their $(k+1)^{th}$ update values using the zero transverse acceleration control. This change improves the approximation to the full one-step-ahead dynamic programming algorithm without significantly increasing the amount of computation required. This is because the difference between the position components of any two of the possible states at update $(k+1)$ for a particular UAV are smaller than the difference between any one of the possible positions at update $(k+1)$ and the position at update k. Another change has also been made, i.e., for each UAV, once its lowest cost control is determined, it is used in place of the zero acceleration control for that UAV when computing the lowest cost control for all subsequent UAVs considered. This further improves the cost calculation approximation. The pseudocode in Algorithm 5.2 shows the algorithm with the improvements described above.

Algorithm 5.2 Improved Approximation to One-step-ahead Dynamic Programming

for $i = 1$ to N_u
  calculate *Ref_ UAV_ State(i)* by applying zero turn rate control to UAV i;
end;
for $i = 1$ to N_u
  for $j = 1$ to N_c
    calculate *Test_ UAV_ State(i, j)* by applying control j to UAV i;
    CanAvoidNoFlyZones = *NoFlyZoneTest(Test_ UAV_ State(i, j))*;
    if *CanAvoidNoFlyZones* = true
      calculate *Cost*(i, j) associated with *Test_ UAV_ State(i, j)*,
      Ref_ UAV_ State(k), $k = 1...N_u, k \neq i$ and predicted target
      state estimates;
    else
      Cost$(i, j) = \infty$;
    end;
  end;
  find the control number $j_{\min}$ of the control that gives the
  lowest cost, i.e., $j_{\min} = \arg\min_j$ *Cost*(i, j);
  set *Selected_ Control(i)*$= j_{\min}$;
  set *Ref_ UAV_ State(i)*=*Test_ UAV_ State*$(i, j_{\min})$;
end;

5.4 Maintenance of Minimum Distance to Target

As with avoidance of no-fly zones, incorporating the fixed-wing UAV dynamics model brings with it added difficulties in maintaining a minimum distance between each UAV and any of the targets. This again arises because the UAV cannot instantaneously change its velocity. However, using Algorithm 5.1 to maintain the minimum distance has the problem that the targets are potentially moving, albeit fairly slowly; also their positions and motion are not known exactly, with the estimates varying somewhat from one update to the next. Hence making predictions about the targets' positions and associated no-go regions around them multiple updates into the future can have sufficient inaccuracy associated with it for the UAVs to end up in positions where their next update can't take them out of a no-go region around a target, irrespective of the control used, when using the Algorithm 5.1. However, because the no-go regions around the targets are simply based on range, a simple approach involving the addition of a cost associated with nearness to a target can be used instead. The approach is as follows. Let R_t be the range of the UAV from the *nearest* target, and $R_{t_{\min}}$ be the minimum allowable range, then define the "range cost" C_R associated with the UAV's position to be $C_R = K_R \left(R_{t_{\min}} - R_t\right) / R_{t_{\min}}$ when $R_t < R_{t_{\min}}$ and $C_R = 0$ when $R_t \geqslant R_{t_{\min}}$, where K_R is a positive constant which is made sufficiently large to make C_R quickly become much larger than other costs associated with the UAV's position as R_t decreases below $R_{t_{\min}}$. This cost is then added to the other costs associated with position of the UAV in question. The effect is that there will be a strong propensity for the UAV to veer away from targets for which $R_t < R_{t_{\min}}$ while having no effect on UAV behaviour as long as $R_t \geqslant R_{t_{\min}}$. This approach was used for maintaining distance from targets

and found to work quite successfully. Note that this approach was not used for the no-fly zones because the method used there has the flexibility to deal with any shape of no-fly zone, and is better at strictly keeping the UAVs outside of the no-fly zone perimeters when the perimeters are fixed and known exactly.

5.5 Simulations

Simulation software was written (in Matlab) to assess performance of the model-predictive controller with the above dynamics incorporated. Two simulations were performed, one using the rollout based planner and the other using the heuristic based planner. The scenario that was considered consisted of two slowly moving targets travelling at 5 m/s, four UAVs travelling at a speed of 30 m/s with direction of travel being the controlled variable, and two no-fly zones obstructing the paths of the UAVs. The first no-fly zone was composed of the union of 3 circles with centres at (18000, 15000), (26500, 20000), (35000, 15000) and radius 6000 metres. The second no-fly zone consisted of a single circle with centre (11000, 27000) and radius 5000 metres. The scenario was designed to be quite challenging. To that end one of the no-fly zones has its concave side facing the starting point of the UAVs so as to

- maximize the potential for entrapment in local minima,
- test the ability of the multi-step-look-ahead algorithm to guarantee that the UAVs can turn in time to avoid intrusion into the no-fly-zones,
- and challenge the path planning algorithm's ability to find a path around the concavity in the no-fly zone.

The second no-fly zone adds complexity to the planning component of the algorithm by testing that the algorithm recursions perform correctly.

The simulation corresponded to a scenario of 30 minutes duration. The targets were mechanically scanned radars with a 4 degree beamwidth and a scan rate of 72 deg./s. The UAVs each had an ES sensor with a detection range of 40 km for the radars in question, with a bearing measurement error standard deviation of 5 deg. The tracker Kalman filters and the controllers assumed an acceleration process noise standard deviation of 1 m/s^2. The planner update interval was 60 sec. Figure 5.2 shows the UAV trajectories for the controller using heuristic path planning. The UAV trajectories for the controller using rollout path planning is shown in Figure 5.3. The execution times for the controller using heuristic path planning and the controller using rollout path planning were 240 seconds and 291 seconds, respectively, on a PC with a 2.4GHz Intel Core Duo CPU.

Figure 5.4 shows an enlarged view of the terminal phase trajectories of the two UAVs that approach target 1 for the case of the controller with rollout based path planning. Here the outcome of the inability of the UAVs to stay in a fixed position is demonstrated. Instead of stopping at some optimal position, the UAVs must constantly perform slow (5.0 m/s^2) turns to maintain an approximately optimal geometry with regard to the target in question. Within the constraints of the UAV dynamics, the control algorithm was found to perform very well in this regard. One would expect that once the UAVs settle into their terminal phase they would tend to move into positions that are approximately 90^o apart when viewed from the target and at the minimum allowable distance from the target. The reader is referred to [13] for an analysis of optimal sensor geometries for angle of arrival sensors. Note that the small circles at the end of the UAV and target trajectories in Figure 5.4, which represent the positions of the UAVs and target at the termination of the scenario, demonstrate this

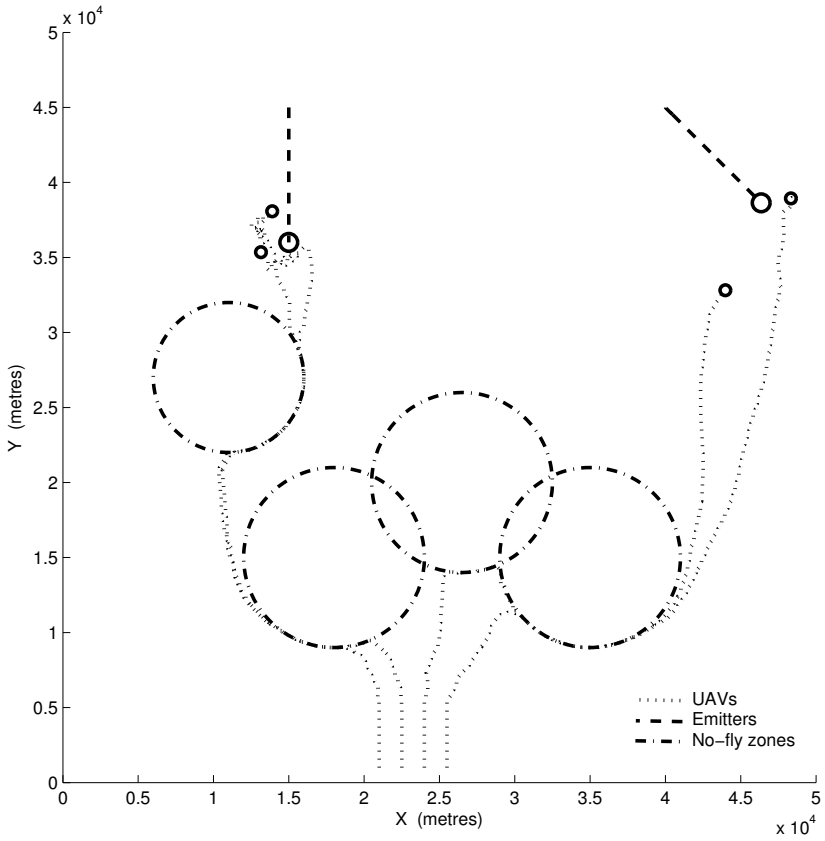

Figure 5.2: Fixed-Wing UAV and Emitter Trajectories for Controller with Heuristic Path Planning

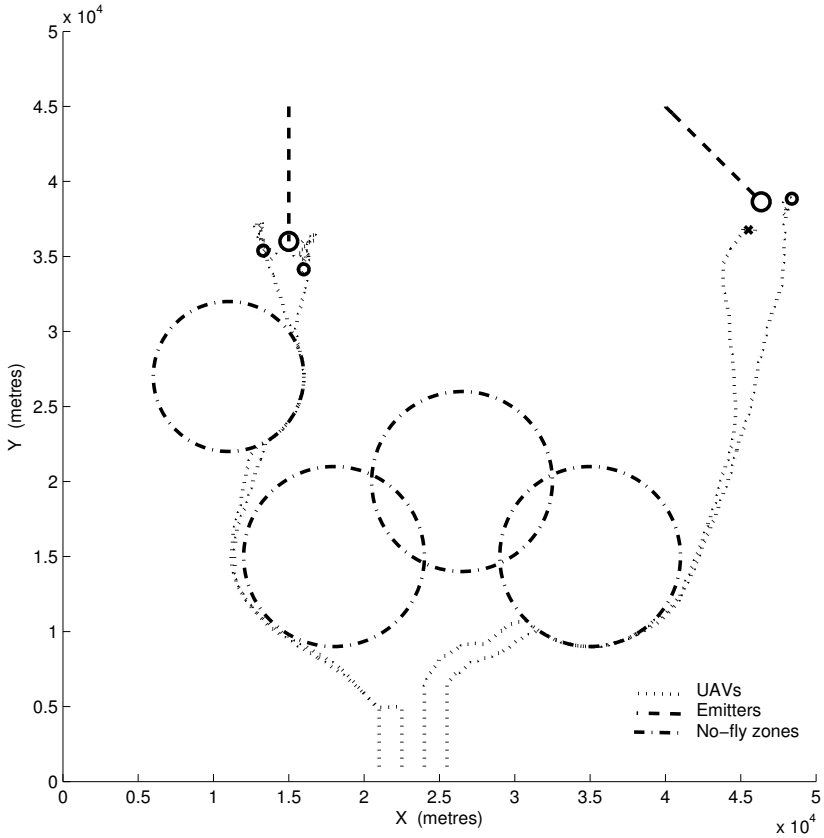

Figure 5.3: Fixed-Wing UAV and Emitter Trajectories for Controller with Roll-out Path Planning

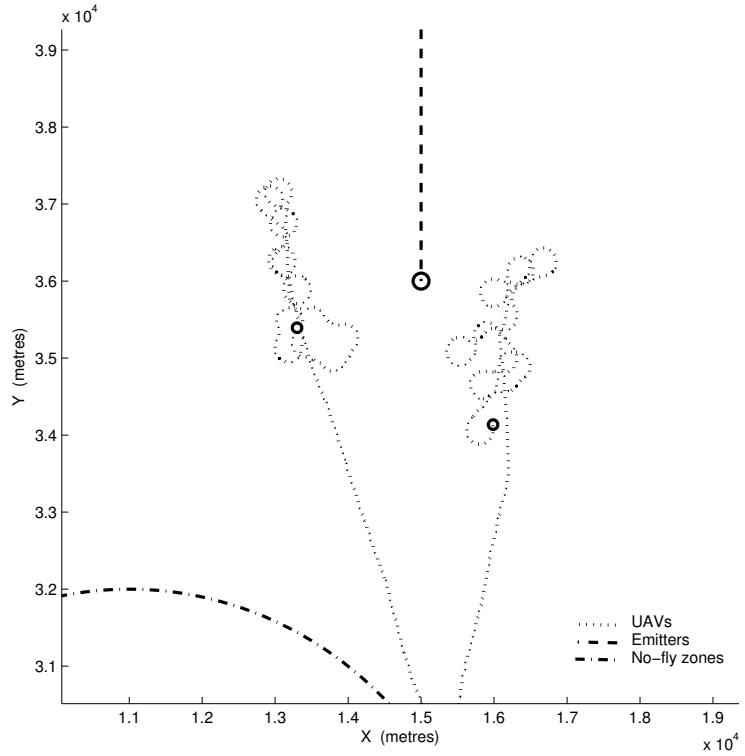

Figure 5.4: Enlarged View of Fixed-Wing UAV and Emitter Trajectories for Terminal Phase of Controller with Rollout Path Planning

geometry. Figure 5.5 gives a clearer indication of the actual UAV to target distances as well as the angles subtended by the UAVs 1 and 2 about target 1 during the entire terminal phase of the scenario. Note that both the distances and the angle subtended oscillate around the expected distance and angle. The oscillations are not unexpected given the inability of the UAVs to stay in a fixed position, the target is not stationary, and there is noise in the target position estimates. Nevertheless, there is no indication of instability and the resulting position estimates are good.

While the primary aim of the simulations was to demonstrate the control algorithms' ability to perform well when the fixed-wing aircraft dynamics are imposed, a quick look at measurement accuracy is worthwhile. Figures 5.6 and 5.7 show the emitter position estimation errors for the scenarios using the heuristic and rollout planning algorithms respectively. The figures demonstrate similar position accuracies for target 1, but for target 2 the rollout based algorithm gives significantly improved accuracy. This is due to the significantly more optimal trajectories of the UAVs tracking target 2 in the case of the rollout algorithm. The cost incurred to achieve this accuracy improvement is a 21% increase in computation time.

