INFORMATION-THEORETIC MANAGEMENT OF MOBILE SENSOR AGENTS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Zhijun Tang, B.S., M.S.

* * * * *

The Ohio State University

2005

Dissertation Committee:

Prof. Ümit Özgüner, Adviser

Prof. Kevin M. Passino

Prof. Yuan F. Zheng

Approved by

Adviser

Graduate Program of
Electrical and Computer
Engineering

ABSTRACT

Sensor management (SM) is one of the key factors that determine the performance of a multi-sensor system, which becomes even more crucial when the system consists of mobile sensor agents (MSA). Due to the strong interconnection between information processing and MSA motion control, a new set of theoretical foundations, design principles and performance metrics are needed to develop a successful MSA management approach. This dissertation presents a few attempts toward this goal by jointly studying the target track maintenance problem and the MSA motion-planning problem through the MSA-target scenario, which is motivated by the emerging applications of Unmanned Air Vehicles (UAV) in both military and civilian sensing tasks, such as surveillance, reconnaissance and rescuing.

First, by analyzing the advantages and disadvantages of existing target tracking approaches, a generic method for target track maintenance, the BF-HMap approach, is proposed based on the Bayesian filtering method and the *hospitability map*. An advanced version of this approach with much less computational and memory load using a particle filter, the PF-HMap algorithm, is also introduced in this work. Due to the flexible scheme of Bayesian inference inherent in both algorithms, BF-HMap and PF-HMap are capable of exploiting non-analytic prior environmental knowledge as well as handling intermittent and regional measurements caused by the coverage and motion constraints on MSAs. Meanwhile, a generalized particle filter for both

in-sequence and out-of-sequence measurements, the *Universal Particle Filter* (UPF), is developed for possible extensions of the PF-HMap algorithm to distributed MSA networks.

Secondly, the MSA motion-control problem is studied in such an information-theoretic way that the conditional entropy (i.e. given the measurements) of the target state is chosen as a generic performance metric. The evolutions of the entropy in both the linear and the non-linear case are analyzed, and several key properties of the evolution of the entropy are identified. These properties are further exploited to model the *sensor management* problem in two cases of studies: target search and target surveillance, which are modeled as a stabilization problem of the entropy and an optimization problem to minimize the average revisit time on each target, respectively. Based on these information-theoretic formulations, a necessary condition and a sufficient condition by means of the number of MSAs to perform a non-escape search for a moving target are derived. Given a sufficient number of MSAs, a cooperative search formation, called the *Progressively-Spiral-In* algorithm (PSI), is also developed for the MSA team to find the target in finite time. In the meantime, a cooperative motion-planning approach for multi-target surveillance by multiple non-holonomic MSAs is proposed. In this approach, the targets are divided into disjoint groups through on-line task decomposition. For each target group, a sub-optimal traversal motion plan is generated for an MSA to periodically update the track information of the targets in the shortest time.

The dissertation concludes with a summary of the current work and a discussion on possible extensions of the proposed methodologies in future work.

ACKNOWLEDGMENTS

My thanks go first and foremost to my wife, Yi, and my parents, who have supported me with love, patience, and understanding throughout my tenure as a graduate student.

I also extend my gratitude to my advisor, Dr. Ümit Özgüner, who gave me the inspiration, confidence, and freedom to explore in the challenging field of sensor management.

My gratitude also goes to the entire committee, Dr. Kevin Passino and Dr. Yuan F. Zheng, who provided the moral and technical support that carried this dissertation through to completion.

Finally, special thanks go to my colleagues, Dr. Keith Redmill, Mr. Jone, Mr. Hai Yu, Ms. Yiting Liu, Mr. Qi Chen, Ms. Lu Xu, Mr. Rong Xu, Mr. George McWilliams, Mr. Yongling Zheng and all my friends, whom I have shared the journey through the graduate school with. Their company in the good times and their advice in times of trouble have been invaluable.

This work was supported in part by a TRC ITS Graduate Fellowship, a DAGSI Graduate Fellowship, and by the grant from AFRL/VA and AFOSR Collaborative Center of Control Science under contract F33615-01-2-3154.

VITA

January 30, 1975 ..... Born - Jinzhou, China

1997 ..... B.S. E.E.
Zhejiang University, China.

2000 ..... M.S. E.E.
Zhejiang University, China.

PUBLICATIONS

Z. Tang and U. Ozguner, "Motion Planning for Multi-target Surveillance with Multiple Mobile Sensor Agents", To appear in *IEEE Trans. Robotics*, 2005.

Z. Tang and U. Ozguner, "Sensor fusion for target track maintenance with multiple UAVs based on the Bayesian filtering method and the hospitability map", *in Proc.* 42^{th} *IEEE Conf. Decision and Control*, vol.1 , pp. 19-24, 2003.

Z. Tang and U. Ozguner, "An adaptive motion correspondence algorithm for multiple vehicle tracking by an airborne platform", *in Proc. IEEE International Conf. ITS*, pp. 49-54, 2002.

FIELDS OF STUDY

Major Field: Electrical and Computer Engineering

Studies in:

Computer Engineering
Signal Processing
Control
Intelligent Transportation Systems

TABLE OF CONTENTS

Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivations

Modern information gathering tasks such as domestic traffic surveillance, environment monitoring, border patrol and battlefield reconnaissance often require multiple sensor systems to work cooperatively to produce and maintain a high-level representation of the environment. *Sensor management* (SM), which has been referred as the general process of controlling and using available sensor resources [1, 3, 4], is a key factor in determining the overall sensing performance.

As pointed out by Johansson et al. [16], with the increasing complexity of sensor platforms (as well as that of their applications), the sensor management problem has been extended considerably beyond the scope of control. In particular, there is a close coupling between information processing and sensor control in intelligent sensor systems. Controlling the sensors is basically for the purpose of a better acquisition of information, which will be processed to build up the sensors' knowledge of what is happening in the environment. At the same time, the knowledge built by a good information processing approach can also help the sensors to make appropriate control decisions, which will further lead to a better acquisition of information in the future.

In recent years, rapid advances in technology have made it possible to perform complex sensing tasks using multiple inexpensive, intelligent, mobile sensor platforms such as Unmanned Air Vehicles (UAVs). We denote this type of sensor platforms with an integrated capability of sensing, processing, communication and locomotion as *mobile sensor agents* (MSAs) in this study. This definition is adapted from the term *agent* in the domain of artificial intelligence [18,19], which is defined as an entity that is able to perceive the environment and react upon its perceptions.

The mobility of MSAs allows themselves to collect information in hostile or remote environments such as battlefields, where stationary sensors are not available. In addition, MSAs can also reconfigure their tasks, objectives and kinematic states dynamic environments such as transportation systems and flow networks. All these advantages of MSAs over traditional stationary sensors, however, do not come without a price. Many MSA applications (e.g. target search) have a field of interest much larger than the field of view (FOV) of each individual MSA. With a limited FOV, the observations of an MSA is highly related to its kinematic status (e.g. position and orientation). However, an MSA can only adjust its position, speed and orientation through a continuous trajectory, possibly restricted by some other physical constraints (e.g. minimum turning radius, minimum/maximum speed). The restricted FOV and physical motion constraints of MSAs have brought new challenges to both information processing and sensor management.

First, although problems in various sub-domains of sensor information processing have been extensively studied by researchers, most of existing approaches either suitable to stationary sensors only, or neglect possible motion and coverage constraints

on MSAs. Consequently, the focus has always been on how to generate the best estimate (usually by means of minimum mean square error (MMSE)), given a sequence of measurements. However, due to a limited FOV, the measurements that an MSA can obtain are directly affected by where it is and where it is looking at. Unfortunately, to incorporate this factor with existing sensor information processing approaches is not a trivial job. The highly non-linear sensor model brought by a limited FOV of an MSA complicates the solution to the information processing problem, even in a very simple (e.g. linear and Gaussian) setup. In addition, since a good observation is not a given any longer, more pressure has been put on the information processing module to provide the motion controller an effective, on-line representation of the environment, both prediction-wise and estimation-wise.

Second, the majority of previous work in the area of *sensor management* treats the sensor control problem as (or similar to) a scheduling problem [6–15]. The cooperation of multiple sensors is often achieved by choosing different sensors (or sensor modes) for different tasks (targets) at different time. Motion planning, on the other hand, is seldom involved. Unfortunately, because of the physical motion constraints of MSAs, the transition between assignments of an MSA has to satisfy a physically feasible path. For the same reason, there may also be a non-trivial transition time between assignments. As a result, many existing sensor-scheduling methods fall short in solving the management problem for MSAs. Recently, more and more original work dealing with sensing problems in the context of MSAs (e.g. target search or target surveillance by UAVs/ground robots) has been reported in literature, especially in the domain of control science [42,49,50,81,82,84,85,87,88], in which motion planning has replaced scheduling/tasking in the formulation of sensor management problems.

However, very limited effort has been made to encode the evolution of information into the modeling of the *sensor management* problem of MSAs. Instead, a variety of heuristic assumptions have been used without rigorous mathematical justification. Thus, it is very difficult to validate a control algorithm's feasibility and quality, or to compare different sensor management approaches. There also have been a few pioneer attempts [17] to jointly consider the information processing problem and sensor control problem for MSAs, which, almost without any exception, focus on the linear-Gaussian case only. Consequently, the capability of these approaches in handling non-linear *sensor management* problems for MSAs with restricted FOVs and physically motion constraints is limited.

1.2 Problem Statement

In this work, we investigate the *sensor management* problem for MSAs through the MSA-target scenario.

Each target here is modeled as a moving object on a 2-D plane, whose kinematic state is denoted as $x^j(t) \in \mathcal{R}^{n_x}$. Let N be the number of targets and $x = \{x^j\}_{j=1..N} \in \mathcal{R}^{N \times n_x}$. The motion of $x(t)$ is assumed to be governed by the following Ito equation:

$$dx(t) = f(x,t) + g(x,t)d\beta(t),\ t \geq t_0. \tag{1.1}$$

Here $f(\cdot)$ models the deterministic part of the dynamics of $x(t)$, which comes from the prior knowledge of how $x(t)$ evolves over time. $\beta(t)$ is the process noise, and $g(x,t)d\beta(t)$ stands for the random part of the dynamics of $x(t)$.

Each MSA is simply modeled as a point mass moving at a constant speed on a 2-D plane as follows, which has been widely used as the kinematic model for fixed-wing

UAVs in research [21, 38, 55, 86].

$$\begin{cases} \dot{x}_s^i = v_s^i \cos\theta_s^i, \\ \dot{y}_s^i = v_s^i \sin\theta_s^i, \\ \dot{\theta}_s^i = u^i. \end{cases} \tag{1.2}$$

Here $s^i = [x_s^i, y_s^i]^T \in \mathcal{R}^2$ and θ_s^i denote the 2-D position and orientation of the i^{th} MSA, and u^i stands for the controller on the i^{th} MSA and v_s^i is its the cruise speed. Let M be the number of MSAs, We also use $s = \{s^i\}_{i=1..M} \in \mathcal{R}^{2M}$, $\theta = \{\theta^i\}_{i=1..M} \in \mathcal{R}^M$ and $u = \{u^i\}_{i=1..M} \in \mathcal{R}^M$ as collection of the positions, orientations and controllers of all the MSAs. Note that in reality, this model is subject to some motion constraints such as a minimum turning radius.

Each MSA is assumed to be equipped with an on-board sensor system, which has a restricted FOV (i.e a footprint) centered at s^i. The measurement $y(t)$, which is often taken by the MSAs at discrete time instants $\{t_1, t_2, ..., t_k, ...\}$ is modeled as:

$$y(t_k) = H(x(t_k), t_k) + \nu(t_k),\ k = 1, 2, \ldots. \tag{1.3}$$

where $\nu(t_k)$ is the measurement noise.