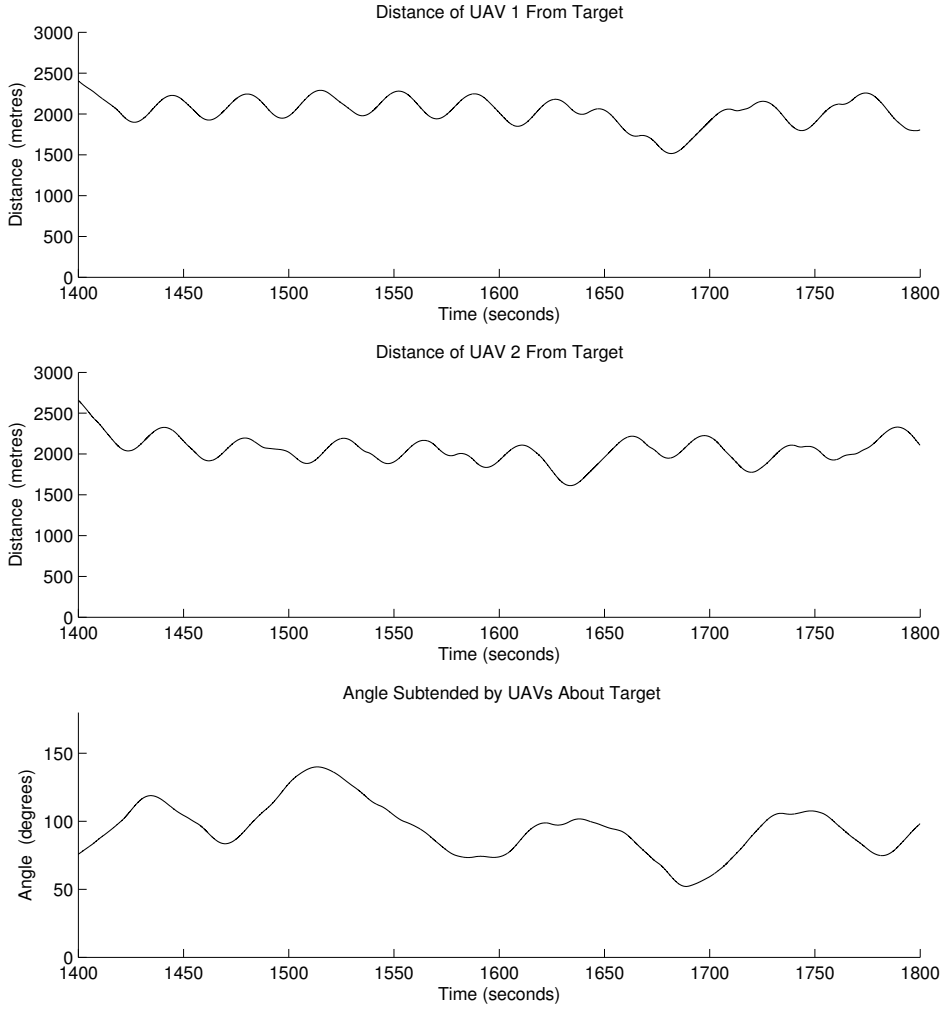

Figure 5.5: UAV to Target Distances and Angles Subtended about Target by UAVs 1 and 2

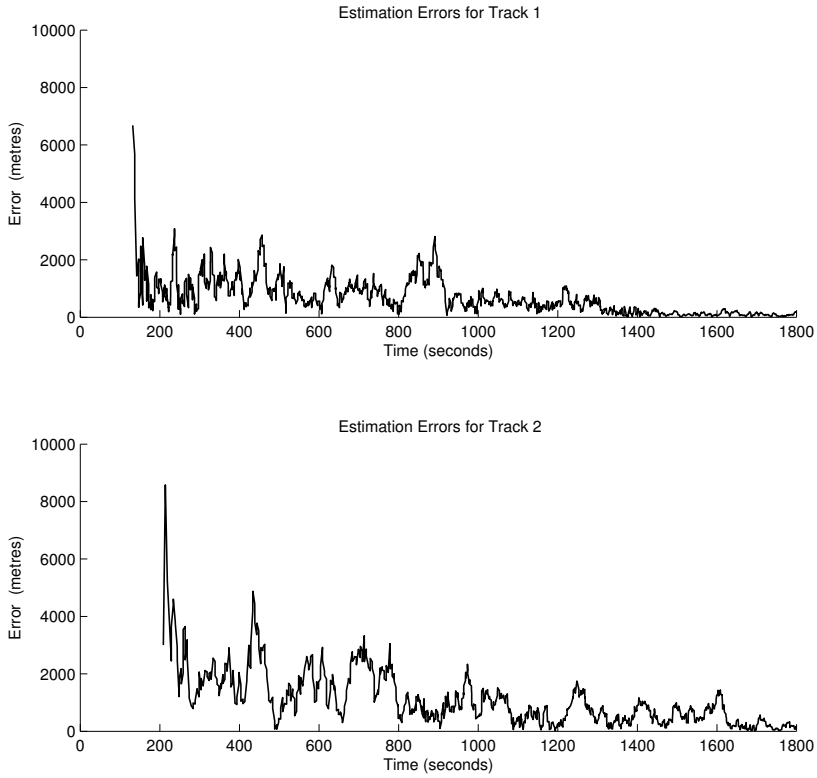

Figure 5.6: Emitter Position Estimation Errors for Fixed-Wing UAV Controller with Heuristic Path Planning

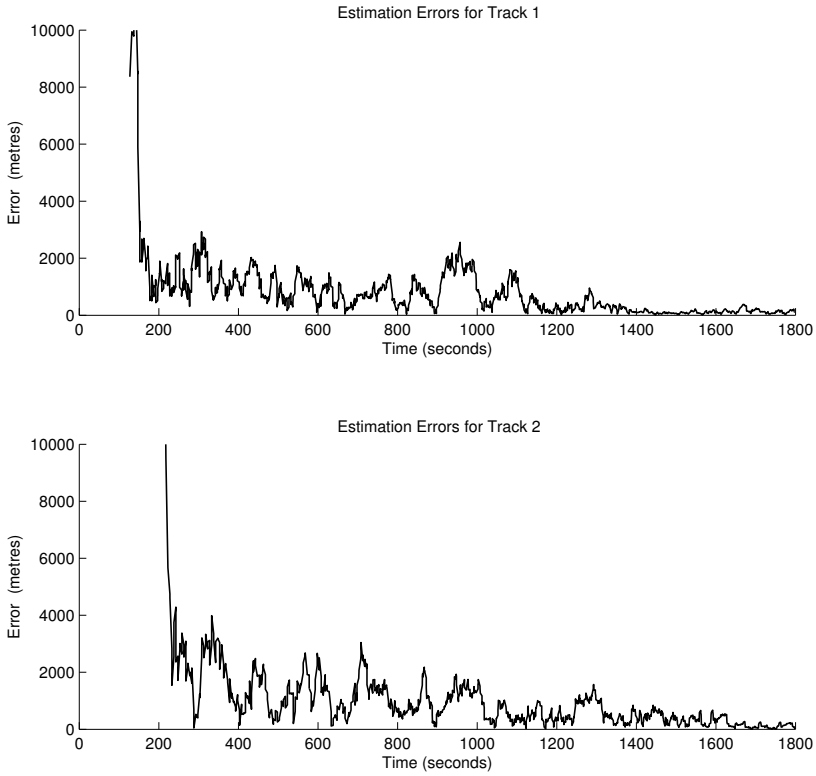

Figure 5.7: Emitter Position Estimation Errors for Fixed-Wing UAV Controller with Rollout Path Planning

Chapter 6

Stability Analysis

An important property that must be satisfied by any control system is *stability* [59]. Essentially, stability means that a small difference between the actual conditions of operation of the system and the conditions the control system is designed for must not completely alter the system behaviour. In this chapter, the controllers developed in Chapters 4 and 5 will be analysed for stability. It will become evident that the nature of these controllers makes proving stability a challenging task; however, some headway is made by concentrating on components of the system, developing mathematical proofs where possible, and aiding the demonstration of stability with graphical demonstration of convergence for regions in the state space for which a mathematical proof could not be found.

We will first begin with a summary of stability analysis techniques in the following section. The focus will be on Lyapunov's direct method as this was found to be most helpful for this problem and was, in fact, the method that was used. We will then give an outline of our stability proof approach for the controllers developed in Chapters 4 and 5. This will include some intuitive arguments regarding stability of the overall system as well as a discussion regarding what further work would need to be done to complete a full proof. Finally,

we will look in detail at what can be said about stability of the low-level control component of the model predictive controllers presented in Chapters 4 and 5. The controller described in Chapter 4 will be analysed first and then the controller described in Chapter 5 will be considered.

6.1 Background Theory

In this section we will define what stability means in some detail and then go on to describe the technique that we will later use to begin to produce some stability proofs for the system we are concerned with. The texts [59] and [57] cover the theory of stability of dynamical systems in detail. The following subsections will give only a short summary of the theory, concentrating on that which will be used in the proofs later in this chapter. The summary will closely follow the notation and descriptions in [59]. We will first consider stability definitions and Lyapunov's direct method for proving the stability of continuous systems (as they were first developed for continuous time), and then how they are extended to discrete time systems, of which our problem is an example. It will be found that the definitions and techniques are almost identical for the two types of system.

6.1.1 State Vector Representation of Dynamical Systems

Consider a continuous dynamical system described by the following first-order vector differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t; \mathbf{u}(t))$$

where $\mathbf{x}(t)$ is the value of the n_x-dimensional state vector $\mathbf{x}$ at time t, and $\mathbf{u}(t)$ is the value of the n_u-dimensional (control) input vector $\mathbf{u}$ at time t.

The dynamical system may also have an output of the form

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), t; \mathbf{u}(t))$$

where $\mathbf{y}(t)$ is the value the n_y-dimensional output vector $\mathbf{y}$ at time t.

The solution to the above system, with the given initial conditions $\mathbf{x}_0$ at t_0, will be denoted by $\mathbf{x}(t; \mathbf{x}_0, t_0)$.

If the input $\mathbf{u}(t) \neq \mathbf{0}$ for some t, the system is called a *forced system*. The dynamic system is called *free* or *unforced* if there is no input, i.e., $\mathbf{u}(t) = \mathbf{0}$, $\forall t$; the governing equation then becomes

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

From a mathematical point of view, there is no distinction between a free system and a forced system with a given input function.

The system is called *stationary* if the vector function $\mathbf{f}$ does not depend explicitly on time. A system that is both free and stationary is called *autonomous*. This type of system is governed by the following equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$$

If for all t

$$\mathbf{f}(\mathbf{x}_e, t) = \mathbf{0}$$

for some $\mathbf{x}_e$, then

$$\mathbf{x}(t; \mathbf{x}_e, t_0) = \mathbf{x}_e$$

for any t_0; this means that any solution that passes through $\mathbf{x}_e$ will remain there for all future time. The solution $\mathbf{x}_e$ is called the *equilibrium solution*, and, if $\mathbf{x}_e = \mathbf{0}$, the *null solution*, and the state $\mathbf{x}_e$ is referred to as the *equilibrium state*.

Before going further, a short note on some notation that will be used is required; the *norm* of a vector $\mathbf{x}$ will be denoted by $\|\mathbf{x}\|$. It will, in fact, be the

Euclidean norm, i.e.,

$$\|\mathbf{x}\| = \left(\sum_{i=1}^{n_x} x_i^2 \right)^{\frac{1}{2}}$$

6.1.2 Definition of Stability

The definition of stability that is of interest to us deals with the stability of an equilibrium with respect to some initial conditions. This definition was first discussed rigorously by the Russian mathematician Lyapunov and is hence referred to as *stability in the sense of Lyapunov*. This definition will now be summarized below.

Let $\mathbf{x}_e$ be an equilibrium state of the free dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

with

$$\mathbf{f}(\mathbf{x}_e, t) = \mathbf{0}$$

for all t.

The equilibrium state $\mathbf{x}_e$, or the equilibrium solution $x(t) = \mathbf{x}_e$, is called *stable* if for any given t_0 and positive ϵ, there exists a positive $\delta(\epsilon, t_0)$ such that

$$\|\mathbf{x}_0 - \mathbf{x}_e\| < \delta$$

implies

$$\|\mathbf{x}(t; \mathbf{x}_0, t_0) - \mathbf{x}_e\| < \epsilon$$

for all $t \geq t_0$.

The equilibrium state $\mathbf{x}_e$ is called *convergent* (or *attractive*) if for any t_0 there exists a $\delta_1(t_0)$ such that

$$\|\mathbf{x}_0 - \mathbf{x}_e\| < \delta_1$$

implies

$$\lim_{t\to\infty} \mathbf{x}\left(t; \mathbf{x}_0, t_0\right) = \mathbf{x}_e$$

for all t_0.

The equilibrium state $\mathbf{x}_e$ is called *asymptotically stable* if it is convergent and stable.

In words, the concepts of stability and asymptotic stability are as follows. If, when a system is perturbed slightly from its equilibrium state, all subsequent motions remain in a correspondingly small neighbourhood of the equilibrium state, then it is *stable*. If in addition, all subsequent motions return to the equilibrium, then it is *asymptotically stable*.

Note that the properties of stability and asymptotic stability are local; they state that there exists some region in state space surrounding the equilibrium state such that all trajectories starting from within that region are stable or asymptotically stable, but they say nothing about how large that region is. In the case of asymptotic stability the region is called the *domain of attraction*. In practice it is important to know how large the region is, and it is desirable (although not always necessary) for that region to be the entire state space. This leads us to the following definition.

The equilibrium state $\mathbf{x}_e$ is called *globally asymptotically stable* if it is stable and if *every motion, from anywhere within the state space*, converges to the equilibrium state as $t \to \infty$.

6.1.3 The Direct Method of Lyapunov

A general technique for proving the stability of a dynamical system is the *direct method of Lyapunov* (also referred to as *Lyapunov's second method)*. Before describing the method some definitions are required as follows.

Let $V(\mathbf{x})$ be a real scalar function of the vector $\mathbf{x}$, and let S be a closed bounded region in the $\mathbf{x}$ space, containing the origin, then we can define the following.

The function $V(\mathbf{x})$ is *positive semi-definite* in S if, for all $\mathbf{x}$ in S,

(i) $V(\mathbf{x})$ has continuous partial derivatives with respect to the components of the vector $\mathbf{x}$,

(ii) $V(\mathbf{0}) = 0$,

(iii) $V(\mathbf{x}) \geq 0$.

The function $V(\mathbf{x})$ is *positive definite* in S if, for all $\mathbf{x}$ in S,

(i) $V(\mathbf{x})$ has continuous partial derivatives with respect to the components of the vector $\mathbf{x}$,

(ii) $V(\mathbf{0}) = 0$,

(iii) $V(\mathbf{x}) > 0$.

The control system that has been developed in this thesis is stationary, and although there are control inputs $\mathbf{u}$, they are known, i.e., they are a function of the current state of the system, hence the system can be treated mathematically as a free system. Being both stationary and free thus makes it an *autonomous* system. Hence the Lyapunov stability theorems that will be presented in the sequel will be restricted to autonomous systems. Also, the stability theorems that will be presented concern stability of a system at the origin. This does not, however, represent any loss of generality since the problem of the stability of any equilibrium state can be reduced to the problem of the stability of the null solution of another dynamical system, via a suitable transformation.

Let us now consider the stability properties of the null solution of the autonomous system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{6.1}$$

with $\mathbf{f}(\mathbf{0}) = \mathbf{0}$. The following two theorems constitute Lyapunov's direct method for autonomous systems.

Theorem 6.1 *The null solution, or the equilibrium state at the origin of the system represented by Equation 6.1, is* ***stable*** *if there is some neighbourhood of the origin where a positive definite function* $V(\mathbf{x}(t))$ *exists such that its derivative* $\dot{V}(\mathbf{x}(t))$ *with respect to the solutions of Equation 6.1 is* ***negative semi-definite*** *in that region.*

Theorem 6.2 *The null solution, or the equilibrium state at the origin of the system represented by Equation 6.1, is* ***asymptotically stable*** *if there is some neighbourhood of the origin where a positive definite function* $V(\mathbf{x}(t))$ *exists such that its derivative* $\dot{V}(\mathbf{x}(t))$ *with respect to the solutions of Equation 6.1 is* ***negative definite*** *in that region.*

As was mentioned earlier, it is important to be able to estimate the region about the equilibrium point where all motions initiating in that region converge to that equilibrium. The following theorem provides us with that information.

Theorem 6.3 *Let* $V(\mathbf{x}(t))$ *be a scalar function. Suppose that the region* $R = \{\mathbf{x}(t) \,|V(\mathbf{x}(t)) < a\}$ *is bounded. Let* $\dot{V}(\mathbf{x}(t))$ *be the derivative of* $V(\mathbf{x}(t))$ *along the solutions of Equation 6.1. If* $V(\mathbf{x}(t))$ *is positive definite and* $\dot{V}(\mathbf{x}(t))$ *negative definite in* R*, then the origin is an asymptotically stable equilibrium state and all the motions starting in* R *converge to the origin as* $t \to \infty$.

6.1.4 Discrete Time Systems

Let us now consider how the theory for continuous time systems presented in the previous sections translates to discrete time systems.

The dynamic equations of a discrete time system can be written in the following general form

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), k, \mathbf{u}(k))$$

where $\mathbf{x}(k)$ is the value of the n_x-dimensional state vector $\mathbf{x}$ at time $t = t_k$, $\mathbf{u}(k)$ is the value of the n_u-dimensional (control) input vector $\mathbf{u}$ at time $t = t_k$, and k is a discrete time index which only takes on integer values.

The dynamical system may also have an output of the form

$$\mathbf{y}(k) = \mathbf{g}(\mathbf{x}(k), k; \mathbf{u}(k))$$

where $\mathbf{y}(k)$ is the value the n_y-dimensional output vector $\mathbf{y}$ at time $t = t_k$.

This system is *free* if there is no control input; the resulting dynamic equation then becomes

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), k)$$

It is *stationary* if the vector function $\mathbf{f}$ does not depend explicitly on the time index k; the dynamic equation then takes the form

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$$

It is *autonomous* if it is both free and stationary, resulting in the following dynamic equation

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k))$$

Now let us consider how the stability definitions apply to discrete time systems. Consider now the free discrete time system described by the following

difference equation

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), k)$$

The state $\mathbf{x}_e$ is an *equilibrium state* of the system if for all k

$$\mathbf{x}_e = \mathbf{f}(\mathbf{x}_e, k)$$

and a solution starting at state $\mathbf{x}_e$ at time t_{k_0} will remain in that state for all time, i.e.,

$$\mathbf{x}(k; \mathbf{x}_e, k_0) = \mathbf{x}_e$$

The stability definitions which were presented in Section 6.1.2 for continuous time systems can be transferred almost verbatim to discrete time systems, with the only change being the replacement of time t with the discrete time index k, and the initial time t_0 with time index corresponding to the initial time, i.e., k_0.

Lyapunov's stability theorems for continuous time systems, as presented in Section 6.1.3, are also almost identical in the case of discrete time systems, with the required modifications being as follows. Firstly, as for the stability definitions, time t is replaced with the discrete time index k. The other modification is that the time derivative $\dot{V}(\mathbf{x}(t))$ of the Lyapunov function is replaced by its *difference function* along the solutions of the system equation, i.e.,

$$\Delta V(\mathbf{x}(k)) = V(\mathbf{x}(k+1)) - V(\mathbf{x}(k))$$

Although very similar to the continuous case, the discrete versions of the theorems presented in Section 6.1.3 will now be presented below, essentially because of their importance for the proofs in the subsequent sections of this chapter.

Consider the stability properties of the null solution of the autonomous system

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) \tag{6.2}$$

with $\mathbf{f}(\mathbf{0}) = \mathbf{0}$. The following theorems constitute Lyapunov's direct method for *discrete time* autonomous systems.

Theorem 6.4 *The null solution, or the equilibrium state at the origin of the system represented by Equation 6.2, is* ***stable*** *if there is some neighbourhood of the origin where a positive definite function* $V(\mathbf{x}(k))$ *exists such that its difference function* $\Delta V(\mathbf{x}(k))$ *with respect to the solutions of Equation 6.2 is* ***negative semi-definite*** *in that region.*

Theorem 6.5 *The null solution, or the equilibrium state at the origin of the system represented by Equation 6.2, is* ***asymptotically stable*** *if there is some neighbourhood of the origin where a positive definite function* $V(\mathbf{x}(k))$ *exists such that its difference function* $\Delta V(\mathbf{x}(k))$ *with respect to the solutions of Equation 6.2 is* ***negative definite*** *in that region.*

Theorem 6.6 *Let* $V(\mathbf{x}(k))$ *be a scalar function. Suppose that the region* $R = \{\mathbf{x}(k) | V(\mathbf{x}(k)) < a\}$ *is bounded. Let* $\Delta V(\mathbf{x}(k))$ *be the difference function of* $V(\mathbf{x}(k))$ *along the solutions of Equation 6.2. If* $V(\mathbf{x}(k))$ *is positive definite and* $\Delta V(\mathbf{x}(k))$ *negative definite in* R*, then the origin is an asymptotically stable equilibrium state and all the motions starting in* R *converge to the origin as* $k \to \infty$*.*

6.1.5 Stability of Model Predictive Optimal Control

Proving that a model predictive control optimization scheme leads to a stable closed loop control system is a non-trivial task. In general model predictive controllers are implemented with a receding horizon strategy, so all that can be said is that the optimization up to the (current) horizon is in some sense optimal. As it turns out, however, if the cost function is positive definite it can be used

as a Lyapunov function. This approach was first suggested in the early 1980's and has over time become almost universally employed as a natural Lyapunov function for stability analysis of model predictive control [42, Sect. 2.4], [18, Sect. 4.4]. However, there is still another difficulty that must be dealt with, in that the optimization problem that is solved is defined only up to a finite optimization horizon, so that anything that can be said about stability within the finite horizon does not necessarily say anything about the overall stability over a period of time that is greater than the optimization horizon. A trick that is often used to resolve this is to add appropriate cost function weighting and restrictions on the terminal state of the optimization horizon to account for the impact of events that lie beyond the end of the optimization horizon [18, Sect. 4.4], [42]. Hence, one must steer the system into this restricted terminal region over the finite time available in the optimization window, in order to guarantee stability. Three early and important papers on the use of these techniques, i.e., use of the cost function as a Lyapunov function and imposition of terminal constraints to guarantee stability, are [9], [29] and [41].

6.2 Outline of Stability Proof Approach

The solution that we use for our control problem is a somewhat unconventional application of model predictive control and as a result our stability proofs will need to be modified somewhat. Importantly, the cost function can still be used as a Lyapunov function; however, the use of terminal constraints to guarantee stability, in the sense described above, is not possible. What has been achieved with regard to proof of stability will now be described.

As described in earlier chapters, because of the combinatorial aspects of the multisensor multitarget problem that we are dealing with, the solution that has

been developed uses a hierarchical form of MPC. As currently implemented, a planner computes all UAV assignments and nominal paths at fixed intervals of time that are substantially longer than the controller update interval. At each of these planner updates the nominal UAV paths are computed using the current best estimate of the target states and the known UAV states. The computations are open loop, leaving no possibility of instability. The only issues are whether the planner will always find a feasible path for each UAV and how optimal the assignments and paths will be. While not stability issues, let us consider these issues now.

Firstly, the assignment algorithm is a direct one, considering all possible assignments and selecting the lowest cost one. Assuming the path computations are performed successfully, there is no possibility of the assignment algorithm failing, and it is guaranteed to give an optimal assignment if the computed paths for each possible assignment are optimal.

With regard to the planner's path computations, let us first consider the heuristic-based trajectory calculation algorithm presented in Section 4.4.2. With not very restrictive constraints on the no-fly-zone number, shapes and spacings, this algorithm will always find a (suboptimal) path from the current UAV position to the current estimate of the assigned target, as long as a one exists. The second trajectory calculation algorithm, i.e., the algorithm described in Section 4.4.3, uses a rollout policy which in turn uses the heuristic algorithm as its base policy. This algorithm produces paths that are at least as close to optimal as those of the base policy [5, Prop. 6.3.1 (p. 293), and Sect. 6.4 (p. 316)] and often considerably closer to optimal. With the same non-restrictive constraints as the heuristic algorithm, this algorithm will also always find a path if one exists.

Now let us consider what occurs across multiple planner updates. At each

update of the planner, the estimates of the target states are different, firstly because targets move, but more importantly because there is substantial random measurement error and consequently target state estimation error. The issue here is how will the control algorithm deal with these random variations in target state estimates. This in fact is an issue of robustness, i.e., the maintenance of properties such as stability and performance in the presence of uncertainty [42, Sect. 4]. We will not deal with the issue of robustness rigorously, leaving it as future work to be done; however, with regard to robustness to variations in target state estimates across multiple planner updates some intuitive arguments can be made that are reasonably convincing. Firstly, if the one-step-ahead controller is stable between planner updates, then the UAVs will on average move to positions where target state measurements become more accurate during successive planner updates. The target state estimates will improve even more than the measurements, because of the cumulative effect of the multiple measurements made during the planner update period. Hence during successive updates of the planner, the random variations in the target state estimates will on average decrease, resulting in a self correcting effect. Secondly, the changes in the UAV position from one update of the planner to the next will usually not be very large relative to target distances, so even if the random measurement error is such that the UAVs move toward poorer positions for the occasional planner update period, the degradation in UAV geometry will only be small and should almost always be more than compensated for by the fact that many target state measurements are made during a planner update period thus still leading to improved target state estimates for use in the subsequent planner update. Hence, there will almost always still be a self correcting effect. This has in fact been corroborated by the simulations that were presented in earlier chapters, in that there was no sign of erratic behaviour of the controller.

Given the above, what is left to prove is, given the planner trajectories of any selected update of the planner, will the one-step-ahead controller, using the cost functions associated with the nominal planner paths and estimation accuracy based cost be able to compute and control trajectories of the UAVs without instability occurring. The one-step-ahead controller (locally) optimizes over a very short horizon, $N_H = 1$. The short horizon is a consequence of the combinatorial aspects of taking into account the interaction of the multiple UAV positions, which make longer optimization horizons computationally infeasible. Because of this short horizon there is no way of implementing terminal constraints to guarantee stability. What can be attempted, however, is to use the one-step-ahead controller's cost function as a Lyapunov function and try to prove stability by demonstrating that the solutions (i.e., controls) of the one-step-ahead controller, over multiple updates, converge to the desired equilibrium point. To do this mathematically would be extremely difficult, if not impossible because of the implicit nature of the control law and the dependence of the cost function on the interaction between UAVs. In addition to these difficulties there is also the uncertainty (randomness) in the target state estimates, which leads to the issue of robustness, as was discussed above for the planner updates. Hence no attempt will be made to give a full proof, leaving it as possible future work. However, a start will be made, concentrating on a component of the one-step-ahead controller's cost function to begin to get an indication of the stability performance of the controller. This will allow some mathematical proofs to be produced, which, with the aid of some graphical demonstrations will begin to add to our confidence in the stability behaviour of the controller. Referring to the definitions C_P and C_E in Section 4.5, the component of the cost function that we will consider is the cost C_P associated with the divergence of the UAV trajectory from the path calculated by the path planner; this is

somewhat easier to deal with than the cost C_E associated with target position estimation error. The primary reasons C_P is somewhat easier to deal with are that it doesn't depend on the interaction between the UAV's relative positions, and it is not affected (during a single planner update interval) by the random variations of the target state estimates, i.e., it is deterministic.