By periodically extracting target information from newly obtained measurements, the MSAs are adding to their knowledge of the evolution of $x(t)$. Although such an information processing operation is commonly considered as a procedure of *target tracking*, it actually takes place in almost all the information gathering problems. Due to this reason, we use a more general term, *target track maintenance*, in this work to denote the information processing problem in the MSA-target scenario.

Since *target track maintenance* is strongly coupled with sensor management here, we study these two problems jointly in this work. Considering the lack of comprehensive theories for the management of MSAs and the gap between contemporary studies in *target track maintenance* and mobile sensor control, the main goal of this study

is to seek theoretical foundations for the management of MSAs from the point of view of information evolution, and use them to develop practical sensor management approaches in different MSA-target scenarios. More specifically, we want to

1. Develop a generic approach to represent and maintain the evolution of target information for MSAs;

2. Seek a generic information metric to quantify the gain and loss of MSA control decisions, and use it to develop strategic guidelines for MSA management;

3. Apply the theoretical results above to the development of sensor management approaches in different MSA-target scenarios.

1.3 Contributions and the Outline of the Dissertation

The rest of this dissertation begins with a brief review of existing target-track maintenance approaches along with discussions on their feasibilities in different MSA applications in Chapter 2.

Considering the non-linear, terrain-dependent motion tendency of ground objects and the non-linear, non-analytic sensor model due to a limited FOV of an MSA, a generic approach for target track maintenance based on the Bayesian filtering method [66, 68, 69] and the Hospitability Map [70], the BF-HMap algorithm, is introduced in Chapter 3. An advanced version of this approach with much less computational and memory load using a particle filter, the so-called PF-HMap algorithm, is proposed. Due to the flexible scheme of Bayesian inference inherent in both algorithms, BF-HMap and PF-HMap are capable of handling intermittent and regional measurements caused by coverage and motion constraints on MSAs.

The study in target-track maintenance using particle filters is further extended to distributed mobile sensor networks (DMSN) in Chapter 4. One fundamental problem in sensor fusion in DMSN is the out-of-sequence measurements (OOSM) due to non-trivial communication delays. The *Universal Particle Filter* (UPF), which is capable of dealing with both in-sequence and out-of-sequence measurements is proposed. Some practical issues on the application of the UPF in real-world tracking problems, such as how to budget the computational load and the memory cost, is also discussed in Chapter 4.

Based on the research in target track maintenance outlined above, we model the *sensor management* problem of MSAs in an information-theoretic way in Chapter 5. In this study, we choose the entropy of the posterior probability density function, $p(x, t|Y(t))$, as a generic measure of information, where $Y(t)$ is the collection of all the measurements obtained by the MSAs up to time t. By studying the evolutions of entropy in both the linear and the non-linear cases, several key properties of the evolution of the entropy are identified. These properties are then utilized to model the *sensor management* problem in two cases of studies: target search and target surveillance, which are further studied in detail in Chapter 6 and Chapter 7, respectively.

Since the general goal of target search is to reduce the uncertainty of $x(t)$ and the entropy is essentially a measure of uncertainty, a target search problem can then be interpreted as a stabilization problem of the entropy. Based on this consideration, Chapter 6 presents a necessary condition and a sufficient condition (by means of the number of MSAs) to fulfill a non-escape search for a moving target. A cooperative search formation, called the *Progressively-Spiral-In* algorithm (PSI), is also introduced

in this chapter. Given a sufficient number of MSAs, the PSI algorithm can lead the MSA team to find the target in finite time.

In the second case of study, the *sensor management* problem for target surveillance is addressed in a multi-MSA-multi-target (MMMT) setup in Chapter 7. Based on the fact that the entropy of each target track is proportional to the sampling rate of effective measurements, the motion-planning problem here is modeled as an optimization problem, for which the objective is to minimize the average revisit time for each target. A cooperative motion-planning approach for multi-target surveillance by multiple non-holonomic MSAs is proposed. In this approach, the targets are divided into disjoint groups using a geometric clustering method. For each target group, a sub-optimal traversal motion plan is generated for an MSA to periodically update the track information of the targets in the shortest time. As the targets are moving around, on-line re-planning is conducted asynchronously on each MSA in a decentralized way. Target hand-off will be triggered among MSAs if a new target-grouping result is obtained.

As the summary of this dissertation, Chapter 8 concludes the work presented in this dissertation and discusses possible extensions of the proposed methodologies in future work.

CHAPTER 2

REPRESENTATION OF TARGET INFORMATION AND ITS MAINTENANCE

The subject of interest in many sensing problems is the kinematic state of one or multiple objects, which can be described by a stochastic process $x(t)$. The general task of a sensor system then is to estimate the evolution of $x(t)$ by extracting useful information of $x(t)$ from its observations (i.e. measurements). We call such an information processing procedure *target track maintenance* in this work to describe the process of accumulating target information from measurements.

In this chapter, we start from a generic representation of target information (section 2.1). Then, several representative approaches for *target track maintenance* are reviewed in section 2.2. Based on that, we will discuss the feasibilities of these approaches in different MSA-target scenarios in section 2.3.

2.1 Representation of Target Information

Without loss of generality, let us assume that the evolution of $x(t)$ is governed by the following stochastic differential equation:

$$dx(t) = f(x,t)dt + g(x,t)d\beta(t),\ t \geq t_0. \tag{2.1}$$

Here $f(\cdot)$ models the deterministic part dynamics of $x(t)$, which comes from the prior knowledge of how $x(t)$ evolves over time. $\beta(t)$ is the process noise, which stands for the random part of the dynamics of $x(t)$.

In the case of discrete-time system, (2.1) becomes

$$x(t_k) = F(x(t_{k-1}), t_k) + G(x(t_{k-1}), t_k)\beta(t_k),\ k > 0. \tag{2.2}$$

By moving around in the sensing space, the MSAs are taking measurements, which usually come at discrete time instants as follows:

$$y(t_k) = H(x(t_k), t_k) + \nu(t_k),\ k = 0, 1, 2, \ldots, \tag{2.3}$$

where $y(t_k) \in \mathcal{R}^m$ is the measurement taken at time instant t_k, $H(\cdot)$ is the sensor model and $\nu(t_k)$ denotes the measurement noise, which is often assumed to be Gaussian.

Theoretically, the posterior probability density function (PDF) $p(x, t|Y(t_k))$ provides us a complete online description of an MSA's knowledge of $x(t)$, where $Y(t_k)$ is the collection of measurements up to time instant t_k. Thus, we choose $p(x, t|Y(t_k))$ in this work as a generic representation of the target information. Clearly, one can get any statistics of $x(t)$ from $p(x, t|Y(t_k))$, such as the expectation (i.e. the minimum-mean-square-error (MMSE) estimate) and the variance. In most cases, we are only interested in the current or future status of $x(t)$ (i.e. $t \geq t_k$). In what follows, we shall call $p(x, t|Y(t_k))$ the *predictive PDF* if $t > t_k$, and the *estimative PDF* if $t = t_k$.

2.2 Maintaining the Target Information

Similar to $x(t)$, $p(x, t|Y(t_k))$ is a dynamic process as well, which evolves over time following (2.1). In the meantime, $p(x, t|Y(t_k))$ also changes as new measurements

are obtained by the MSAs. Depending on one's prior knowledge and assumptions on $f(x,t)$, $g(x,t)$, $\beta(t)$, $H(x,t)$ and $\nu(t)$, there have been a variety of theories and approaches to maintain the evolution of $p(x,t|Y(t_k))$, which are summarized briefly in the next few subsections.

2.2.1 Kalman Filter/Entended Kalman Filter (KF/EKF)

When the whole system is linear and the noise is Gaussian, we have $f(x,t) = f(t)x(t)$, $g(x,t) = g(t)x(t)$ in (2.1), $F(x(t_{k-1}),t_k) = F(t_k)x(t_{k-1})$, $G(x(t_{k-1}),t_k) = G(t_k)x(t_{k-1})$ in (2.2), and $H(x,t_k) = H(t_k)x(t_k)$ in (2.3). In this case, both the predictive and the estimative PDF are Gaussian, which can be concisely represented by their first and second-order moments (i.e. mean vectors $\hat{x}(t|t_k) = E[x(t)|Y(t_k)]$ and the co-variance matrices $P(t|t_k)$, $t \geq t_k$). With the help of further simplifications (such as independent noises and etc.), the Kalman filter is able to provide the MMSE prediction/estimate of $x(t)$ based on previous estimates as follows

$$\frac{d}{dt}\hat{x}(t) = f(t)\hat{x} + \bar{K}(t)[y(t) - h(t)\hat{x}(t)], \tag{2.4}$$

$$\bar{K}(t) = P(t)H^T(t)R^{-1}(t), \tag{2.5}$$

$$\frac{d}{dt}P(t) = f(t)P(t) + P(t)f^T(t) - \bar{K}(t)R(t)\bar{K}^T(t) + g(t)Q(t)g^T(t), \tag{2.6}$$

where $Q(t)$ and $R(t)$ denote the covariance matrices of $\beta(t)$ and $\nu(t)$, respectively.

The above is the continuous-time version of the Kalman filter. In practice, the discrete-time version of the Kalman filter is more commonly used, which runs recursively through a prediction stage and an estimation stage:

Prediction stage:

$$\begin{cases} \hat{x}(t_k|t_{k-1}) = F(t_k)\hat{x}(t_{k-1}|t_{k-1}), \\ P(t_k|t_{k-1}) = F(t_k)P(t_{k-1}|t_{k-1})F^T(t_k) + G(t_k)Q(t_k)F^T(t_k). \end{cases} \tag{2.7}$$

Estimation stage:

$$\begin{cases} \hat{x}(t_k|t_k) = \hat{x}(t_k|t_{k-1}) + K_{t_k}(y_k - H(t_k)\hat{x}(t_k|t_{k-1})), \\ K_{t_k} = P(t_k|t_{k-1})H^T(t_k)[H(t_k)P(t_k|t_{k-1})H^T(t_k) + R^{-1}(t_k)], \\ P(t_k|t_k) = [I - K_{t_k}H(t_k)]P(t_k|t_{k-1}). \end{cases} \tag{2.8}$$

The EKF approach is basically an extension of the Kalman filter to non-linear Gaussian systems, which are locally (i.e. at each step) linearizable. Usually, the first term in the Taylor expansion of the non-linear part of a system is used as its linear approximation. For instance, in the prediction stage of a discrete time EKF, $F(t_k)$ in (2.7) will be replaced by $\bar{F} = \{F_{i,j}\}_{n\times n}$, where $F_{i,j} = \frac{\partial^2 F}{\partial x_i x_j}$, and x_i denotes the i^{th} component of $x(t)$. For a more complete explanation of the KF/EKF method and their applications, please see [62].

2.2.2 Interacting Multiple Model (IMM) Filter

One of the major limitations of KF/EKF-based methods is that only a single analytic model $F(\cdot)$ can be consider in describing the evolution of $x(t)$, which is often inadequate in tracking maneuvering targets. In order to cope with this problem, Blom [63] introduced a multiple-model formulation, in which the target model (2.1) can switch among a finite set of analytic models $Mode = \{mode_1, mode_2, ...mode_M\}$. The transition between two modes is assumed to be governed by a Markovian chain $\{m_{t_0}, m_{t_1}, ..., m_{t_k}\}$, where $p(m_{t_k} = mode_l | m_{t_{k-1}} = mode_{l'}) = \pi_{l/l'}$, $l, l' = 1..m$. Here $\pi_{l/l'}$ is a pre-known probability of $x(t)$'s switching from model l to model l'. Usually, A Kalman filter (or EKF) is used for each model to track its mean vector ($\hat{x}^l(t_k|t_k)$) and covariance matrix ($P^l(t_k|t_k)$). In addition, a probability factor $\mu^l(t_k|t_k)$,$l = 1..M$, is also maintained to record the evolution of the probability of each model.