In the following section we will analyse the one-step-ahead controller assuming the idealized dynamics model for the UAVs that was presented in Chapter 4. Considering only the cost function C_P, we will provide a mathematical proof of global asymptotic stability. In the subsequent section we will then consider the one-step-ahead controller, again assuming only the cost function C_P, but this time using the fixed-wing UAV dynamics model that was presented in Chapter 5. We will mathematically prove this controller to be, at worst, locally stable, and then go on to provide graphical demonstrations of asymptotically stable behaviour given a range of UAV starting positions, thus providing strong evidence of stability at points that are a considerable distance from the equilibrium point.

6.3 One-Step-Ahead Controller Stability Analysis - Idealized Dynamics

Let us first consider the one-step-ahead controller described in Chapter 4 as it has simpler dynamics than the controller described in Chapter 5. In this controller, an idealized model of UAVs is used in which control is implemented by changing the direction of travel of the UAVs, while they travel at a constant pre-set speed, or by setting the speed to zero. At each step the control that is selected is the one that results in the minimum value of a cost function. The

changes in direction of travel, or between the preset speed and zero speed, are assumed to occur instantaneously.

As mentioned in the previous section, to analyse stability for this controller, Lyapunov's direct method will be used, and the cost function will be used as the Lyapunov function. As previously defined in Section 4.5, $C_P = C_d + C_v$, where (i) C_d is the cost associated with the distance d of the UAV from the reference path calculated by the path planner for that UAV and (ii) C_v is the cost associated with v_{diff}, the magnitude of the difference between the component of the UAV velocity vector in the direction of the (directed) line corresponding to the current leg of the path calculated by the planner, and the speed of the UAV. Let us now look at the components of the cost function C_P associated with the cost C_d of the distance d and the cost C_v of v_{diff}, as was derived in Section 4.5. The key equations presented in Section 4.5 are reproduced below with the subscripts representing the control option and UAV subscripts removed for convenience.

The current leg of the reference path for a particular UAV is represented by the line segment joined by (x_1, y_1) and (x_2, y_2) and the current UAV position is (x, y). Also $\mathbf{a} = (x_2, y_2) - (x_1, y_1)$, $\mathbf{b} = (x, y) - (x_1, y_1)$ and θ is the angle between $\mathbf{a}$ and $\mathbf{b}$. It can easily be shown that the distance d is

$$d = \frac{|(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{6.3}$$

The cost of using a particular control option for a particular UAV is then defined as $C_d = k_d d^2$ where k_d is a positive constant.

Now let us revisit C_v. The vector $\mathbf{v}$ is the velocity of a particular UAV and $\mathbf{w}$ is the orthogonal projection of $\mathbf{v}$ on the vector $\mathbf{a}$. The vector $\mathbf{z}$ is a vector with magnitude v_M, i.e., the maximum possible speed of the UAV, but with the same direction as $\mathbf{a}$. Also note that the components of $\mathbf{v}$ are $(\dot{x}, \dot{y})$, and note

that $|\mathbf{z}| = v_M$, then

$$v_{diff} = v_M - \frac{[\dot{x}(x_2 - x_1) + \dot{y}(y_2 - y_1)]}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{6.4}$$

The cost of using a particular control option for a particular UAV is then defined as $C_v = k_v v_{diff}^2$, where k_v is a positive constant. The total cost associated with d and v_{diff} is then

$$C_P = C_d + C_v \tag{6.5}$$

Without loss of generality, let $y_1 = y_2 = 0$, i.e., the line segment $(x_2, y_2) - (x_1, y_1)$ is assumed to be on the x-axis. Substituting $y_1 = y_2 = 0$ into Equation 6.3 results in $d = |y|$ and hence $C_d = k_d y^2$. Similarly, substituting $y_1 = y_2 = 0$ into Equation 6.4 results in $v_{diff} = v_M - \dot{x}$ and hence $C_v = k_v (v_M - \dot{x})^2$. Substituting the equations for C_d and C_v into Equation 6.5 gives

$$C_P = k_d y^2 + k_v (v_M - \dot{x})^2 \tag{6.6}$$

Now let us represent the UAV's velocity vector $\mathbf{v}$ in polar form, i.e., $\mathbf{v} = v\angle\theta$, where v is the UAV's speed and θ is the UAV's heading. Recall that the speed of the UAVs is constant, and equal to v_M, for all but one control option in the controller described in Chapter 4; this particular control option was employed primarily to enable stopping of the UAVs once they reached their desired terminal states (note that *all* the control options in the controller described in Chapter 5 assume constant speed). So, assuming constant speed v_M, the heading θ and distance y fully define the aspects of the UAV's state that determines the cost C_P. Rewriting Equation 6.6 in terms of θ and y gives

$$\begin{aligned} C_P(\theta, y) &= k_d y^2 + k_v (v_M - v_M \cos\theta)^2 \\ &= k_d y^2 + k_v v_M^2 (1 - \cos\theta)^2 \end{aligned} \tag{6.7}$$

From Equation 6.7 we see that

$$C_P(\theta, y) > 0 \qquad \forall \theta, y > 0$$

and

$$C_P(\theta, y) = 0 \quad \text{when } \theta = 0, y = 0$$

hence $C_P(\theta, y)$ is positive definite, as required by Lyapunov's Theorems in Section 6.1.4.

At this point it pays to show Equation 6.7 graphically; Figure 6.1 shows the cost C_P as a function of the heading θ and distance y. The ranges of the heading θ and distance y in this plot were $-\pi \leq \theta \leq \pi$ radians and $-3600 \leq y \leq 3600$ metres respectively, and the values of k_d, k_v and v_M were $k_d = 0.001$, $k_v = 3.6$ and $v_M = 30$ m/s.

With regard to the stability proof, the first thing to be noted is that since at any individual step there is always the option of remaining stationary, the UAV states will never diverge to higher cost states than the one that the UAV is currently in. Hence, on this basis alone, the system is Lyapunov stable (but not necessarily asymptotically stable). Asymptotic stability will now be proved in the following paragraphs.

First let us determine the state update equations of the controller described in Chapter 4. It is mathematically convenient to consider the change in the direction of travel of a UAV at time t_k to be the result of an impulse $\Omega_k \delta_k$ where δ_k is a unit impulse starting at time t_k and ending at time $t_{k+\Delta}$ with $\Delta \to 0$. This impulse is thus considered to be the control at time t_k. The change in direction of travel that results thus takes the form of a step function with step-size $\Delta\theta_k = \Omega_k$, and we can arrive at any $\theta_{k+1} = \theta_k + \Omega_k$ by making the appropriate choice for Ω_k.

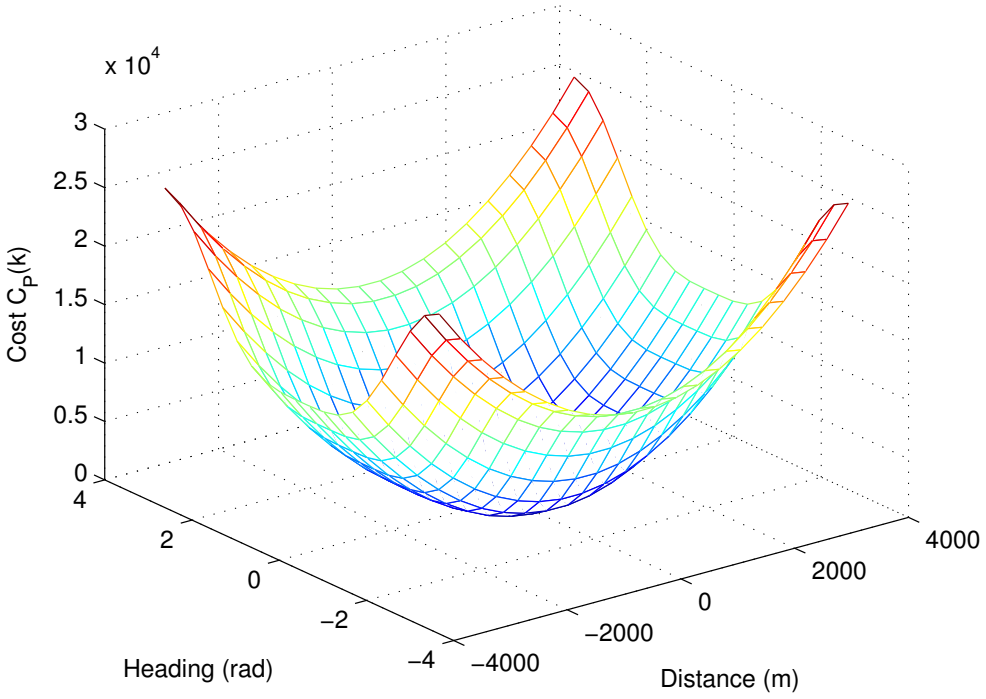

Figure 6.1: Cost C_P as a Function of the UAV's Heading θ and Distance y.

The transition equations for each component of the state vector are then

$$\begin{aligned} x_{k+1} &= x_k + vT\cos\theta_{k+1} \\ \dot{x}_{k+1} &= v\cos\theta_{k+1} \\ y_{k+1} &= y_k + vT\sin\theta_{k+1} \\ \dot{y}_{k+1} &= v\sin\theta_{k+1} \end{aligned} \tag{6.8}$$

where θ_{k+1} is the heading during the time period $t_k < t \leq t_{k+1}$, and $T = t_{k+1} - t_k$. Note also that

$$\begin{aligned} \dot{x}_k &= v\cos\theta_k \\ \dot{y}_k &= v\sin\theta_k \end{aligned} \tag{6.9}$$

where θ_k is the heading during the time period $t_{k-1} < t \leq t_k$.

Now, let Equation 6.7 be the Lyapunov function (or equivalently, Equation 6.6) and note that we have already shown that it is positive definite. Hence, from Theorem 6.6, if $C_P(k+1) - C_P(k)$ is negative definite and the only cost were C_P, then the controller is *globally asymptotically stable*. In other words, if it can be proven that at every step of the controller, irrespective of the starting point, the value of the Lyapunov function decreases, i.e., $C_P(k+1) - C_P(k) < 0$ if $(\theta_k, y_k) \neq (0,0)$ and remains constant, i.e., $C_P(k+1) - C_P(k) = 0$ if $(\theta_k, y_k) \neq (0,0)$, then if the only cost were C_P, the controller is globally asymptotically stable. This proof will now be given in the following paragraphs.

Before beginning the proof, for the sake of convenience (compactness and consistency with the Lyapunov theorems in the previous section), let us now define the following nomenclature

$$V_k \triangleq C_P(k) \tag{6.10}$$

and

$$\Delta V_k \triangleq C_P\left(k+1\right) - C_P\left(k\right) \tag{6.11}$$

The above two substitutions will be used in what follows.

With reference to Equations 6.6 and 6.11, the change in the cost function C_P i.e., ΔV_k, over two consecutive updates is

$$\Delta V_k = k_d y_{k+1}^2 + k_v\left(v_M - \dot{x}_{k+1}\right)^2 - k_d y_k^2 - k_v\left(v_M - \dot{x}_k\right)^2 \tag{6.12}$$

Rearranging and substituting Equations 6.8 and 6.9 into Equation 6.12 and considering only the constant pre-set speed (i.e., $v = v_M$) cases, gives

$$\begin{aligned}\Delta V_k &= k_d\left[\left(y_k + v_M T\sin\theta_{k+1}\right)^2 - y_k^2\right] \\ &\quad + k_v\left[\left(v_M - v_M\cos\theta_{k+1}\right)^2 - \left(v_M - v_M\cos\theta_k\right)^2\right]\end{aligned}$$

Expanding, cancelling terms and rearranging the above equation gives

$$\begin{aligned}\Delta V_k &= k_d\left(v_M^2 T^2\sin^2\theta_{k+1} + 2y_k v_M T\sin\theta_{k+1}\right) \\ &\quad + k_v v_M^2\left[\left(\cos^2\theta_{k+1} - \cos^2\theta_k\right) - 2\left(\cos\theta_{k+1} - \cos\theta_k\right)\right]\end{aligned} \tag{6.13}$$

The most direct way to prove $\Delta V_k < 0$ is to take the partial derivative of Equation 6.13 with respect to θ_{k+1}, equate it to zero to find all the turning points of ΔV_k, calculate the value of $\left[\Delta V_k\right]_j$ at these turning points as well as at the maximum and minimum values of θ_{k+1} (i.e., $\theta_{k+1} = \pm\pi$), and then find the minimum of these, $\min\left[\Delta V_k\right]_j$. Note that $\min\left[\Delta V_k\right]_j$ is chosen because the controller always chooses θ_{k+1} so as to minimize ΔV_k. If the minimum value of ΔV_k can be proven to always be less than zero (except at the equilibrium point where it's equal to zero) then the controller is asymptotically stable in the Lyapunov sense.

Let us now take the partial derivative with respect to θ_{k+1} of Equation 6.13 to see if the above approach is feasible. Taking the derivative and then

simplifying gives

$$\begin{aligned}\frac{\partial \Delta V_k}{\partial \theta_{k+1}} &= 2k_d v_M T \left(v_M T \sin\theta_{k+1}\cos\theta_{k+1} + y_k \cos\theta_{k+1}\right) \\ &\quad +2k_v v_M^2 \left(\sin\theta_{k+1} - \sin\theta_{k+1}\cos\theta_{k+1}\right)\end{aligned} \tag{6.14}$$

The turning points and inflections of Equation 6.14 occur at

$$2k_d v_M T \left(v_M T \sin\theta_{k+1} + y_k\right)\cos\theta_{k+1} + 2k_v v_M^2 \left(1 - \cos\theta_{k+1}\right)\sin\theta_{k+1} = 0$$

We could not find a closed form solution for θ_{k+1} (which may, in fact, be impossible to find), hence another approach is required. Note that we need to prove that a value of θ_{k+1} can always be found that results in $\Delta V_k < 0$ (or $\Delta V_k = 0$ at the equilibrium point); to do this we do not necessarily need to find the optimal value of θ_{k+1}. We will thus consider all the possible UAV states at time t_k and prove that a value θ_{k+1} can always be found that gives $\Delta V_k < 0$ when $(\theta_k, y_k) \neq (0,0)$ and $\Delta V_k = 0$ when $(\theta_k, y_k) = (0,0)$. The optimal θ_{k+1} will always do as well or better, and will be found by the controller to within the resolution of the search that is performed. This proof follows.