One fundamental problem of the multiple-model method is that the number of hypotheses (i.e. models) increases exponentially over time. To handle this problem,

Blom and Bar-Shalom [64] modified the original multi-model formulation and proposed the *Interacting Multiple Model* (IMM) algorithm, which is illustrated in Fig. 2.1.

The IMM algorithm also runs in a recursive way with each recursion consisting the following four steps:

1. To mix the previous estimates for each model: $\hat{x}^l(t_{k-1}|t_{k-1})$, $P^l(t_{k-1}|t_{k-1})$, and $\mu^l(t_{k-1}|t_{k-1})$:

$$\mu^l(t_k|t_{k-1}) = \sum_{l'}^{m} \pi_{l/l'} \mu^{l'}(t_{k-1}|t_{k-1}). \tag{2.9}$$

$$\bar{x}^l(t_{k-1}|t_{k-1}) = \frac{1}{\mu^l(t_k|t_{k-1})} \sum_{l'}^{m} \pi_{l/l'} \mu^{l'}(t_{k-1}|t_{k-1}) \hat{x}^{l'}(t_{k-1}|t_{k-1}). \tag{2.10}$$

$$\bar{P}^l(t_k|t_{k-1}) = \frac{1}{\mu^l(t_k|t_{k-1})} \sum_{l'}^{m} \pi_{l/l'} \mu^{l'}(t_{k-1}|t_{k-1}) [P^{l'}(t_{k-1}|t_{k-1}) + \Omega_{l/l'}], \tag{2.11}$$

   where

$$\Omega_{l|l'} = [\hat{x}^{l'}(t_{k-1}|t_{k-1}) - \bar{x}^l(t_{k-1}|t_{k-1})][\hat{x}^{l'}(t_{k-1}|t_{k-1}) - \bar{x}^l(t_{k-1}|t_{k-1})]^T. \tag{2.12}$$

2. To run the Kalman filter for each model using $\bar{x}^l(t_{k-1}|t_{k-1})$ in (2.10) and $P^l(t_{k-1}|t_{k-1})$ in(2.11) as the previous estimates of the mean vector and the co-variance matrix, respective. This step basically generates the new state estimate $\hat{x}^l(t_k|t_k)$ and covariance matrix $P^l(t_k|t_k)$ for each model.

3. To update the probability of each model $\mu^l(t_k|t_k)$:

$$\mu^l(t_k|t_k) = \frac{1}{C} \mu^l(t_k|t_{k-1}) |\det K_l(t_k)|^{-1/2} \exp\left(\frac{-1}{2} \delta_l(t_k)^T K_l(t_k)^{-1} \delta_l(t_k)\right), \tag{2.13}$$

where C is the normalization factor, $\delta_l(t_k) = y_l(t_k) - H^l(t_k)x(t_k)$, and $K_l(t_k)$ can be obtained in a similar way as in the second row of (2.8).

4. To generate the global estimate by combining the results from all the models:

$$\hat{x}(t_k|t_k) = \sum_{l}^{m} \mu^l(t_k|t_k)\hat{x}^l(t_k|t_k). \tag{2.14}$$

$$\begin{aligned} & P(t_k|t_k) \\ = & \sum_{l}^{m} \mu^l(t_k|t_k)\{P^l(t_k|t_k) + [\hat{x}(t_k|t_k) - \hat{x}^l(t_k|t_k)][\hat{x}(t_k|t_k) - \hat{x}^l(t_k|t_k)]^T\} \end{aligned} \tag{2.15}$$

The IMM filter has been widely used in applications where the target dynamics is very uncertain and non-deterministic [64]. It is worth noting that after merging the outcomes of all the models in the end of each recursion, the IMM filter only produces the global estimates of the first and second order moments of $p(x, t_k|Y(t_k)$, although multiple models are considered inside the IMM recursions.

2.2.3 Bayesian Filter (BF)

The Bayesian filtering method is a general tool to track non-linear system, in which the whole predictive/estimative PDF is maintained.

The predictive PDF can be obtained via the Chapman-Kolmogorov equation [66, 104]:

$$p(x, t|Y(t_k)) = \int p(x, t|\chi, t_k))p(\chi, t_k|Y(t_k))d\chi, \quad t > t_k, \tag{2.16}$$

where $p(x, t|\chi, t_k)$ is determined by (2.1).

The estimative PDF can be computed based on the Bayes' rule as follows:

$$p(x, t_k|Y(t_k)) = \frac{p(y(t_k)|x(t_k))p(x, t_k|Y(t_{k-1}))}{p(y(t_k)|Y(t_{k-1}))}, \tag{2.17}$$

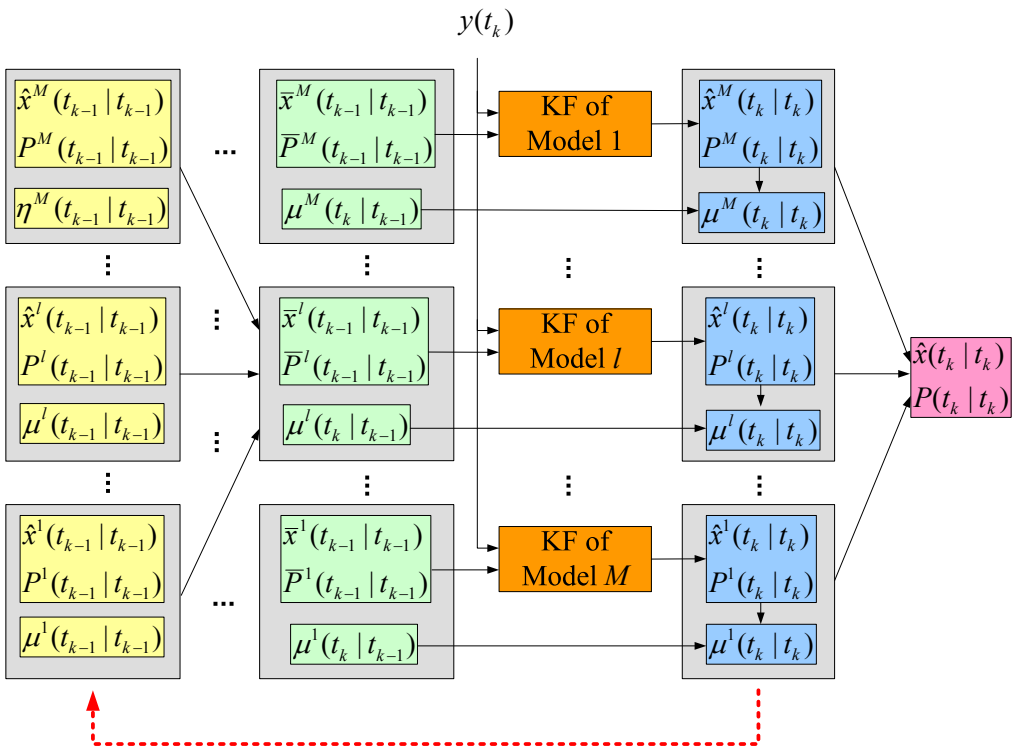


Figure 2.1: The IMM filter.

where

$$p(y(t_k)|Y(t_{k-1})) = \int p(y(t_k)|x)p(x|Y(t_{k-1}))dx, \tag{2.18}$$

The measurement PDF $p(y(t_k)|x(t_k))$ is determined by the sensor model, i.e. $H(\cdot)$ and $\nu(t)$ in (2.3).

Theoretically, the *Bayesian Filtering* method outlined above applies to any target and sensor models. In particular, a BF will reduce to a standard Kalman filter with the linear-and-Gaussian assumption. Due to the difficulty in obtaining an analytic

form of $p(x,t|Y(t_k))$ for a non-linear, non-Gaussian system. A Bayesian filter is usually implemented in a discrete-space way. The most straightforward approach is to decompose the state space into discrete grids. The finite difference method or other similar methods [68] are often used in to computer the track PDF numerically. As a result, a large memory space is needed to store an accurate grid-based PDF. Meanwhile, it is also computationally costly (and sometimes even impossible) to update $p(x(t_k)|x(t_{k-1}))$ and $p(x(t_k)|x(t_{k-1}), y(t_k))$ in every grid in practice. In [69], Challa et al. prune the grid-space by only considering the grids that are inside the 2σ-window of the track PDF. A more comprehensive survey on the general Bayesian filter can be found in [66–68].

2.2.4 Particle Filter (PF)

The *particle filtering* method is essentially an approximation of the *Bayesian filtering* method, which approximates the posterior PDF of a target track by a set of samples and their weights $\{X^i_{t_1:t_k}, w_i(t_k)\}_{i=1..N}$:

$$p(X(t_k)|Y(t_{k'})) \approx \sum_{i=1}^{N} w_i(t_k)\delta(X(t_k) - X^i_{t_1:t_k}), \quad k \geq k', \tag{2.19}$$

where $X(t_k) = \{x(t_0), x(t_1), ..., x(t_k)\}$; $X^i_{t_1:t_k} = \{x^i_{t_0}, x^i_{t_1}, ..., x^i_{t_k}\}$; $\delta(X) = 0$ for $\forall X \neq 0$ and $\int_{-\infty}^{+\infty} \delta(X)dX = 0$. Note that $\sum_{i=1}^{N} w_i(t_k) = 1$.

By taking integrals on both sides of (4.1) with respect to $X(t_{k-1})$, we can also get an approximation of the marginal PDF $p(x(t_k)|Y(t_{k'}))$ as:

$$p(x(t_k)|Y(t_{k'})) \approx \sum_{i=1}^{N} w^i_{t_k}\delta(x(t_k) - x^i_{t_k}). \tag{2.20}$$

At each time instant t_k, a set of new samples $X^i_{t_k} = \{x^i_{t_k}\}_{i=1..N}$ are drawn from a proposal $q(\cdot)$ called the *importance density*:

$$X^i_{t_k} \sim q(X(t_k)|Y(t_k)), \qquad i = 1..N. \tag{2.21}$$

In order to guarantee that (4.1) converges to the true PDF as $N \to \infty$, the weights $w^i_{t_k}$ should satisfy:

$$w^i_{t_k} \propto \frac{p(X^i_{t_1:t_k}|Y(t_k))}{q(X^i_{t_1:t_k}|Y(t_k))}, \quad i = 1..N. \tag{2.22}$$

Based on the Bayes' rule, we have:

$$\begin{aligned} p(X^i_{t_1:t_k}|Y(t_k)) &= \frac{p(y(t_k)|x^i_{t_k})p(x^i_{t_k}|X^i_{t_1:t_{k-1}})}{p(y(t_k)|Y(t_{k-1)})} p(X^i_{t_1:t_{k-1}}|Y(t_{k-1})) \\ &\propto p(y(t_k)|x^i_{t_k})p(x(t_k)|X^i_{t_1:t_{k-1}})p(X^i_{t_1:t_{k-1}}|Y(t_{k-1})) \end{aligned} \tag{2.23}$$

If we further choose the *importance density* in such a way that

$$q(X^i_{t_1:t_k}|Y(t_k)) = q(x^i_{t_k}|X^i_{t_1:t_{k-1}}, Y(t_k))q(X^i_{t_1:t_{k-1}}|Y(t_{k-1})), \tag{2.24}$$

we can rewritten (2.22) as

$$w^i_{t_k} \propto w^i_{t_{k-1}} \frac{p(y(t_k)|x(t_k))p(x^i_{t_k}|X^i_{t_1:t_{k-1}})}{q(x^i_{t_k}|X^i_{t_1:t_{k-1}}, Y(t_{k-1}))} \tag{2.25}$$

Therefore, we can update the samples and their weights in a recursive way as the following (at time instant t_k):

1. Draw new samples: $x^i_{t_k} \sim q(x(t_k)|X^i_{t_1:t_{k-1}}, Y(t_k))$, $i = 1..N$;

2. Update the new weights $\{w^i_{t_k}\}_{i=1..N}$ according to (3.24).

The above describes a generic *particle filter* approach for target tracking. Theoretically, the *importance density* can be arbitrarily chosen as long as (2.24) holds.