Let us assume that we choose $\theta_{k+1} = 0$, then Equation 6.13 becomes

$$\begin{aligned}\Delta V_k &= k_v v_M^2 \left[\left(1 - \cos^2\theta_k\right) - 2\left(1 - \cos\theta_k\right)\right] \\ &= k_v v_M^2 \left[\left(1 - \cos\theta_k\right)\left(1 + \cos\theta_k\right) - 2\left(1 - \cos\theta_k\right)\right] \\ &= k_v v_M^2 \left[\left(1 - \cos\theta_k\right)\left(\cos\theta_k - 1\right)\right] \\ &= -k_v v_M^2 \left(1 - \cos\theta_k\right)^2\end{aligned}$$

Simple inspection shows that if $\theta_k \neq 0$, choosing $\theta_{k+1} = 0$ results in $\Delta V_k < 0$, i.e., a cost decrease at every update of the controller.

Now let us assume that $\theta_k = 0$, then the above equation becomes $\Delta V_k = 0$, i.e., choosing $\theta_{k+1} = 0$ results in the cost remaining the same. However as stated above $\theta_{k+1} = 0$ may not necessarily be the lowest cost control. To see if there is

a lower cost control, let us consider the partial derivative with respect to θ_{k+1} of Equation 6.13, i.e., Equation 6.14 and determine its value at $\theta_k = 0, \theta_{k+1} = 0$. The result is

$$\left.\frac{\partial \Delta V_k}{\partial \theta_{k+1}}\right|_{\substack{\theta_k=0,\\ \theta_{k+1}=0}} = 2k_d v_M T y_k$$

Now $k_d > 0$, $v_M > 0$, $T > 0$, hence if $y_k \neq 0$ then $\theta_{k+1} = 0$ is not a minimum, i.e., a lower cost control can be found, resulting in $\Delta V_k < 0$. If, however, $y_k = 0$, then $\theta_{k+1} = 0$ is a minimum resulting in $\Delta V_k = 0$, which is exactly as required since $(\theta_k, y_k) = (0, 0)$ corresponds to the equilibrium point.

Now let us consider the case of the UAV being stationary at time t_k (i.e., the zero speed control option) and determine if there is a θ_{k+1} for which $\Delta V_k < 0$. For this case, Equations 6.9 are replaced by

$$\begin{aligned} \dot{x}_k &= 0 \\ \dot{y}_k &= 0 \end{aligned} \tag{6.15}$$

Rearranging and substituting Equations 6.8 and 6.15 into Equation 6.12 gives

$$\Delta V_k = k_d \left[(y_k + v_M T \sin\theta_{k+1})^2 - y_k^2 \right] + k_v \left[(v_M - v_M \cos\theta_{k+1})^2 - v_M^2 \right]$$

Expanding, cancelling terms and rearranging the above equation gives

$$\Delta V_k = k_d \left(v_M^2 T^2 \sin^2\theta_{k+1} + 2 y_k v_M T \sin\theta_{k+1} \right) + k_v v_M^2 \left(\cos^2\theta_{k+1} - 2\cos\theta_{k+1} \right) \tag{6.16}$$

Choosing $\theta_{k+1} = 0$ results in

$$\begin{aligned} \Delta V_k &= -k_v v_M^2 \\ &< 0 \end{aligned}$$

Hence for this case there is always a control that leads to a decrease in cost. This now completes the proof that the above controller is *globally asymptotically stable*.

6.4 One-Step-Ahead Controller Stability Analysis - Fixed-Wing Aircraft Dynamics

Now let us consider the controller described in Chapter 5. In this controller, a somewhat simplified model of fixed-wing UAVs is used, in which each UAV is assumed to be travelling at a fixed speed but can turn in the horizontal plane over a range of turn rates corresponding with transverse accelerations up to $\pm a_{t_{\max}}$. During any period of time $t_k \leq t < t_{k+1}$ the magnitude of the transverse acceleration a_t is constant and its direction is toward a fixed point, the centre of the turn. The model used to represent this type of turn is referred to as a coordinated turn model [2, p. 187].

Referring to Equation 5.1 for the coordinated turn model, and repeating it here for convenience, we have the following state equation for describing the transition from the UAV state $\mathbf{x}(k)$ at time t_k to $\mathbf{x}(k+1)$ at time t_{k+1}

$$\mathbf{x}(k+1) = A(k)\,\mathbf{x}(k) \tag{6.17}$$

where

$$A(k) = \begin{bmatrix} 1 & \frac{\sin\omega_k T}{\omega_k} & 0 & -\frac{(1-\cos\omega_k T)}{\omega_k} \\ 0 & \cos\omega_k T & 0 & -\sin\omega_k T \\ 0 & \frac{(1-\cos\omega_k T)}{\omega_k} & 1 & \frac{\sin\omega_k T}{\omega_k} \\ 0 & \sin\omega_k T & 0 & \cos\omega_k T \end{bmatrix}$$

$$\mathbf{x}(k) = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k \end{bmatrix}^T$$

and

$$\omega_k = \frac{a_t(k)}{v}$$

and where T is the time interval $T = t_{k+1} - t_k$, $a_t(k)$ is the transverse (centripetal) acceleration at time $t_k \leq t < t_{k+1}$, v is the UAV's speed, and ω_k is the

angular velocity of the UAV at time $t_k \leq t < t_{k+1}$.

Rewriting the above state equation in another form gives

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \frac{\sin \omega_k T}{\omega_k} \dot{x}_k - \frac{(1-\cos \omega_k T)}{\omega_k} \dot{y}_k \\ \cos(\omega_k T)\, \dot{x}_k - \sin(\omega_k T)\, \dot{y}_k \\ \frac{(1-\cos \omega_k T)}{\omega_k} \dot{x}_k + y_k + \frac{\sin \omega_k T}{\omega_k} \dot{y}_k \\ \sin(\omega_k T)\, \dot{x}_k + \cos(\omega_k T)\, \dot{y}_k \end{bmatrix} \tag{6.18}$$

Also, the UAV's speed is fixed at $v = v_M$, resulting in

$$\begin{aligned} \dot{x}_k &= v_M \cos \theta_k \\ \dot{y}_k &= v_M \sin \theta_k \end{aligned} \tag{6.19}$$

where θ_k is the heading at time t_k.

As was done in Section 6.3, let Equation 6.7 (or equivalently, Equation 6.6) be the Lyapunov function. Again, as in Section 6.3, from Theorem 6.6, if $C_P(k+1) - C_P(k)$ is negative definite and the only cost were C_P, then the controller is globally asymptotically stable. Hence, again we have to prove that at every step of the controller, $C_P(k+1) - C_P(k) < 0$ if $(\theta_k, y_k) \neq (0,0)$ and $C_P(k+1) - C_P(k) = 0$ if $(\theta_k, y_k) = (0,0)$, in order to prove global asymptotic stability. Let us investigate the possibility of proving this now. As in Section 6.3, we will use the substitutions shown in Equations 6.10, and 6.11, i.e., $V_k \triangleq C_P(k)$ and $\Delta V_k \triangleq C_P(k+1) - C_P(k)$, respectively.

Using the above transition equation, i.e., Equation 6.18, the substitution as per Equation 6.10 and referring to Equation 6.6, we derive the following for the cost (Lyapunov) function at time t_{k+1}

$$\begin{aligned} V_{k+1} &= k_d \left[\frac{(1-\cos \omega_k T)}{\omega_k} \dot{x}_k + y_k + \frac{\sin \omega_k T}{\omega_k} \dot{y}_k \right]^2 \\ &\quad + k_v \left[v_M - (\cos(\omega_k T)\, \dot{x}_k - \sin(\omega_k T)\, \dot{y}_k) \right]^2 \end{aligned} \tag{6.20}$$

Substituting Equation 6.19 into 6.20 gives

$$\begin{aligned} V_{k+1} &= k_d \left[\frac{(1-\cos\omega_k T)}{\omega_k} v_M \cos\theta_k + y_k + \frac{\sin\omega_k T}{\omega_k} v_M \sin\theta_k \right]^2 \\ &\quad + k_v \left[v_M - \left(\cos\left(\omega_k T\right) v_M \cos\theta_k - \sin\left(\omega_k T\right) v_M \sin\theta_k \right) \right]^2 \end{aligned} \tag{6.21}$$

The most direct way to prove $\Delta V_k < 0$ is to take the partial derivative of Equation 6.21 with respect to ω_k, equate it to zero to find all the turning points of V_{k+1} (noting that these are also the turning points of ΔV_k), calculate the value of $[\Delta V_k]_j$ at these turning points as well as at the maximum and minimum values of ω_k, and then find the minimum of these, $\min [\Delta V_k]_j$. If the minimum value of ΔV_k can be proven to always be less than zero (except at the equilibrium point where it's equal to zero) then the controller is globally asymptotically stable in a Lyapunov sense. This approach will now be attempted.

Expanding and simplifying Equation 6.21 gives

$$V_{k+1} = \frac{8\omega_k^2 \sin\left(\frac{1}{2}\left(T\omega_k + \theta_k\right)\right)^4 k_v v_M^2 + 2k_d \left(\left(\cos\theta_k - \cos\left(T\omega_k + \theta_k\right)\right) v_M + \omega_k y_k\right)^2}{2\omega_k^2}$$

Taking the partial derivative of the above equation with respect to ω_k and then simplifying, and letting

$$\begin{aligned} A\left(\omega_k\right) &= \sin\left(\frac{1}{2}\left(T\omega_k + \theta_k\right)\right)^2 \sin\left(T\omega_k + \theta_k\right) \\ B\left(\omega_k\right) &= \left(-\cos\theta_k + \cos\left(T\omega_k + \theta_k\right) + T\omega_k \sin\left(T\omega_k + \theta_k\right)\right) \\ C\left(\omega_k\right) &= \left(\left(-\cos\theta_k + \cos\left(T\omega_k + \theta_k\right)\right) v_M - \omega_k y_k\right) \end{aligned}$$

gives

$$\frac{\partial V_{k+1}}{\partial \omega_k} = \frac{2v_M \left(2k_v v_M T\omega_k^3 A\left(\omega_k\right) - k_d B\left(\omega_k\right) C\left(\omega_k\right)\right)}{\omega_k^3}$$

The above equation then needs to be equated to zero and solved for ω_k to find the turning points. However, inspection of the equation shows that this is not a realistic proposition. The method used for the stability proof for the first

controller is also not feasible as both θ_{k+1} and y_{k+1} simultaneously change in a fairly complex fashion as the control ω_k is changed, and, as a result, values of ω_k that lead to $\Delta V_k < 0$ for every possible starting point (θ_k, y_k) (other than $(\theta_k, y_k) = (0, 0)$ where we require $\Delta V_k = 0$) could not be found. What has been achieved, however, is a less ambitious mathematical proof of stability, that of *local stability* around the equilibrium point $(\theta_k, y_k) = (0, 0)$. This will now be given and then evidence of *asymptotic stability* at points well away from the equilibrium point will be provided graphically. For the proof of local stability, Theorem 6.4 is used.

Let us consider now an approximation to Equation 6.17 which is valid when ω_k is small, as is the case when θ_k and y_k are near the equilibrium point. The derivation of the approximate state equation follows.

Consider an object travelling in two dimensional space (x, y) under constant acceleration $\ddot{\mathbf{x}}_k = (\ddot{x}_k, \ddot{y}_k)$ for a period of time T. The equations describing the x component of its position and velocity are

$$x_{k+1} = x_k + \dot{x}_k T + \frac{1}{2}\ddot{x}_k T^2$$

and

$$\dot{x}_{k+1} = \dot{x}_k + \ddot{x}_k T$$

The same applies to the y component, resulting in the following 2D state equation

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k + \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}_k \tag{6.22}$$

Now, assuming that the acceleration is at right angles to the direction of

motion and referring to figure 6.2, we have

$$\begin{aligned} \ddot{x} &= -a_t \sin\theta \\ \ddot{y} &= a_t \cos\theta \end{aligned}$$

but

$$\begin{aligned} \sin\theta &= \frac{\dot{y}}{v_M} \\ \cos\theta &= \frac{\dot{x}}{v_M} \end{aligned} \tag{6.23}$$

hence

$$\begin{aligned} \ddot{x} &= \frac{-a_t}{v_M}\dot{y} \\ \ddot{y} &= \frac{a_t}{v_M}\dot{x} \end{aligned} \tag{6.24}$$

Substituting Equations 6.24 into Equation 6.22 gives the following state equation

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k + \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix} \begin{bmatrix} \frac{-a_t}{v_M}\dot{y} \\ \frac{a_t}{v_M}\dot{x} \end{bmatrix}_k \tag{6.25}$$

Simplifying Equation 6.25 results in

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & \frac{-a_t}{2v_M}T^2 \\ 0 & 1 & 0 & \frac{-a_t}{v_M}T \\ 0 & \frac{a_t}{2v_M}T^2 & 1 & T \\ 0 & \frac{a_t}{v_M}T & 0 & 1 \end{bmatrix}_k \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k \tag{6.26}$$

If the acceleration is part of a coordinated turn with angular velocity ω we have

$$\omega = \frac{a_t}{v_M}$$

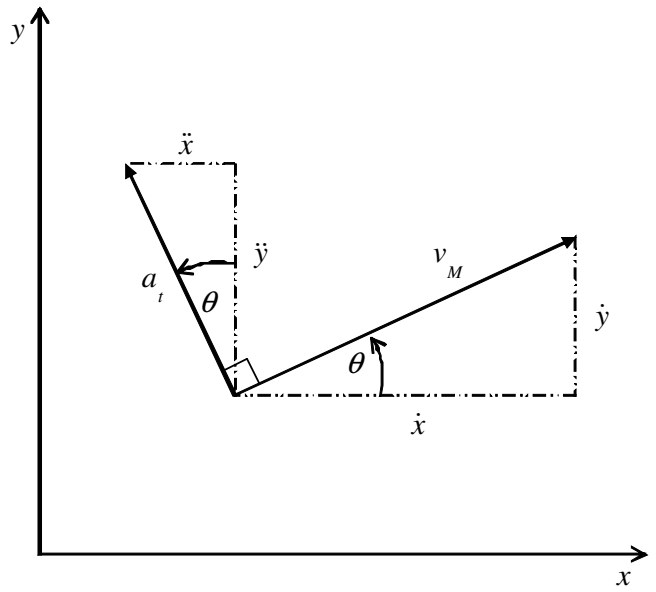

Figure 6.2: UAV Velocity and Acceleration Vectors and their x, y Components

Substituting into Equation 6.26 and moving the time-step subscript inside the matrices results in

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & \frac{-\omega_k}{2}T^2 \\ 0 & 1 & 0 & -\omega_k T \\ 0 & \frac{\omega_k}{2}T^2 & 1 & T \\ 0 & \omega_k T & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix} \tag{6.27}$$

Let us now compare Equation 6.17 with Equation 6.27. Taking the first term of the Taylor series about $\omega_k T = 0$ for $\sin \omega_k T$, $\cos \omega_k T$ and then the first two terms of the Taylor series of $\cos \omega_k T$ we arrive at the following approximations

$$\sin \omega_k T \approx \omega_k T \tag{6.28}$$

$$\cos \omega_k T \approx 1 \tag{6.29}$$

$$\cos \omega_k T \approx 1 - \frac{\omega_k^2 T^2}{2} \tag{6.30}$$

Substituting 6.28 into elements (1,2), (2,4), (3,4), and (4,2) of $A(k)$, 6.29 into elements (2,2) and (4,4) of $A(k)$, and 6.30 into elements (1,4), and (3,2) of $A(k)$ in Equation 6.17 we obtain

$$A(k) \approx \begin{bmatrix} 1 & T & 0 & \frac{-\omega_k}{2}T^2 \\ 0 & 1 & 0 & -\omega_k T \\ 0 & \frac{\omega_k}{2}T^2 & 1 & T \\ 0 & \omega_k T & 0 & 1 \end{bmatrix}$$

which is identical to the state transition matrix in Equation 6.27.

Now that it is shown from two points of view that Equation 6.27 is a valid approximation for the actual state equation in the neighbourhood of $\omega = 0$, let us rewrite Equation 6.27 in a form that is more helpful for determining the cost

difference equation ΔV_k. Rearranging Equation 6.27 gives

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \dot{x}_k T - \frac{\omega_k}{2} T^2 \dot{y}_k \\ \dot{x}_k - \omega_k T \dot{y}_k \\ y_k + \dot{y}_k T + \frac{\omega_k}{2} T^2 \dot{x}_k \\ \dot{y}_k + \omega_k T \dot{x}_k \end{bmatrix} \tag{6.31}$$

Substituting rows 2 and 3 of Equation 6.31 into Equation 6.12 we obtain

$$\begin{aligned} \Delta V_k &= k_d \left(y_k + \dot{y}_k T + \frac{\omega_k}{2} T^2 \dot{x}_k \right)^2 \\ &\quad + k_v \left(v_M - \dot{x}_k + \omega_k T \dot{y}_k \right)^2 - k_d y_k^2 - k_v \left(v_M - \dot{x}_k \right)^2 \end{aligned}$$

and then substituting Equation 6.23 into the above equation gives

$$\begin{aligned} \Delta V_k &= k_d \left(y_k + v_M T \sin\theta_k + \frac{\omega_k}{2} T^2 v_M \cos\theta_k \right)^2 \\ &\quad + k_v \left(v_M - v_M \cos\theta_k + \omega_k T v_M \sin\theta_k \right)^2 - k_d y_k^2 - k_v \left(v_M - v_M \cos\theta_k \right)^2 \end{aligned} \tag{6.32}$$

Expanding and collecting terms results in

$$\begin{aligned} \Delta V_k &= k_d v_M^2 T^2 \sin^2\theta_k + 2 k_d v_M y_k T \sin\theta_k + \\ &\quad \omega_k^2 \left(\frac{1}{4} k_d v_M^2 T^4 \cos^2\theta_k + k_v v_M^2 T^2 \sin^2\theta_k \right) + \\ &\quad \omega_k \left(k_d v_M T^2 \left(v_M T \sin\theta_k + y_k \right) \cos\theta_k + 2 k_v v_M^2 T \left(1 - \cos\theta_k \right) \sin\theta_k \right) \end{aligned}$$

Taking the partial derivative with respect to ω_k gives

$$\begin{aligned} \frac{\partial \Delta V_k}{\partial \omega_k} &= T^3 \cos\left(\theta_k\right) \sin\left(\theta_k\right) k_d v_M^2 + 2T \sin\left(\theta_k\right) k_v v_M^2 - 2T \cos\left(\theta_k\right) \sin\left(\theta_k\right) k_v v_M^2 \\ &\quad + 2\omega_k \left(\frac{1}{4} T^4 \cos^2\left(\theta_k\right) k_d v_M^2 + T^2 \sin^2\left(\theta_k\right) k_v v_M^2 \right) + T^2 \cos\left(\theta_k\right) k_d v_M y_k \end{aligned}$$

Equating the above equation to zero and solving for ω_k results in

$$\omega_k = -\frac{2 \left(k_d v_M T^2 \cos\theta_k \sin\theta_k + 2 k_v v_M \sin\theta_k - 2 k_v v_M \cos\theta_k \sin\theta_k + k_d y_k T \cos\theta_k \right)}{T \left(k_d T^2 \cos^2\theta_k + 4 k_v \sin^2\theta_k \right) v_M} \tag{6.33}$$

At this value of ω_k we must have either a maximum, minimum or inflection. Note here that letting $(\theta_k, y_k) \rightarrow (0,0)$ (i.e., the equilibrium point) in the above equation results in $\omega_k \rightarrow 0$ which is consistent with the assumption that was used to justify the above approximation for the A matrix, i.e., that ω_k is small.

Substituting Equation 6.33 into Equation 6.32, substituting the first two terms of the Taylor series for $\cos\theta_k$ and $\sin\theta_k$, i.e., $\cos\theta_k \approx 1 - \frac{\theta_k^2}{2}$, $\sin\theta_k \approx \theta_k$ into the resulting equation and then simplifying and expanding gives

$$\begin{aligned}
\Delta V_k \approx & -\frac{4T^2k_d^2y_k^2}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} + \frac{4T^2k_d^2y_k^2\theta_k^2}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} \\
& + \frac{24Tk_dk_vv_My_k\theta_k^3}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} + \frac{8T^2k_dk_vv_M^2\theta_k^4}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} \\
& - \frac{T^2k_d^2y_k^2\theta_k^4}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} + \frac{4Tk_dk_vv_My_k\theta_k^5}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} \\
& + \frac{4T^2k_dk_vv_M^2\theta_k^6}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} - \frac{4k_v^2v_M^2\theta_k^6}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2}
\end{aligned} \tag{6.34}$$

Now as $(\theta_k, y_k) \rightarrow (0,0)$, terms 2 to 8 in Equation 6.34 become negligible leaving only term 1, i.e.,

$$\Delta V_k \approx \frac{-4T^2k_d^2y_k^2}{16k_v\theta_k^2 + T^2k_d\left(-2+\theta_k^2\right)^2} \tag{6.35}$$

In the above equation k_v and k_d are positive, hence the denominator is the sum of two terms which are ≥ 0 but which can never both be zero at the same time, and thus, the denominator is > 0. Also, irrespecive of the value of θ_k, the numerator is < 0 for all $y_k \neq 0$, and $= 0$ when $y_k = 0$, therefore $\Delta V_k(\theta_k, y_k) = 0$ at the equilibrium point $(\theta_k, y_k) = (0,0)$, and $\Delta V_k(\theta_k, y_k) \leq 0$ in the neighbourhood of the equilibrium point, i.e., $\Delta V_k(\theta_k, y_k)$ is negative semi-definite. Hence, using Theorem 6.4, the above controller is, at the very least, *locally stable* in the neighbourhood of $(\theta_k, y_k) \rightarrow (0,0)$.

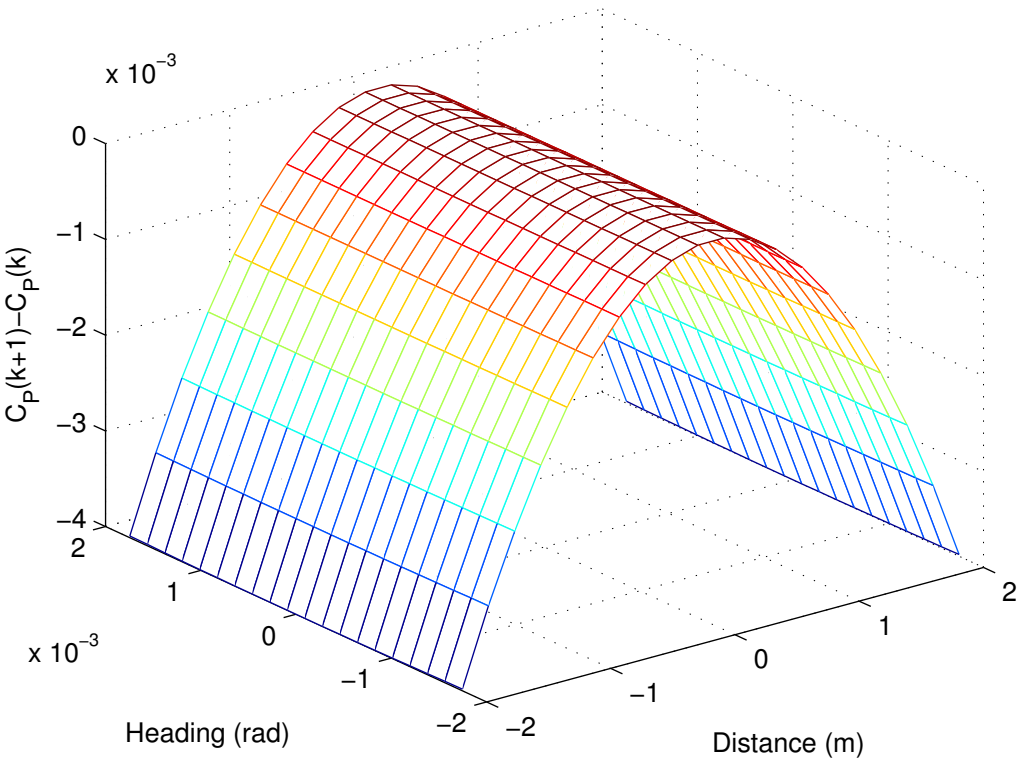

Figure 6.3: Plot of Equation 6.35

Now let us verify the validity of the approximations that we used to arrive at Equation 6.35. Figure 6.3 shows a plot of Equation 6.35 and Figure 6.4 shows a plot of Equation 6.34. Comparison of the two plots shows that they are almost identical over the (small) range of heading and distance considered in the figures. This is as would be expected, with the solutions to the two equations approaching each other as the range of heading and distance is decreased.

Figure 6.5 shows a plot of the *actual iteratively derived cost differences*. Note that to produce this plot the iterations were performed by trying all possible accelerations from -5.0 m/s^2 to +5.0 m/s^2 in steps of 0.01 m/s^2 and finding