However, an appropriately selected *importance density* is very crucial for the tracking performance. Another common problem with PF-based methods is that after a few iterations, most particles will have negligible weights except for a dominant one. Doucet [102] has proved that the variance of the weights monotonically increases over time. A commonly used measure of the degeneracy of a PF is the *estimate of the effective sample size* $\widehat{N}_{eff}$:

$$\widehat{N}_{eff}(t_k) = \frac{1}{\sum_{i=1}^{N}(w_{t_k}^i)^2} \leq N. \tag{2.26}$$

Note that the equality holds if and only if $w_{t_k}^i = \frac{1}{N}$ for all $i = 1..N$.

Doucet [102] has also proved that the optimal *importance density* function $q^*(x_{t_k}^i)$ that minimize $\widehat{N}_{eff}$ is:

$$q^*(x(t_k)|X_{t_1:t_{k-1}}^i, Y(t_k)) = p(y(t_k)|X_{t_1:t_{k-1}}^i, Y(t_k)), \tag{2.27}$$

Unfortunately, it is very difficult to draw samples from the optimal *importance density* $q^*(\cdot)$, except for some simple cases [104]. One practical choice of *importance density* that is commonly used in applications is the prior:

$$q(x(t_k)|X_{t_1:t_{k-1}}^i, Y(t_k)) = p(x(t_k)|X_{t_1:t_{k-1}}^i), \tag{2.28}$$

which leads to

$$w_{t_k}^i \propto= p(y(t_k)|x_{t_k}^i) w_{t_{k-1}}^i. \tag{2.29}$$

For the target system described in (2.2), if we further assume that the process noise (i.e. $\beta(t)$) is zero-mean and white Gaussian, we have

$$p(x(t_k)|X_{t_1:t_{k-1}}^i) = p(x(t_k)|x_{t_{k-1}}^i), \tag{2.30}$$

and

$$x(t_k) \sim N(F(x^i_{t_{k-1}}, t_{k-1}), G^T(t_{k-1}))Q(t_k)G(t_{k-1})), \tag{2.31}$$

where $Q(t_k)$ is the co-variance matrix of $\beta(t_k)$. A more comprehensive review of PF-based methods as its variations can be found in [102, 104].

2.3 Target Track Maintenance by MSAs

The choice of target track maintenance approaches for in a specific problem is basically determined by the characteristics of the target model (i.e. $f(\cdot)$ and $g(\cdot)$ in (2.1)) and the sensor model (i.e. $H(\cdot)$ in (2.3)). In this section, we take the target search problem and the target tracking problem as two typical MSA-target scenarios to study how these factors affect the development of the corresponding track maintenance method.

2.3.1 Track Maintenance in Target Search

In the case of target search, the track PDF usually spreads all over the search space most of the time. Thus, the field of view (FOV) of an MSA can only cover a small part of the track PDF. Before the target is detected, all an MSA can get are non-detection reports, which leads to a highly non-linear measurement function $H(\cdot)$ (see Chapter 3 for details). To maintain a target track base on this type of sensor information, a KF-based method (including the IMM filter) is obviously inadequate. A BF or PF-based approach is probably the only choice in this case. In fact, a great deal of research work in the area of target search assumes that the target information is stored in a map-like structure [73, 75, 79, 82, 83], which is similar to the track PDF maintained by a discrete-space version of the Bayesian filter.

2.3.2 Track Maintenance in Target Tracking

In a conventional setup for target tracking, the measurement is always considered as a given, and each recursion of prediction-and-estimation is activated only if there is a positive sensor reading. The underline assumption is that the sensor has a global coverage of the whole field of interest. With this assumption, the measurements from many sensor system are linear or nearly-linear. Meanwhile, a global coverage also allows the user to assume a high measurement sampling rate, which makes many target models locally linearizable (e.g. constant velocity, constant acceleration, constant left/right-turn). As a result, the KF/EKF, the IMM filter and their variations have been the most dominant tracking methods in the last several decades.

Recently, the BF and its Monte-Carlo alternative, the PF, become more and more popular, especially in map-based or vision-based robot localization problems [103]. The main reason is probably the highly non-linear measurement function $H(\cdot)$ that appears in many robot localization problems, which can not be handled by a KF-like scheme. In the case of MSAs, such a highly non-linear of $H(\cdot)$ is often the case as well. Because of a limited field of view (FOV) and physical motion constraints (e.g. a minimum turning radius) of an MSA, a target can be in and out of surveillance frequently. As a result, the measurements become *intermittent* and *regional*. Here by *intermittent* we mean that there can be a significant time duration between consecutive observations of a target, and the term *regional* refers to the fact that a sensor can only provide the target information inside its FOV. Such *intermittent* and *regional* measurements make both the target model and the sensor model highly nonlinear. In Chapter 3, we will address this problem in more detail and introduce a

pair of track maintenance methods based on the Bayesian filter and the particle filter specifically for MSAs to handle *intermittent* and *regional* measurements.

CHAPTER 3

TARGET TRACK MAINTENANCE WITH REGIONAL AND INTERMITTENT MEASUREMENTS

3.1 Introduction

Many prototype MSAs, such as Uninhabited Air Vehicles (UAVs), are characterized by their spatially-restricted sensing capability, such as a limited field of view (FOV) and physical motion constraints. As a result, the measurements obtained by MSAs become *intermittent* and *regional*, which brings new challenges to the track maintenance problem. Because of a limited FOV, targets are not guaranteed to be observed every time even with a perfect sensor. In order to attain good measurements, a mobile sensor has to adjust its position/pose on-line according to its predictions of the target state. The quality of measurements (as well as that of future estimates) relies on the effectiveness of the predictive track information that an MSA keeps. Thus, maintaining predictions before taking measurements is as important as generating estimates when measurements are available.

Since measurements are not involved at the prediction stage, the quality of predictive track information is mainly determined by how effectively the target dynamics is modeled. It has been well known that the motion of a real-world maneuvering

target is non-linear, multi-modal, and environment-dependent. In order to accommodate such complex target systems, a non-linear and non-analytic dynamics is often needed. Meanwhile, every measurement obtained by a mobile sensor is conditioned by its FOV. As a result, the sensor model becomes a complex function as well, which should treat the information inside and outside the FOV in different ways. In a word, the nature of *intermittent* and *regional* measurements requires that the desired track maintenance algorithm should have the capability of dealing with both non-linear, environment-dependent target dynamics (i.e. $f(x,t)$ in (2.1)) and non-analytic sensor models (i.e. $H(x,t)$ in (2.3)).

The requirement of the capability to handle non-linear, non-analytic $f(x,t)$ and $H(x,t)$ first limits the applicability of the Kalman filter (KF) or the extended Kalman filter (EKF) in the track maintenance problem addressed here. Although the IMM filter has been proved [64] to be able to tracking maneuvering targets with multi-modal motion behavior, it is difficult to apply this method directly to the MSA-target scenario here. The reason is twofold. First, the IMM algorithm is an estimation-centric method, in which the prediction stage only runs right before a new measurement comes. On the other hand, the nature of the IMM formulation requires a relatively high measurement sampling rate. Thus, when there is a long time interval between two consecutive measurements, estimation stages, the predictions generated by multiple KFs/EKFs in the IMM approach degrades quickly over time. Second, each KF/EKF in the IMM approach requires a linear/linearizable sensor model ($H(x,t)$). As a result, non-detection information cannot be utilized directly by the IMM method, which will significantly degenerate the performance of track maintenance.

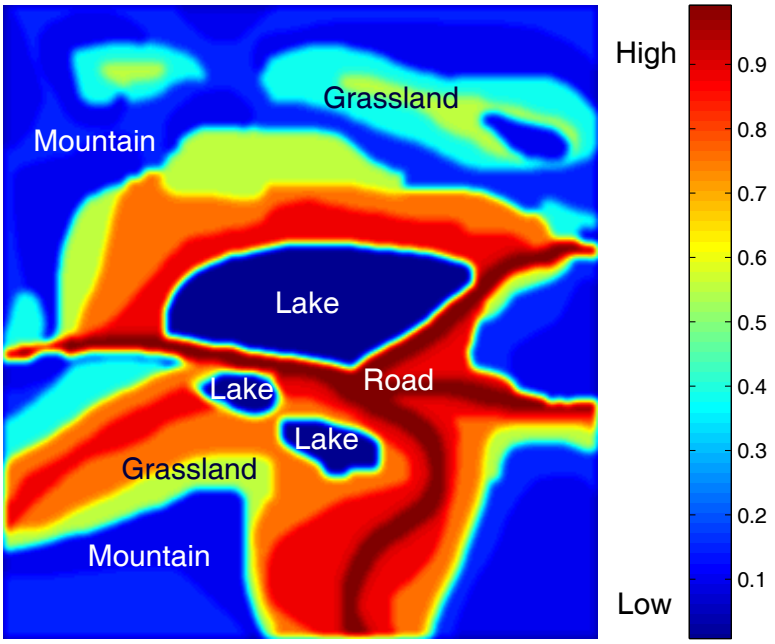


Figure 3.1: An example of an HMap.

A more general way to represent complex target model is the Bayesian filter (BF) [66]. Since the Bayesian filter basically maintains the entire PDF of the target state instead of the first and second order moments, it can provide a complete package of predictive track information for mobile sensors to make their motion decisions. The flexible scheme of Bayesian inference also makes it possible to exploit non-analytic prior knowledge of the target system, the sensor, or the environment to improve both the prediction and estimation performance. In [70], Layne et al. first use the concept of a *Hospitability Map* (HMap) and combine it with a Bayesian filter to maintain predictive track information of an out-of-surveillance target. The HMap is defined in a gridded spatial domain as $hosp(x)$, which is used to describe the effect of different terrain surfaces on the localization and mobility of target objects. Fig. 3.1 shows an

example of an HMap for ground vehicles. In this example, the roads have the highest (1.0) *hospitability* to the vehicle, and the lakes have the lowest (0.0) *hospitability* to the vehicle.

In this chapter, we adapt the HMap concept introduced [70] and propose a generic method for target track maintenance based on the Bayesian filtering method, which is called the BF-HMap algorithm. The main part of the BF-HMap algorithm has been published in [89]. Different from traditional tracking approaches both the prediction stage of the BF-HMap approach consists of an analytic part and and a non-analytic part, which allows itself to incorporate non-analytic environmental information with standard kinematic models. In the meantime, each *measurement* from the sensor are extended to a *measurement report*, which leads to a new non-linear sensor model even if the measurement itself is linear. By doing so, non-detection information caused by a limited sensor footprint is taken into account at the estimation stage. In addition, the possibility of false alarm and miss detection is also consider in the Bf-HMap algorithm.

In order to reduce the memory and computational load inherent in BF-based methods, a new target track maintenance algorithm (PF-HMap) based on particle filtering and the HMap is also introduced. Although the particle filter is essentially an approximation of the Bayesian filter, the PF-HMap algorithm is not merely an approximation of the BF-HMap algorithm. The incorporation of the HMap requires a continuous predictive PDF of the target system to generate particle sets, which is not available in the context of particle filters. In the PF-HMap approach, a local Monte Carlo scheme is used to handle this problem. Instead of drawing new samples from the predictive PDF directly, a set of sample candidates is generated first. Then, the new

sample for each particle sequence is drawn from this finite set of sample candidates. With this local Monte Carlo method, the PF-HMap method does not require a single, analytic $f(x,t)$ as the BF-HMap approach does. Thus, more complex target models such as the IMM scheme can be smoothly absorbed into the prediction stage of the PF-HMap method to improve its performance. It is worth noting that the prediction stage of the IMM version of the PF-HMap algorithm can also be considered as an extension of the IMM-Bootstrap filter proposed in [105]. In fact, the prediction stage of the PF-HMap algorithm will reduce to that of the IMM-Bootstrap filter when the HMap is a constant across the field of interest.

For the purpose of comparing the proposed methods with other tracking approaches, several methodological issues in performance evaluation in non-linear track maintenance problems are discussed. Since the combination of the mean and the covariance is not a sufficient representation of a general non-Gaussian system, the commonly used mean square error (MSE) criterion becomes inadequate as the performance metric in this case. In this chapter, a new performance evaluation method based on *relative entropy* (i.e. Kullback-Leibler Distance) [106] is proposed, which can be regarded as a consentient criterion to compare the performance of different track maintenance approaches for both Gaussian and non-Gaussian target systems.

The rest of the chapter is organized as follows: Section II and III give the details of the BF-HMap and PF-HMap algorithm, respectively. Then, the *relative-entropy*-based performance metric is introduced in Section IV, followed by experimental results and performance analysis in Section V. Finally, conclusions are given in Section VI.

3.2 The BF-HMap Algorithm

Similar to most tracking methods, the BF-HMap algorithm runs recursively through a *prediction stage* (or the *time evolution* stage) and a *estimation stage* (or the *measurement evolution* stage), as illustrated in Fig. 3.2.

3.2.1 The Prediction Stage

The *prediction stage* here cosists of two parts: the analytic part and the non-analytic part.

The Analytic Part:

The analytic part is a primitive model for the target, which is similar to most other tracking methods:

$$\dot{x}(t) = \bar{f}(x(t), t) + g(x(t), t)\beta(t), \tag{3.1}$$

where $x(t) = [x_1(t), x_2(t), ..., x_N(t)] \in \mathbf{R}^N$ stands for the vector of target state at time t; $\bar{f}(\cdot)$ represents all of one's prior knowledge on the dynamics $x(t)$ that can be explicitly by a mathematic function; $\beta(t)$ denotes the process noise.

Derivations in [66] [68] have shown that, with appropriate assumptions (e.g. the continuity and differentiability of $p(x, t|Y(t_k))$, a Brownian process noise $\beta(t)$), the time evolution of the posterior PDF $p(x, t|Y(t_k))$ following (3.1) can be described by the following Fokker-Planck equation (FPE):

$$\frac{\partial p}{\partial t} = -\sum_{i=1}^{n} \frac{\partial(\bar{f}_i p)}{\partial x_i} + \frac{1}{2} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \frac{\partial[(gQg^T)_{i_1 i_2} p]}{\partial x_{i_1} \partial x_{i_2}}, t_k < t < t_{k+1} \tag{3.2}$$

where $\bar{f}_i$ is the i^{th} row of $\bar{f}(x, t)$; p is the short notation for $p(x, t|Y(t_k))$; $Q(t)$ stands for the co-variance matrix of $\beta(t)$.

Theoretically, the predictive PDF, $p(x,t|Y(t_k)), t > t_k$, can be obtained by solving the FPE above. Although there is no general analytic solution to (3.2), we can get the numerical solution using the finite difference method [68] or similar tools.

In the rest of this chapter, we will adapt the commonly used 2-D constant velocity model (CV) as an example of the primitive model (3.1) for the analytic part of $x(t)$. Under the 2-D CV assumption, we have

$$x(t) = [\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{y}(t), \dot{\mathbf{y}}(t)]^T, \tag{3.3}$$

$$\bar{f}(x,t) = \left[\dot{\mathbf{x}}(t), 0, \dot{\mathbf{y}}(t), 0\right]^T, \tag{3.4}$$

$$g(x,t) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}^T, \tag{3.5}$$

and

$$Q(t) = \begin{bmatrix} q_{\dot{\mathbf{x}}}(t) & 0 \\ 0 & q_{\dot{\mathbf{y}}}(t) \end{bmatrix}, \tag{3.6}$$

where $q_{\dot{\mathbf{x}}}(t)$ and $q_{\dot{\mathbf{y}}}(t)$ are the variances of the acceleration in $\mathbf{x}$ and $\mathbf{y}$ direction, respectively.

Hence, the resulting FPE becomes:

$$\frac{\partial \bar{p}}{\partial t} = -\dot{\mathbf{x}}\frac{\partial \bar{p}}{\partial \mathbf{x}} - \dot{\mathbf{y}}\frac{\partial \bar{p}}{\partial \mathbf{y}} + \frac{q_{\dot{\mathbf{x}}}}{2}\frac{\partial^2 \bar{p}}{\partial \dot{\mathbf{x}}^2} + \frac{q_{\dot{\mathbf{y}}}}{2}\frac{\partial^2 \bar{p}}{\partial \dot{\mathbf{y}}^2}. \tag{3.7}$$

The Non-analytic Part:

At this stage, the non-analytic information of the target's motion is taken into account. A typical example of such non-analytic information is the terrain-dependent motion tendency of a ground target, which is described by the *Hospitability Map*. In

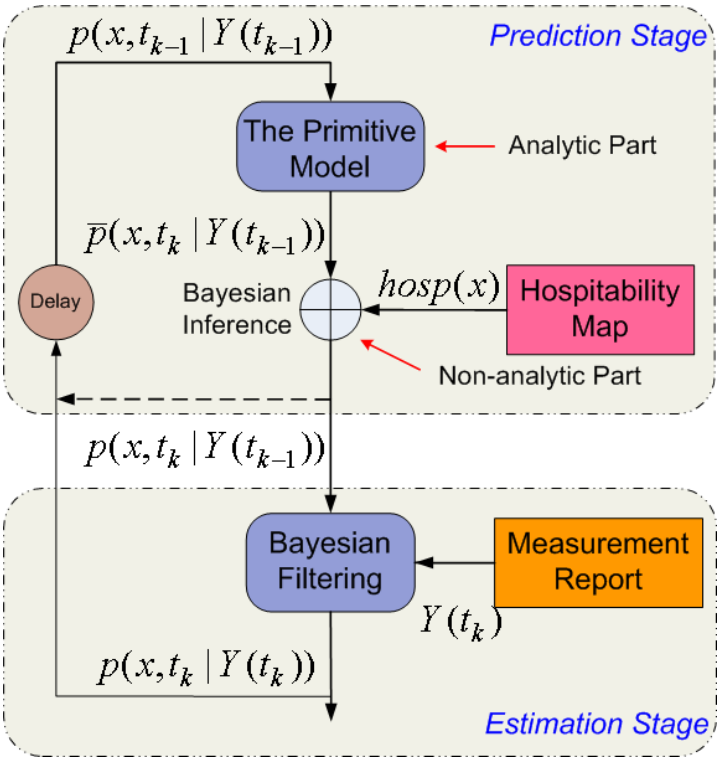


Figure 3.2: Overview of the BF-HMap algorithm.

this work, this type of non-analytic prior knowledge is absorbed into the *predictive PDF* through Bayesian inference as follows:

$$p(x,t|Y(t_k)) = \frac{1}{C_h}\bar{p}(x,t|Y(t_k))hosp(x),\ t > t_k, \tag{3.8}$$

where $C_h = \int p(x,t|Y(t_k))hosp(x)dx$, which is the normalization factor; $hosp(x)$ is the *hospitability map*.

3.2.2 The Estimation Stage

In traditional track maintenance approaches, the estimation stage is involved only if a new measurement $y(t_k)$ is obtained and the sensor coverage is assume to be global.

In reality, most sensors can only provide *regional* observations, which means each one can only validate information from the inside of its FOV. If a global coverage is assumed, non-detection reports are usually be discarded automatically or be treated as a miss-detection event. Since a global coverage is often not true in the case of MSAs, a non-detection reports from an MSA is not merely a miss-detection event any more, which is also an important measurement result and should be utilized to update the track PDF. Based on this consideration, we re-define the measurement in a more general way as $y(t_k) \in= \bigcup\{\emptyset, \mathcal{R}^m\}$, in which $y(t_k) = \emptyset$ represents the non-detection event.

Meanwhile, since the sensors are not perfect in reality [80], there are non-zero possibilities of false alarm (P_0) and miss detection (P_1). Thus, considering the report event only, the posterior probability of the measurement report $y(t_k)$, given the true state of the target (x), is

$$p(y(t_k) \neq \emptyset | x) = \begin{cases} P_0, & \text{if } x \notin F; \\ 1 - P_1, & \text{if } x \in F. \end{cases} \tag{3.9}$$

and

$$p(y(t_k) = \emptyset | x) = \begin{cases} P_1, & \text{if } x \in F; \\ 1 - P_0, & \text{if } x \notin F. \end{cases} \tag{3.10}$$

Here F is the FOV of an MSA, $x \in F$ means that the positional components $(\mathbf{x}, \mathbf{y})$ of target state x is inside F.

If we further assume that a false alarm is uniformly distributed inside the FOV of the sensor, we can then get the measurement probability density function as follows:

$$p(y(t_k)|x) = \begin{cases} \frac{P_0}{A_F}, & y(t_k) \neq \emptyset \text{ and } x \notin F; \\ (1 - P_1)p^*(y(t_k)|x), & y(t_k) \neq \emptyset \text{ and } x \in F; \\ P_1, & \text{if } y(t_k) = \emptyset \text{ and } x \in F; \\ 1 - P_0, & \text{if } y(t_k) = \emptyset \text{ and } x \notin F; \end{cases} \tag{3.11}$$

where $p^*(y(t_k)|x)$ is the probability density function of a non-empty measurement $y(t_k)$ given the true state of the target $x(t_k) = x$, which is basically the distribution of the measurement noise $\nu(t_k)$.

By plugging (3.11) into the measurement update equation in the Bayesian filtering method (2.17), we now have a new measurement update equation as:

$$p(x,t_k|Y(t_k)) = \begin{cases} \frac{P_0}{A_F C_D} p(x,t_k|Y(t_{k-1})), & \text{if } y(t_k) \neq \emptyset \text{ and } x \notin F; \\ \frac{(1-P_1)}{C_D} p(x,t_k|Y(t_{k-1}) p(y(t_k)|x), & \text{if } y(t_k) \neq \emptyset \text{ and } x \in F; \\ \frac{P_1}{C_U} p(x,t_k|Y(t_{k-1})), & \text{if } y(t_k) = \emptyset \text{ and } x \in F; \\ \frac{1-P_0}{C_U} p(x,t_k|Y(t_{k-1})), & \text{if } y(t_k) = \emptyset \text{ and } x \notin F; \end{cases} \tag{3.12}$$

where

$$\begin{aligned} C_D &= \frac{P_0}{A_F}[1 - \int_F p(x,t_k|Y(t_{k-1}))dx] + (1-P_1)\int_F p(x,t_k|Y(t_{k-1}))p^*(y(t_k)|x)dx, \\ C_U &= P_1 \int_F p(x,t_k|Y(t_{k-1}))dx + (1-P_0)[1 - \int_F p(x,t_k|Y(t_{k-1}))dx]. \end{aligned} \tag{3.13}$$

Remark 3.2.1: It is worth noting that (3.12) reduces to the classical Bayesian estimation equation when the sensor coverage is large enough that the whole track PDF, $p(x,t_k|Y(t_{k-1}))$, falls inside the footprint F, which coincides with the assumption made by most classical measurement updating methods.