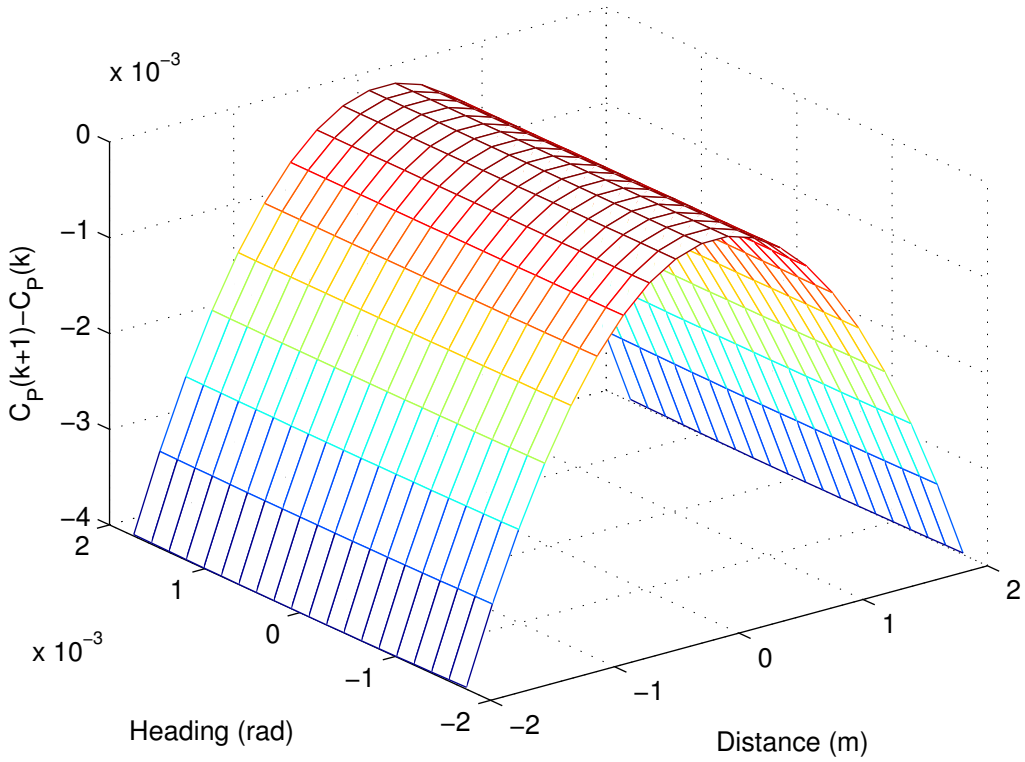

Figure 6.4: Plot of Equation 6.34

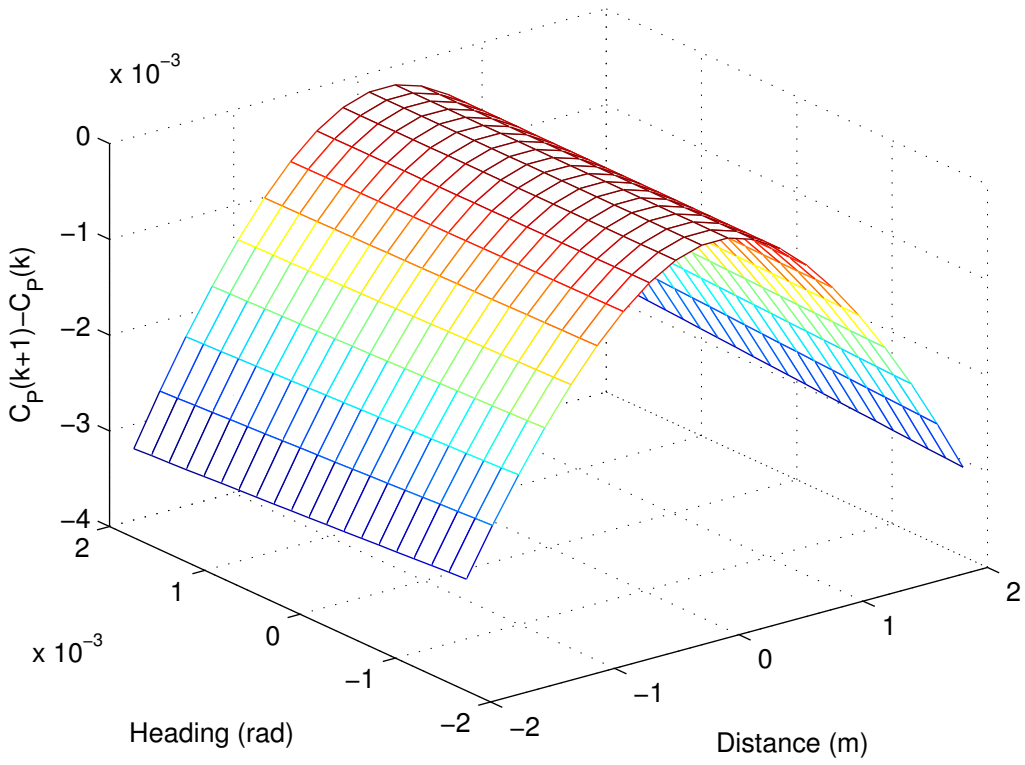

Figure 6.5: Actual Iterative Controller Cost Difference

the acceleration that leads to the lowest cost. In all three plots, i.e., Figures 6.3, 6.4 and 6.5, the range of the heading θ_k and distance y_k considered was $\frac{-\pi}{1800} \leq \theta_k \leq \frac{\pi}{1800}$ radians, i.e., $-0.1 \leq \theta_k \leq 0.1$ degrees and $-2 \leq y_k \leq 2$ metres respectively. Comparison of the first two plots with the third shows that both approximations are very similar to the actual cost differences seen in the third plot over the (small) range of heading θ_k and distance y_k considered, thus confirming the validity of the equations that the first two plots represent.

As mentioned earlier in this section, global stability around the equilibrium

point could not be proved mathematically, however strong evidence of asymptotic stability at points well away from the equilibrium point can be provided graphically and this will be shown now. Firstly, Figure 6.6 shows the single-step transitions for the actual iterative algorithm. As with Figure 6.5, the iterations were performed by trying all possible accelerations from -5.0 m/s^2 to +5.0 m/s^2 in steps of 0.01 m/s^2 and finding the acceleration that leads to the lowest cost for the update in question. However, here the range of the heading θ_k and distance y_k at time k considered was $-\pi \leq \theta_k \leq \pi$ radians and $-3600 \leq y_k \leq 3600$ metres respectively. In the figure the small circles correspond to the values of (θ_k, y_k) at time k, the dots at the other end of the line segments correspond to the values of (θ_{k+1}, y_{k+1}) at time $k+1$, and the line segments joining the circles to the dots represent the transition from time k to time $k+1$. The line segments and associated circle and dot that are in red indicate increasing or constant cost from time k to time $k+1$, whereas those in green indicate decreasing cost. The aim of the figure is to graphically show that for any starting point in the (θ, y) plane within the range considered, the subsequent transition is consistent with a general "flow" towards the equilibrium point at $(\theta, y) = (0, 0)$. Looking at the figure, one sees that for the majority of starting points the transitions are primarily a change in θ towards a small negative value when $y > 0$ and a small positive value when $y < 0$, with a much smaller corresponding change in y. Once a critical value of θ is reached there appears to be an abrupt change in direction of the transitions such that there is primarily a decrease in the magnitude of y. Note that while some of the transitions correspond to an increase in cost for that transition the figure indicates that they soon lead to future transitions where the cost is again decreasing. While covering a large number of starting points, Figure 6.6 doesn't show the change in direction toward decreasing y clearly, as it occurs too abruptly, so Figure 6.7 was produced to show this more clearly.

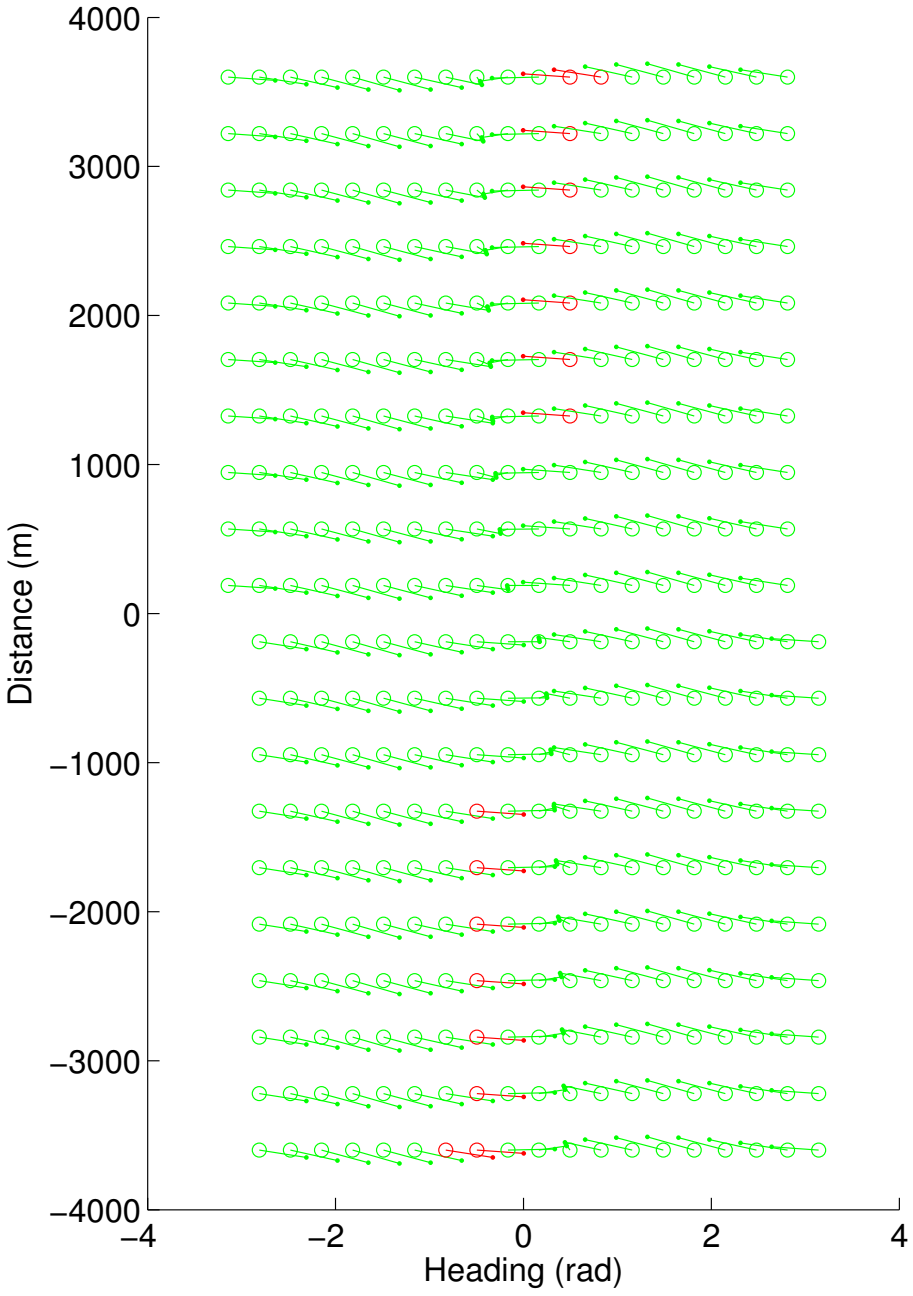

Figure 6.6: Single-Step Transitions of Iterative Controller

Figure 6.7 has much fewer starting points but follows the trajectories all the way to convergence. As for Figures 6.5 and 6.6, the iterative algorithm was used to produce this figure and the iterations were performed by trying all possible accelerations from -5.0 m/s^2 to +5.0 m/s^2 in steps of 0.01 m/s^2. The figure shows the θ and y components of the trajectories from 36 different starting points, which are marked with black dots. The abrupt change in direction of the transitions can now be seen clearly in this figure. Importantly, all the trajectories converge to the equilibrium point $(\theta, y) = (0, 0)$, which is marked with a blue dot in the figure. In this figure also, the parts of the trajectories during which the cost is increasing are shown in red. As can be seen from the figure, despite travelling through sections of increasing cost, the trajectories all converge to the equilibrium point.

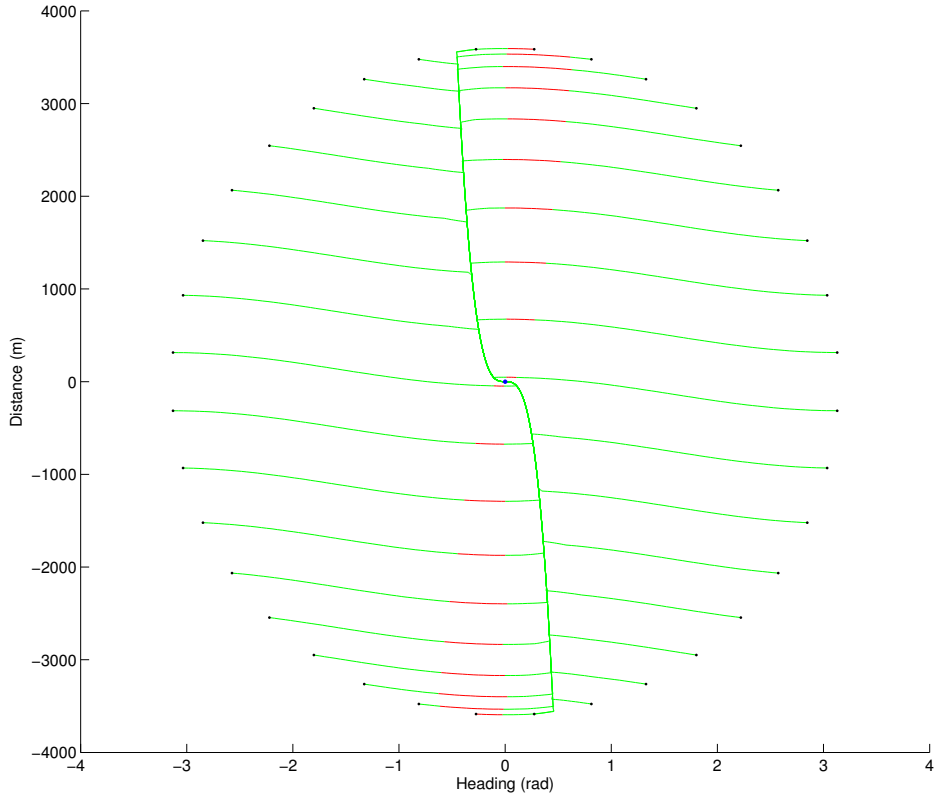

Figure 6.7: Theta and y components of Trajectories from a Range of Starting Points.

Chapter 7

Concluding Remarks

The following section will summarize the outcomes of the work covered by this thesis; this will be followed in the subsequent section by some suggestions for possible future work.

7.1 Summary and Conclusions

The central theme of this thesis has been the development of an algorithm for the control of a UAV-based passive distributed sensing system. The problem that has been dealt with can be considered to fall within a broader class of problems which we have referred to as cooperative sensor networks. It is also clearly a control problem and has considerable overlap with work that has been done in robotics, and of course military doctrine has needed to be taken into account in the development of a solution. Chapter 1 of the thesis developed the motivations for the work and began to introduce some of these concepts. This was followed in Chapter 2 by a comprehensive review of the scientific literature, firstly on broadly related research problems and then on problems

that are more closely related to ours. The review brought together battle-management concepts from the military and research on a number of related electronic warfare sensor management problems. It also demonstrated how our problem fits within the context of these, and described the current state of the art for these types of problems. As a result of this review, it was determined that no other significant works had been presented in the literature that deal with the unique combination of issues dealt with in this thesis.

Chapter 3 presented the theoretical background that is required for the subsequent chapters. The contents of the chapter are the result of a broad survey of modern optimal control theory and techniques, which was performed with the aim of finding the most appropriate solution approach for the sensor management problem being considered. The chapter brings together theory sourced from several texts and research papers and presents them in a concise manner which was designed to help clarify and demonstrate the basis for the development of the algorithms in the subsequent chapters.

A novel controller using a hierarchical variant of the model predictive control approach is then developed in Chapter 4 to address the adaptive multisensor multitarget tracking problem. The controller consists of a fine-grained controller and a planner, organized in a manner that is new to this type of adaptive sensing problem. A key feature of the controller is the incorporation of long-term goals utilizing a dynamic programming based formulation. Also, the control algorithm was designed from the outset to be usable in real-time systems. This goal had a major impact on the solution approach which, in fact, successfully met this requirement. As part of the development of the controller, two novel path planning algorithms were presented. Both algorithms were integrated into the hierarchical controller and the performance of the system tested by simulation and compared with a "myopic" controller similar to that commonly seen in the

literature. The simulations showed clear advantages of the new algorithm, in particular avoidance of entrapment by obstacles.