3.2.3 Extension to the MMMT case

In this subsection, we extend our derivations above to the general multi-MSA-multi-target sensor fusion problem. At this point, we simplify the situation by assuming that the MSA's can communicate with each other and share their measurement reports with each other as global information.

Define $Y^i(t_k) = \{Y^i(t_{k-1}), y^i(t_k)\}$, where $y^i(t_k) = \{y^i_j(t_k)\}_{j=1..n}$ denotes the detection report of the target i from the n different sensor agents at time instant t_k. Under the assumption that the detection of each sensor agent is independent of each other,

we have

$$p(Y^i(t_k)|X^i(t_k)) = \prod_{j=1}^{n} p(y_j^i(t_k)|X^i(t_k)), \tag{3.14}$$

given the true state of target i, $X^i(t_k)$. If there is no measurement report from MSA j at time t_k, we can just set $p(y_j^i(t_k)|X^i(t_k))$ to be a positive constant.

Combining (2.17) and (3.14), we obtain the measurement evolution equation for the MMMT case as:

$$\begin{aligned} p(x^i, t_k|Y^i(t_k)) &= \frac{p(X^i, t_k|Y^i(t_{k-1}))p(y^i(t_k)|x^i)}{\int p(x, t_k|Y^i(t_{k-1})]p[y^i(t_k)|x)dx} \\ &= \frac{p(x^i, t_k|Y^i(t_{k-1}))\prod_{j=1}^{n} p(y_j^i(t_k)|x^i)}{\int p(x, t_k|Y^i(t_{k-1}))\prod_{j=1}^{n} p(y_j^i(t_k)|x)dx}. \end{aligned} \tag{3.15}$$

3.2.4 Discrete-space-discrete-time (DSDT) Implementation of the BF-HMap Algorithm

Practically, we can only compute the whole target PDF in a discrete space and discrete time version, which is derived as the following.

The Prediction Equation

Using the explicit forward-time-central-space finite difference method [68] [70], we get the following numerical prediction equation for a target from (3.7):

$$\begin{aligned} \bar{P}^-(k+1, s, u, v, w) = (1 - \frac{\Delta t q_{\dot{x}}}{\Delta \dot{x}^2} - \frac{\Delta t q_{\dot{y}}}{\Delta \dot{y}^2}) P_i(k, s, u, v, w) \\ -\frac{\Delta t u \Delta \dot{x}}{2\Delta x}[\bar{P}(k, s+1, u, v, w) - \bar{P}(k, s-1, u, v, w)] \\ -\frac{\Delta t w \Delta \dot{x}}{2\Delta y}[\bar{P}(k, s, u, v+1, w) - \bar{P}(k, s, u, v-1, w)] \\ +\frac{\Delta t q_{\dot{x}}}{2\Delta \dot{x}^2}[\bar{P}(k, s, u+1, v, w) + \bar{P}(k, s, u-1, v, w)] \\ +\frac{\Delta t q_{\dot{y}}}{2\Delta \dot{y}^2}[\bar{P}(k, s, u, v, w+1) + \bar{P}(k, s, u, v, w-1)] \end{aligned} \tag{3.16}$$

where

$$\bar{P}(k,s,u,v,w) = p(x,t_k|Y(t_{k-1})) = [s\Delta x, u\Delta\dot{x}, v\Delta y, \Delta\dot{y}]^T|Y(t_k)). \tag{3.17}$$

Here $\Delta x, \Delta\dot{x}, \Delta y, and \Delta\dot{y}$ are the sizes of the grid. Note that according to [69], to guarantee the finite difference method stable, we need:

$$0 < \Delta t \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta\dot{x}^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta\dot{y}^2}\right) \leq \frac{1}{2}. \tag{3.18}$$

Similar to (3.8), we further incorporate the time evolution with the *hospitability map* and modify the prediction as:

$$P^-(k+1,s,u,v,w) = \frac{1}{C_h}\bar{P}^-(k+1,s,u,v,w)hosp(s,u,v,w)$$

where C_h is the normalization factor, and $hosp(s,u,v,w)$ is the discretized *hospitability map*.

The Estimation Equation

When new measurements are received, the DSDT version of measurement evolution equation (3.15) can be derived as:

$$P_i^+(k+1,s,u,v,w) = \frac{1}{C}P^-(k+1,s,u,v,w)\prod_{j=1}^{n} p\left(y_j^i(t_{k+1})|k+1,s,u,v,w\right), \tag{3.19}$$

where C is the normalization factor, and

$$p\left(y_j^i(t_{k+1})|k+1,s,u,v,w\right) = p\left(y_j^i(t_{k+1})|x^i(t_{k+1}) = [s\Delta x, u\Delta\dot{x}, v\Delta y, w\Delta\dot{y}]^T\right), \tag{3.20}$$

which can be determined by the characteristic of the measurement noise in (2.3).

3.2.5 An Illustrative Example

Fig. 3.3 shows one experiment that we simulated to demonstrate how our algorithm works in maintaining target tracks, in which we have one moving target and

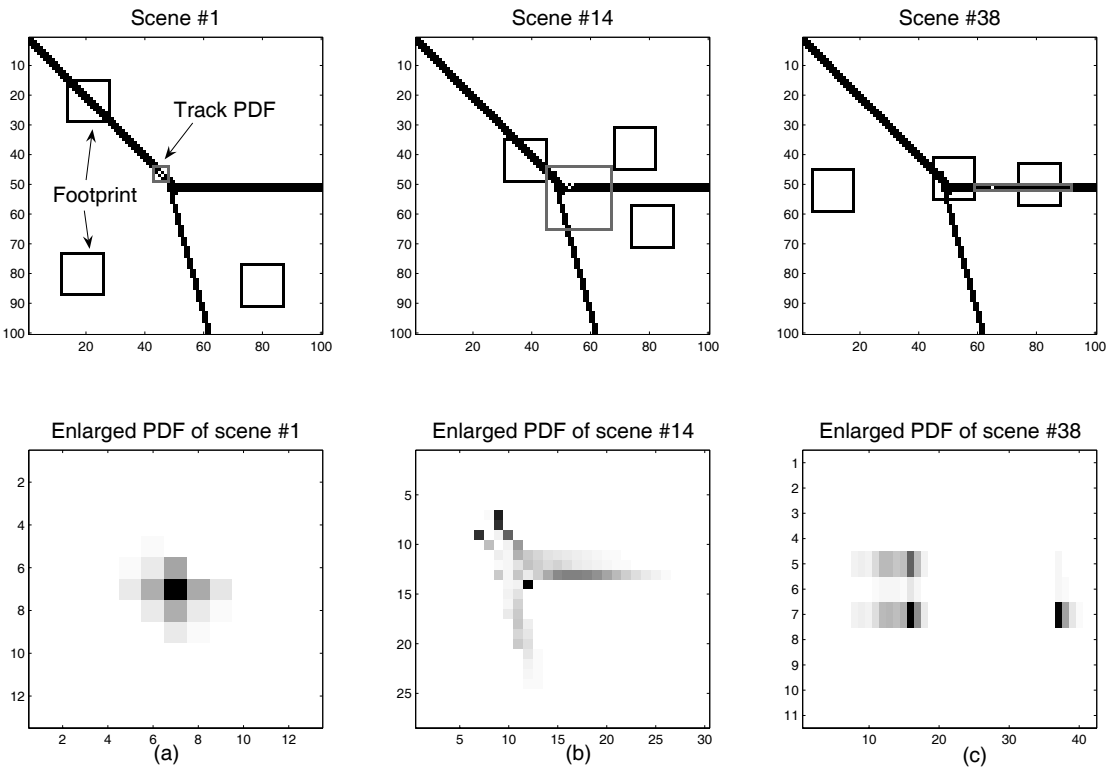


Figure 3.3: Track maintenance for a vehicle in a road network by 3 UAVs using the BF-HMap algorithm

three UAVs. The field of interest is a road network. Here we assume that the target always moves on the road. In other words, the *hospitability map* of this scenario is simplified as $hosp(x, y, \dot{x}, \dot{y}) = 1$ if (x, y) is on the road, and 0 otherwise. In each sence, the three squares emulate the footprints of the UAVs, and the small circle indicates the target. The size-variant rectangle around the target stands for the effective range of the majority of the PDF. The PDF is enlarged and shown beside each scene. The darker the pixels in the PDF figure, the larger the value of the PDF at that position. Fig. 3.3.a shows the initial situation. Before any UAV re-capture the target, its track

PDF keeps growing along the direction of the road (Fig. 3.3.b). When two UAV's approach the target (Fig. 3.3.c), the PDF of the target track converges even though it has not been detected by either UAV, because the "undetected" reports from the UAV's also help to narrow down the possible states of the target.

3.3 The PF-HMap Algorithm

The basic idea of the PF-HMap algorithm is to replace the Bayesian filter in the BF-HMap algorithm by a particle filter, which approximates the track PDF by a set of sample sequences and their weights $\{X^i_{t_1:t_k}, w^i_{t_k}\}_{i=1..N}$:

$$p(X_{t_1:t_k}|Y_{t_1:t_k}) \approx \sum_{i=1}^{N} w^i_{t_k} \delta(X_{t_1:t_k} - X^i_{t_1:t_k}), \tag{3.21}$$

where $X^i_{t_1:t_k} = \{x^i_{t_1}, ..., x^i_{t_k}\}$ is a simulated sequence of $x(t)$ from t_1 to t_k; $\delta(x) = 0$ for $\forall x \neq 0$ and $\int_{-\infty}^{+\infty} \delta(x)dx = 1$; $\sum_{i=1}^{N} w^i_{t_k} = 1$.

Similar to other tracking approaches, the PF-HMap algorithm also consists of a prediction stage (i.e. drawing new samples) and a estimation stage (i.e. updating weights), which will be explained in detail in the following two sub-sections.

3.3.1 The Prediction Stage

At the prediction stage, a new set of particles are drawn from a user defined distribution $q(\cdot)$, called the *importance density* [102]. A common choice of $q(\cdot)$ is $p(x, t_k|X_{t_1:t_{k-1}})$, which is the predictive PDF of the target state from t_{k-1} to t_k. Unfortunately, because of the non-analytic part (i.e. the HMap) of the prediction stage, the corresponding $p(x, t_k|X_{t_1:t_{k-1}})$ is a non-analytic distribution in general even if the primitive model $\bar{p}(x, t_k|X_{t_1:t_{k-1}})$ is linear and Gaussian. Thus, it is difficult

to draw new samples from the predictive PDF directly. In this work, we use the following local Monte Carlo method to realize the sampling procedure.