Chapter 5 takes the development of the control algorithm further, first incorporating a dynamics model for fixed-wing aircraft and then making the necessary modifications to the algorithm to enable it to achieve its task with the more restrictive constraints imposed by the type of UAV being considered. Simulations were then performed to demonstrate the performance of the algorithm in its optimization of UAV trajectories. The simulations showed that, with the changes made, the algorithm works well with the constraints imposed by the type of fixed-wing aircraft considered, thus further demonstrating the effectiveness of the approach and usability in a practical distributed sensing application.

A fundamental requirement of any control system is that it be stable. The controllers that were developed in Chapters 4 and 5, when tested by simulation, showed performance that was consistent with expectations and, in particular, demonstrated stable behaviour. Generally in addition to simulation and testing it is desirable to prove this property analytically. Chapter 6 presents an effort to go some way to producing an analytical proof. As it turns out, stability analysis of the controllers that have been developed is a highly challenging task; however substantial inroads were made by focusing on components of the system. Together with the results of simulations in the chapters 4 and 5 the analysis provided additional evidence that the controller should be stable; however, further mathematical analysis is still required to give a deeper understanding of the behaviour of the controllers and more confidence in their overall stability.

It is important to note the control approach that has been developed for the problem considered in this thesis has much more general applicability than may be immediately apparent. From the outset the aim was to produce a solution that is scalable. Clearly simple extensions such as varying the numbers of UAVs

that are assigned to each target, or varying the type of sensor (or target) can be relatively easily incorporated. But more importantly, the work can, for example, be readily generalized to combine the problem of sensor platform steering with jammer control, radar control, and even weapon trajectory control, using the same general approach. However, ingenuity is required, because for all these cases the computational complexity of a full direct dynamic programming solution will be beyond the capability of realizable computing systems. For each extension or variation of the problem, approximations specific to that problem will in general need to be used to achieve implementable solutions. Unfortunately there will be no "magic bullet" solution for the type of complex real-world problems described in this thesis; however, the optimal control approach described can be used as a guiding principle to develop solutions that, while approximate, still have a sound basis in theory.

7.2 Future Work

As mentioned in the previous section, the algorithms presented in this thesis concentrate on a fairly specific problem while using techniques that can be easily modified for application to variants of the problem that may for example involve different types of sensor. The techniques are also scalable to more complex problems. Possible areas of future work that have practical application are:

(a) Considering different types of sensors on the UAVs. Particular types that have military relevance are detected-power-level sensors, time-difference-of-arrival sensors, and a technique called scan-based localization [23] which uses a less sensitive type of time-difference-of-arrival algorithm, which only applies to the geolocation of mechanically scanned radars.

(b) Detecting and tracking different types of emitters. An important example of this is the geolocation of communications emitters such as mobile phones, or military communications devices.

(c) Expanding the problem to more complicated systems, with multiple types of platforms being controlled. An example of this could, say, be a system where we have multiple UAVs carrying passive sensors as considered in this thesis as well as possibly optical sensors, jammers mounted on other UAVs, and a manned aircraft with an adaptively scanned radar. Considering how this system could be optimized to detect, geolocate and destroy a surface based radar while minimizing the risk to the manned aircraft's crew is clearly a very important problem to the military. It would also provide an excellent test/demonstration of the scalability of the techniques developed in this thesis.

The other main area of future work that could be done is to complete the stability analysis of the controllers, which was partially completed in Chapter 6, building up to a thorough proof of stability of the entire systems. Again relating to stability, time delays in the system and their effect on stability have not been considered to date. Depending on how the actual control system is implemented, time delays may possibly be an issue that needs to be considered and this should also be considered as possible future work.

Bibliography

[1] James S. Albus, Alexander M. Meystel, *Engineering of Mind - An Introduction to the Science of Intelligent Systems*, John Wiley and Sons, Inc., 2001.

[2] Yaakov Bar-Shalom, Xiao-Rong Li, *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, 1993.

[3] John Bellingham, Arthur Richards, Jonathan P. How, "Receding Horizon Control of Autonomous Aerial Vehicles", *Proceedings of the American Control Conference*, Anchorage, Alaska, May 8-10, 2002.

[4] Richard Bellman, *Dynamic Programming*, Courier Dover Publications, 2003, Originally published by Princeton University Press, 1957.

[5] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control - Volume I*, Second Edition, Athena Scientific, Belmont, Massachusetts, 2000.

[6] D. P. Bertsekas and D. A. Castanon, "Rollout Algorithms for Stochastic Scheduling Problems", *Journal of Heuristics*, Vol. 5, pp. 89-108, 1999.

[7] Stephen Boyd, Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[8] E. F. Camacho and C. Bordons, *Model Predictive Control*, Second Edition, Springer-Verlag London Ltd., 2007.

[9] C. C. Chen and L. Shaw, "On Receding Horizon Feedback Control", *Automatica*, 18, pp. 349-352, 1982.

[10] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein C., "*Introduction to Algorithms*", Second Edition, MIT Press, 2001.

[11] Thomas M. Cover, Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

[12] Kutluyil Dogancay, "Online Optimization of Receiver Trajectories for Scan-Based Emitter Localization", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 43, No. 3, pp. 1117-1125, July 2007.

[13] Kutluyil Dogancay, Hatem Hmam, "Optimal Angular Sensor Separation for AOA Localization", *Signal Processing*, Vol. 88, Issue 5, p.p. 1248-1260, May 2008.

[14] Robin Evans, Vikram Krishnamurthy, Girish Nair, and Len Sciacca, "Networked Sensor Management and Data Rate Control for Tracking Maneuvering Targets", *IEEE Transactions on Signal Processing*, Vol. 53, No. 6, June 2005.

[15] A. Farina, F. A. Studer, *Radar Data Processing; Volume I - Introduction and Tracking*, Research Studies Press Ltd., 1985.

[16] A. Finn, K. Brown, T. Lindsay, "Miniature UAV's & Future Electronic Warfare", *Land Warfare Conference*, October 2002, pp. 93-106, Brisbane, Australia.

[17] Eric W. Frew, Cory Dixon, Brian Argrow, and Tim Brown, "Radio Source Localization by a Cooperating UAV Team", *Infotech@Aerospace*, 26-29 September 2005, Arlington, Virginia, USA.

[18] G. C. Goodwin, M. M. Seron, J. A. De Dona, *Constrained Control and Estimation - An Optimisation Approach*, Springer-Verlag London Ltd, 2005.

[19] T. Hanselmann, M. Morelande, "Multiple target tracking with asynchronous bearings-only-measurements", *10th International Conference on Information Fusion 2007*, 9-12 July 2007.

[20] Thomas Hanselmann, Mark Morelande, Bill Moran, Peter Sarunic, "Sensor Scheduling for multiple target tracking and detection using passive measurements", *2008 11th International Conference on Information Fusion*, June 30 - July 3, 2008.

[21] Ying He, Edwin K. P. Chong, "Sensor Scheduling for Target Tracking in Sensor Networks", *43rd IEEE Conference on Decision and Control*, December 14-17, 2004.

[22] O. Hernandez-Lerma, *Adaptive Markov Control Processes*, Springer-Verlag New Yorke Inc., 1989.

[23] H. Hmam, "Scan-Based Emitter Passive Localization", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 43, No. 1, January, 2007.

[24] G. J. Holland, P. J. Webster, J. A. Curry, G. Tyrrell, D. Gauntlett, G. Brett, J. Becker, R. Hoag, and W. Vaglienti, "The Aerosonde Robotic Aircraft: A New Paradigm for Environmental Observations", *Bulletin of the American Meteorological Society*, pp. 889-901, 2001.

[25] Colin Howson, Peter Urbach, *Scientific Reasoning: The Bayesian Approach*, Open Court Publishing Company, 1989.

[26] Rufus Isaacs, *Differential Games*, John Wiley and Sons, Inc., 1965.

[27] R. A. Jarvis, "Collision-free Path Trajectory using Distance Transforms", *Proc. National Conference and Exhibition on Robotics* - Melbourne, 1984.

[28] Finn V. Jensen, *Bayesian Networks and Decision Graphs*, Springer-Verlag New York, Inc., 2001.

[29] S. S. Keerthi and E.G. Gilbert, "Optimal, Infinite Horizon Feedback Laws for a General Class of Constrained Discrete Time Systems: Stability and Moving Horizon Approximations", *Journal of Optimization Theory and Application*, 57, pp. 265-293, 1988.

[30] O. Khatib, "Real-time Obstacle Avoidance for Manipulatiors and Mobile Robots", *International Journal of Robotics Research*, 5(1), 1986.

[31] Vikram Krishnamurphy, "Algorithms for Optimal Scheduling and Management of Hidden Markov Model Sensors", *IEEE Transactions on Signal Processing*, Vol. 50, No. 6, June 2002.

[32] Vikram Krishnamurphy and Robin J. Evans, "Hidden Markov Model Multiarm Bandits: A Methodology for Beam Scheduling in Multitarget Tracking", *IEEE Transactions on Signal Processing*, Vol. 49, No. 12, December 2001.

[33] P. R. Kumar, Pravin Varaiya, *Stochastic Systems - Estimation, Identification and Adaptive Control,* Prentice-Hall Information and System Sciences Series, 1986.

[34] B. F. La Scala, W. Moran, R. J. Evans, "Optimal adaptive waveform selection for target detection", *Proceedings of the International Radar Conference*, 3-5 September, 2003.

[35] L. S. Lasdon, *Optimization Theory for Large Systems*, MacMillan, New York, 1971.

[36] Steven M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[37] S. M. LaValle, "Rapidly-exploring Random Trees: A New Tool for Path Planning", TR 98-11, Computer Science Dept., Iowa State University, October, 1998.

[38] Cindy Leung, Shoudong Huang, Ngai Kwok, Gamini Dissanayake, "Planning Under Uncertainty Using Model Predictive Control for Information Gathering", *Robotics and Autonomous Systems* 54 (2006), pp. 898-910.

[39] Martin M. Lipschutz, *Theory and Problems of Differential Geometry*, Schaum's Outline Series, McGraw-Hill Book Company, 1969.

[40] Mohamed Marzouqi and Ray A. Jarvis, "Efficient Robotic Pursuit of a Moving Target in a Known Environment Using a Novel Convex Region Segmentation", *2nd International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, December 13-15, 2004.

[41] Mayne, D. Q. and Michalska, H., "Receding Horizon Control of Non-linear Systems", *IEEE Transactions on Automatic Control*, 35(5), pp.814-824, 1990.

[42] Mayne, D. Q., Rawlings, J. B., Rao, C. V. and Scokaert, P. O. M., "Constrained Model Predictive Control: Stability and Optimality", *Automatica*, Vol. 36, pp. 789-814, 2000.

[43] Morari, M., and Lee, J . H., "Model Predictive Control: Past, Present, and Future", *Computers and Chemical Engineering*, Vol. 23, pp. 667-682, 1999.

[44] N. J. Nilsson, "A Mobile Automation: An Application of Artificial Techniques", *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pp. 509-520, 1969.

[45] Yaakov Oshman, Pavel Davidson, "Optimization of Observer Trajectories for Bearings-Only Target Localization", *IEEE Trans. on Aerospace and Electronic Systems*, vol.35, no. 3, pp. 892-902, July 1999.

[46] Fazlollah M. Reza, *An Introduction to Information Theory*, McGraw-Hill Book Company, Inc., 1961.

[47] Riccardo Scattolini, "Architectures for Distributed and Hierarchical Model Predictive Control - A Review", *Journal of Process Control* 19 (2009), pp. 723-731.

[48] L. Sciacca, "Distributed Electronic Warfare Sensor Networks", *Association of Old Crows Convention*, 2002.

[49] L. J. Sciacca, A. Cullen and G. K. McCleery, "Cooperative Sensor Networks: Sensor Web Concepts for the Modern Warfighter", *Land Warfare Conference 2001*, Sydney, November 2001.

[50] Len J. Sciacca, Robin J. Evans, William Moran, Sofia Suvorova, "Cooperative Sensor Networks: A Stochastic Sensor Scheduling Approach", *Final Program and Abstracts, Information, Decision and Control Conference*, 2002.

[51] P. W. Singer, "War of the Machines", *Scientific American*, July 2010.

[52] M. G. Singh, "*Dynamical Hierarchical Control*", North-Holland Publishing Co., 1977.

[53] Madan G. Singh, Stephen A. W. Drew, John F. Coales, "Comparisons of Practical Hierarchical Control Methods for Interconnected Dynamical Systems", *Automatica*, Vol. 11, pp 331-350, Permagon Press, 1975.

[54] Sumeetpal S. Singh, Nikolaos Kantas, Ba-Ngu Vo, Arnaud Doucet, Robin J. Evans, "Simulation-based Optimal Sensor Scheduling with Application to Observer Trajectory Planning", *Automatica* 43 (2007), pp. 817-830.

[55] D. J. Torrieri, "Statistical Theory of Passive Location Systems", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-20, No. 2, March 1984.

[56] Gregory J. Toussaint, Pedro De Lima, Daniel J. Pack, "Localizing RF Targets with Cooperative Unmanned Aerial Vehicles", *Proceedings of the 2007 American Control Conference*, New York City, USA, July 2007.

[57] M. Vidyasagar, *Nonlinear Systems Analysis*, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey.

[58] Kruger White, Jason Williams, Peter Hoffensetz, "Radar Sensor Management for Detection and Tracking", *2008 11th International Conference on Information Fusion*, June 30 - July 3, 2008.

[59] J. L. Willems, *Stability Theory of Dynamical Systems*, Thomas Nelson and Sons Ltd., 1970.

[60] Hans S. Witsenhausen, "Separation of Estimation and Control for Discrete Time Systems", *Proceedings of the IEEE*, Vol. 59, No. 11, November 1971.