Drawing New Samples by Local Monte Carlo

The basic idea of this approach is to generate a set of sample candidates $\{\tau_j^i\}_{j=1..N_l}$ from the primitive model $\bar{p}(x, t_k|X^i_{t_1:t_{k-1}})$ first, and then approximate $p(x, t_k|X^i_{t_1:t_{k-1}})$ by:

$$p(x, t_k|X^i_{t_1:t_{k-1}}) \approx \sum_{j=N_l}^{N_l} w_j^i(t_k)\delta(x^- \tau_j^i(t_k)), \tag{3.22}$$

where

$$w_j^i(t_k) = \frac{p(\tau_j^i, t_k|X^i_{t_1:t_{k-1}})hosp(\tau_j^i)}{\sum_{j=M}^{M} p(\tau_j^i, t_k|X^i_{t_1:t_{k-1}})hosp(\tau_j^i)}. \tag{3.23}$$

Therefore, instead of drawing a new sample from $p(x, t_k|X^i_{t_1:t_{k-1}})$ directly, we can draw the new sample from a finite set $\{\tau_j^i(t_k)\}_{j=1..N_l}$ with respect to their weights $\{w_j^i(t_k)\}_{j=1..N_l}$.

Remark 3.3.1: Unlike the IMM filter, the prediction stage of a particle filter is a closed-form solution itself. Thus, when there is no measurement at t_k, one can just run the prediction stage and get an estimate of the track PDF.

Drawing New Samples though Interacting Multiple Models

With the help of (3.22), the PF-HMap approach does not need the whole profile of $p(x, t_k|X^i_{t_1:t_{k-1}})$ as the BF-HMap method does. Therefore, non-analytic models such as a multi-modal structure are also acceptable as the primitive model in the PF-HMap approach, as long as new sample candidates can be drawn from the primitive model $\bar{p}(x, t_k|X^i_{t_1:t_{k-1}})$. In this subsection, we will incorporate the IMM structure into the prediction stage of the PF-HMap algorithm.

According to the IMM method, a model code $m^i(t_k)$ is added to each sample sequence. Thus, to generate a new sample candidate, one has to choose a model code m_i with respect to $\{\mu^{j_m}_{m^i(t_k)}\}_{j_m=1..N_m}$ first, and the generate a sample candidate based on $p^{m_i(t_k)}(x, t_k|X^i_{t_1:t_{k-1}})$. Here $\mu^{j_m}_{m^i(t_k)}$ is the switching probability from model $m^i(t_k)$ to model j_m. $p^{m_i(t_k)}(x, t_k|X^i_{t_1:t_{k-1}})$ stands for the predictive PDF of the model $m_i(t_k)$.

Remark 3.3.2: To reduce the computational cost introduced by the local Monte Carlo method, a practical way to generate sample candidates is to produce one candidate from each model. Thus, the computational complexity of the PF-HMap method is $O(N_m N)$, instead of $O(N_l N)$. A similar sampling strategy can be found in the IMM-Bootstrap method proposed in [105]. In fact, the IMM-Bootstrap method can be considered a special case (i.e. when the HMap is a constant) of this IMM version of the PF-HMap algorithm at the prediction stage.

3.3.2 The Estimation Stage

Since the new sample set is drawn from the predictive PDF $p(x, t_k|X^i_{t_1:t_{k-1}})$, the weights can be simply updated as follows:

$$w^i_{t_k} \propto w^i_{t_{k-1}} p(y(t_k)|x(t_k)), \tag{3.24}$$

where $y(t_k)$ is the extended measurement report that is defined in the BF-HMap algorithm, which is capable of handling the effect of a limited sensor FOV as well as the possibility of false alarms and miss detection

By combining (3.24) with (3.11), we then get the following weight updating equation for the PF-HMap algorithm:

$$w^i(t_k) \propto \begin{cases} \frac{P_0}{A_F} w_i(k-1)), & \text{if } y(t_k) \neq \emptyset \text{ and } x^i_{t_k} \notin F; \\ (1-P_1) p(y(t_k)|x^i_{t_k}) w_i(k-1)), & \text{if } y(t_k) \neq \emptyset \text{ and } x^i_{t_k} \in F; \\ P_1 w_i(k-1)), & \text{if } y(t_k) = \emptyset \text{ and } x^i_{t_k} \in F; \\ (1-P_0) w_i(k-1)), & \text{if } y(t_k) = \emptyset \text{ and } x^i_{t_k} \notin F; \end{cases} \tag{3.25}$$

Remark 3.3.3: With the measurement model above, we can run the estimation stage even without a detection report. A prediction-only loop can be considered as a special case here, in which $y(t_k) = \emptyset$ and $x^i_{t_k} \notin F$ for all $i = 1..N$.

3.3.3 Algorithm Summary

The whole procedure of the PF-HMap algorithm is summarized as the following (at time instant t_k):

1. For each sample $\{x^i_{t_k}, w^i_{t_k}\}$:

   Generate M_l sample candidates $\{\tau^i_j\}$ based on the primitive model $p(x, t_k|X^i_{t_1:t_{k-1}})$

   Calculate the weight $\{w^i_j(t_k)\}$ of each candidate according to (3.23);

   Draw a new sample $x^i_{t_k}$ from $\{\tau^i_j\}$ with respect to $\{w^i_j(t_k)\}$.

2. Update the new weights $\{w^i(t_k)\}_{i=1..N}$ according to (3.25).

3. Re-sample if the effective size of the sample set is below a pre-defined threshold N_{eff}.

3.4 A Relative-Entropy-Based Performance Measure For Non-Gaussian Tracking Problems

Since the combination of the mean and the covariance is not a sufficient representation of non-Gaussian target systems, the commonly used mean square error (MSE) becomes inappropriate in evaluating the the performance of non-Gaussian tracking algorithms. It is necessary to have a more comprehensive measure of performance that applies to both Gaussian and non-Gaussian systems.

Denote $p(x, t_k)$ as the true PDF of the target system described by (2.1) at t_k. One way to evaluate the performance of a track maintenance algorithm is to compare

the profile of the estimated posterior PDF $p(x, t_k|Y(t_k))$ with $p(x, t_k)$, where $Y(t_k)$ stands for the collection of the measurements obtained by the sensor up to t_k. Note that the more similar the two PDFs are, the better a track maintenance algorithm's performance is.

According to information theory, the similarity of two PDFs can be measured by their *relative entropy* [106]:

$$\begin{aligned} RE &= \int p(x, t_k) \log \left(p(x, t_k) / p(x, t_k|Y(t_k)) \right) dx \\ &= -\int p(x, t_k) \log p(x, t_k|Y(t_k)) dx + C. \end{aligned} \tag{3.26}$$

We define the first term in the right side of (3.26) as the *Relative Entropy Variance* (REV) of an estimated posterior PDF $p(x, t_k|Y(t_k))$:

$$REV = -\int p(x, t_k) \log p(x, t_k|Y(t_k)) dx. \tag{3.27}$$

Therefore, to compare the performance of two algorithms, we can just compare their REVs. The smaller REV one has, the better. Unfortunately, the true PDF $p(x, t_k)$ is not available for most systems. In this work, we use the following Monte Carlo method to estimate the true PDF.

Assume that one randomly runs the stochastic system (2.1) M times, and get M realizations of the target system at time t_k: $\{\bar{x}^j_{t_k}\}_{j=1..M}$. The true PDF of the target system at t_k can then be approximated as:

$$\bar{p}(x, t_k) = \frac{1}{M} \sum_{j=1}^{M} \delta(x - \bar{x}^j_{t_k}). \tag{3.28}$$

By substituting $p(x, t_k)$ in (3.27) with $\bar{p}(x, t_k)$, we get the approximate REV as:

$$REV(p(x, t_k|Y(t_k)))$$

$$\approx \quad -\int \bar{p}(x,t_k))\log p(x,t_k|Y(t_k))dx$$
$$= \quad -\frac{1}{M}\sum_{j=1}^{M}\log p(\bar{x}_{t_k}^j,t_k|Y(t_k)). \tag{3.29}$$

Here we further decompose the summation in (3.29) and define the following *individual relative entropy variance* ($iREV$) for each individual realization x:

$$iREV(x) = -\log p(x,t_k|Y(t_k)) \tag{3.30}$$

Remark 3.4.1: By comparing (3.29) with (3.30), one can see that REV can also be treated as the average $iREV$ of multiple tests. We consider REV and $iREV$ as a pair of tracking performance metrics for both Gaussian and non-Gaussian target systems. In fact, it can be easily shown that, for a Gaussian distribution, $iREV$ is proportional to the normalized (by the covariance matrix) square error.

In the case of particle filters, such as the proposed PF-HMap algorithm, the estimated posterior PDF itself is a summation of pulse functions (i.e. $\delta(\cdot)$), which makes it very difficult to quantify REV and $iREV$. A simple way to handle this problem is to replace the $\delta(\cdot)$ in (4.1) by a continuous Gaussian pulse as follows:

$$p(x,t_k|Y(t_k)) \approx \sum_{i=1}^{N} w_{t_k}^i N(x_{t_k}^i,\Sigma^i), \tag{3.31}$$

where $\Sigma^i = d_i^2 I_{n\times n}$, and $d_i = \min\{\|x_{t_k}^i - x_{t_k}^l\|, l = 1..N, l \neq i\}$, and $I_{n\times n}$ is the n-dimensional identity matrix.

In the next section, we will compare the performance of the proposed PF-HMap approach and other methods using the $REV/iREV$ criterion.

Remark 3.4.2: Eq. (3.31) above is essentially a special case of various ways to approximate a continuous PDF through multiple radial base functions. In practice, it is possible to use a small number (less than N) of Gaussian pulses to approximate

a continuous function, and the centers of these base functions are not necessarily to be the discrete samples. In fact, in the experiment shown in the next section, a single Gaussian base function is used to calculate the $iREV$ of the PF-HMap algorithm for scenario 3.

3.5 Simulations and Performance Evaluation

We apply the PF-HMap approach to a series of benchmark scenarios as follows:

- Scenario 1: First, we assume that a ground vehicle was detected by some remote sensor systems in the middle of a $3km \times 3km$ area described by the HMap in Fig 3.1. A UAV is assigned to validate this detection report and keep tracking the target. Before the UAV arrives in this area, there is a significant period of time that the target is out of the surveillance of any sensor system. The performance of the PF-HMap approach in maintaining predictive track information is tested in this scenario.

- Scenario 2: Then, when the UAV is approaching where the target was found, it is actually doing a search job. Experiments at this stage will show how the PF-HMap method is able to provide on-line target track information as the UAV is searching for the target.

- Scenario 3: After the UAV find the target, it starts the tracking job by circling around the target. Because of a limited FOV and physical motion constraints on the UAV, the target may still be out of surveillance intermittently. The performance of PF-HMap for tracking maneuvering target with intermittent and regional measurements is then tested in this scenario.

The performance of the PF-HMap approach will be compared with three existing representative methods, a Kalman filter, a IMM filter and the BF-HMap method along in each scenario. A constant velocity model is used in the KF method, which is also used as the primitive model in the BF-HMap approach. In the IMM filter, five models are used, which are a random walk model (mode 0) and four constant speed model toward east, south, west and north (mode $1-4$). The same IMM structure is used in the PF-HMap algorithm in this experiment. The sample size in PF-HMap is $N = 1000$ in this experiment. The switching coefficients between different models are chosen as

$$\mu_j^i = \begin{cases} 0.92, & \text{if } i = j, i > 0; \\ 0.02, & \text{if } i \neq j, i > 0; \\ 0.2, & \text{if } i = 0; \end{cases} \tag{3.32}$$

The UAV is modeled as a *Dubins car*:

$$\begin{cases} \dot{x}_u(t) = V_M cos\theta_u(t); \\ \dot{y}_u(t) = V_M sin\theta_u(t); \\ \dot{\theta}_u(t) = u(t), \qquad\qquad |u(t)| \leq V_M/R_M; \end{cases} \tag{3.33}$$

where $V_M = 100m/sec$, $R_M = 100meters$. Each UAV is assumed to has an on-board sensor system, which covers a local circular area (of radius $R = 150meters$) around the UAV. For the purpose of simplicity, the measurement model (before considering the effect of sensor FOV) is chosen as $H(x,t) = [\mathbf{x}(t), \mathbf{y}(t)]$, where $(\mathbf{x}(t), \mathbf{y}(t))$ denotes the horizontal position of the target on the ground. In addition, we assume that the sensor is perfect (i.e. $P_0 = P_1 = 0$).

In initialization, each filter is given the same positional measurement $[\mathbf{x}_0, \mathbf{y}_0]$ from the remote sensor system with a Gaussian covariance matrix P_0. The true trajectory of the target in each simulation starts from $[\mathbf{x}_0, \mathbf{y}_0]$ as well, with a speed randomly picked from 0 to $v_{max} = 10m/s$. The initial heading of the target is also randomly

selected. After that, the true trajectory of the target generated by the five-model-IMM scheme, except that the speed of the target is limited to v_{max}.

Remark 3.5.1: It is worth noting that PF-HMap is applicable to non-linear, complex measurement model as well. In fact, using a non-linear measurement model will only gives PF-HMap more advantages over the KF and the IMM approach.

3.5.1 Scenario 1: Track Maintenance for an Out-of-Surveillance Target

Since the target vehicle is mobile, it will probably not stay where it was found. It is important to maintain a good prediction of the positional information of the target before the UAV arrives. Fig. 3.4 shows how the predicted PDF maintained by the PF-HMap method evolves over time. The blue dots stands for the particles. For the purpose of illustration, the particles have been re-sampled, so that the density of the particles here is proportional to the predicted PDF. Through this example, we see that the terrain information carried by the HMap is effectively exploited.

We also compare the prediction performance the PF-HMap approach with the other three methods (i.e. KF, IMM and BF-HMap). 1000 Monte Carlo simulations are run. The true system PDF approximated by the 1000 realizations after 3 minutes is shown in Fig. 3.5.a. For comparison, the predicted PDFs maintained by the KF and the IMM methods are also illustrated by randomly generated samples. Note that in order to give the IMM filter a little bit more favor, we adjust the weights on the Gaussian blobs predicted by the five models with respect to the HMap. Therefore, the terrain information is used in all the four methods except for the Kalman filter. Comparing Fig. 3.4.d with Fig. 3.5.b, 3.5.c and 3.5.d, we can find that PF-HMap gives the best estimate of the true PDF of the target system. The evolutions of REVs

of the predictive PDF maintained by different methods are shown in Fig. 3.6. The results show that the predictions from the Kalman filter quickly diverge from the true target system. The IMM filter performs better than the Kalman filter initially, but becomes over-fitting as time increases. The BF-HMap method generates a much better predictive PDF, and the PF-HMap algorithm is the best in this experiment.

Remark 3.5.2: Since the PF-HMap approach approximates the predictive PDF by samples, numerical errors will degenerate its performance. However, the PF-HMap method still out-performs BF-HMap in this experiment. The reason behind this phenomenon probably is that the primitive model in the PF-HMap approach here (an IMM structure) is a better model than that of the BF-HMap (a constant velocity model) for maneuvering targets.

3.5.2 Scenario 2: Track Maintenance for Moving Target Search

In this scenario, the UAV arrives at the area after 3 minutes. However, since the vehicle has been out of surveillance for a long time. The UAV has to search around for the vehicle to validate the previous detection report.

In this experiment, the motion control strategy on the UAV is simply chosen as to drive the UAV toward the nearest particle. Experimental results in Fig. 3.7 show that the PF-HMap successfully utilizes the non-detection information from the UAV to update the PDF of the target state on-line, and thus helps the UAV to finds the vehicle in the end. Here we remove the HMap background in Fig. 3.7 for a better illustration.

Remark 3.5.3: The main purpose of this experiment is to show the capability of BF/PF-HMap in providing on-line target track information for the motion controller

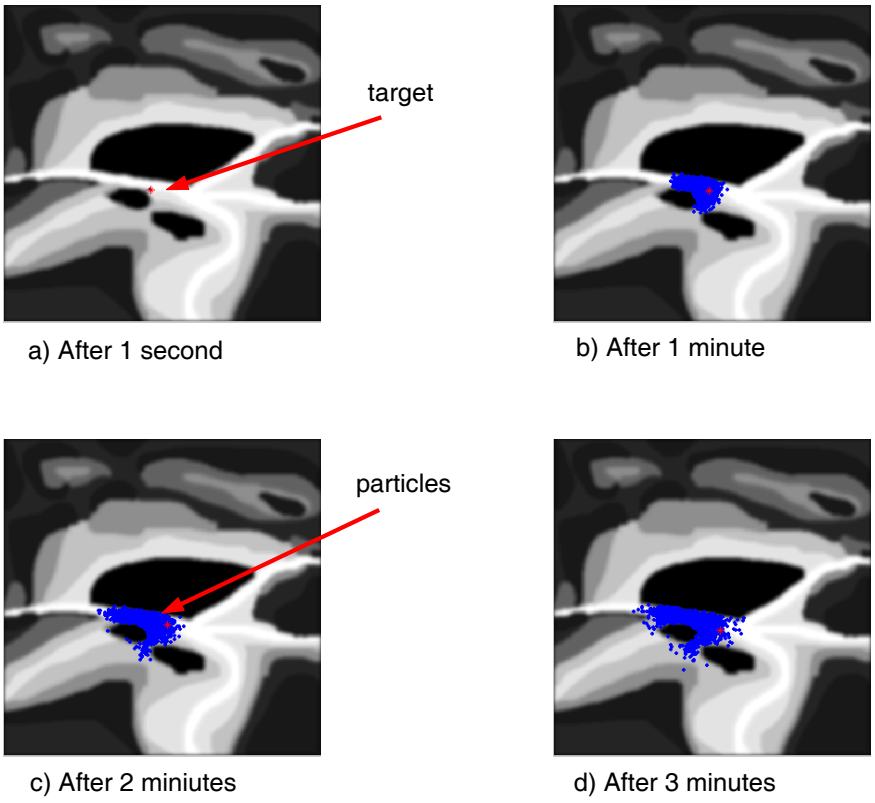


Figure 3.4: The evolution of the predictive PDF maintained by the PF-HMap approach in scenario 1.

of a mobile sensor, rather than the motion control itself. A more sophisticate motion control strategy is expected to get a better search/tracking result.

3.5.3 Scenario 3: Target Tracking with Intermittent and Regional Measurements

After the UAV finds the vehicle, it starts to circle around the target. Due to the motion constraints on the UAV, its FOV cannot cover the target all the time. The measurements obtained by the sensor are still intermittent and regional. Fortunately,

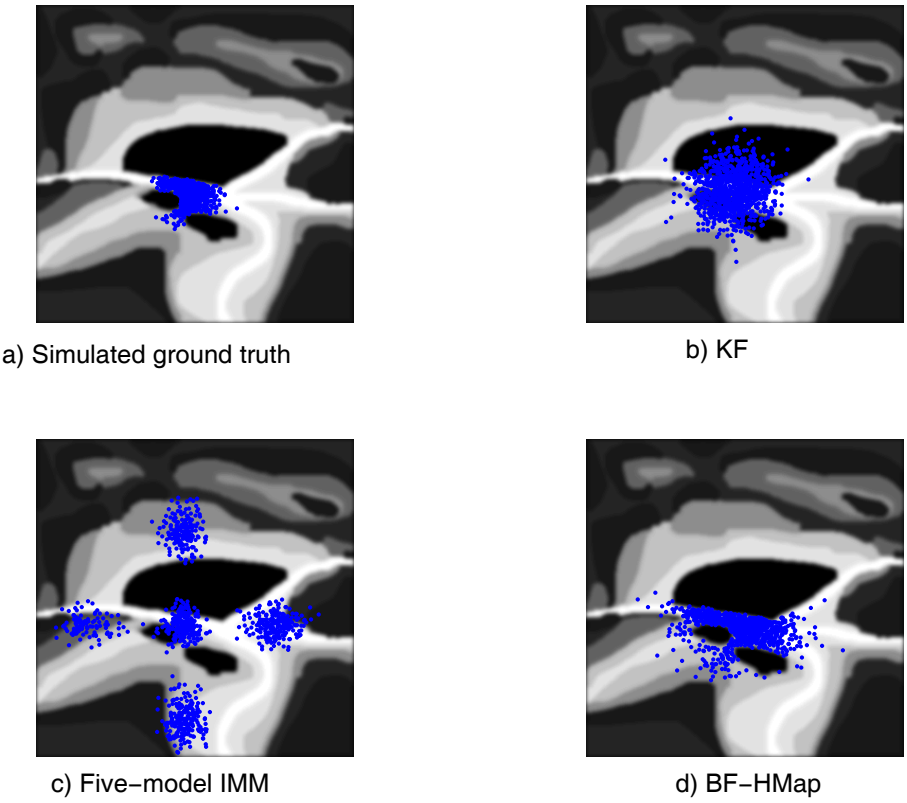


Figure 3.5: The PDF of the true system and the predictive PDF maintained by other approaches (in scenario 1 after 3 min.).

the flexible scheme of the PF-HMap approach allows this type of incomplete measurement information to be utilized. The same control strategy as in Scenario 2 is applied here as well.

We also compare the performance of the PF-HMap algorithm with other methods at this stage by varying the radius (R_s) of the sensor FOV from 20 meters to 300 meters. Fig. 3.8 shows the average REVs of different algorithms from 50 simulations. As we expect, the REVs decrease as R_s increases, because it is easier for the UAV to follow the target with a larger FOV. The performance of all the four methods are

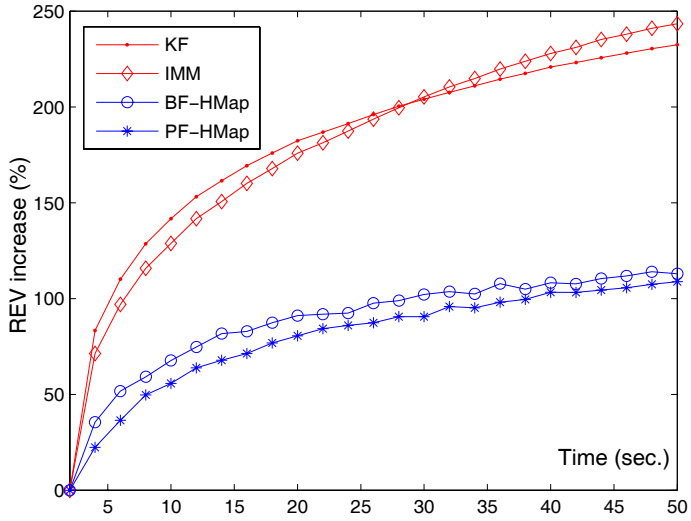


Figure 3.6: Prediction performance comparison in scenario 1.

close with a large R_s. As R_s decreases, the target is not guaranteed to be observable any longer, PF-HMap and BF-HMap then show a much better performance than the KF and the IMM approach, because of their capability of exploiting non-detection measurement information.

3.6 Conclusions

In this chapter, a generic framework (BF-HMap) for maintaining target track information is proposed based the Bayesian filtering method and the *hospitability map* concept. In the meantime, a computational efficient realization of this framework, the PF-HMap algorithm based on particle filters is introduced. The BF/PF-HMap algorithm have the following two key features:

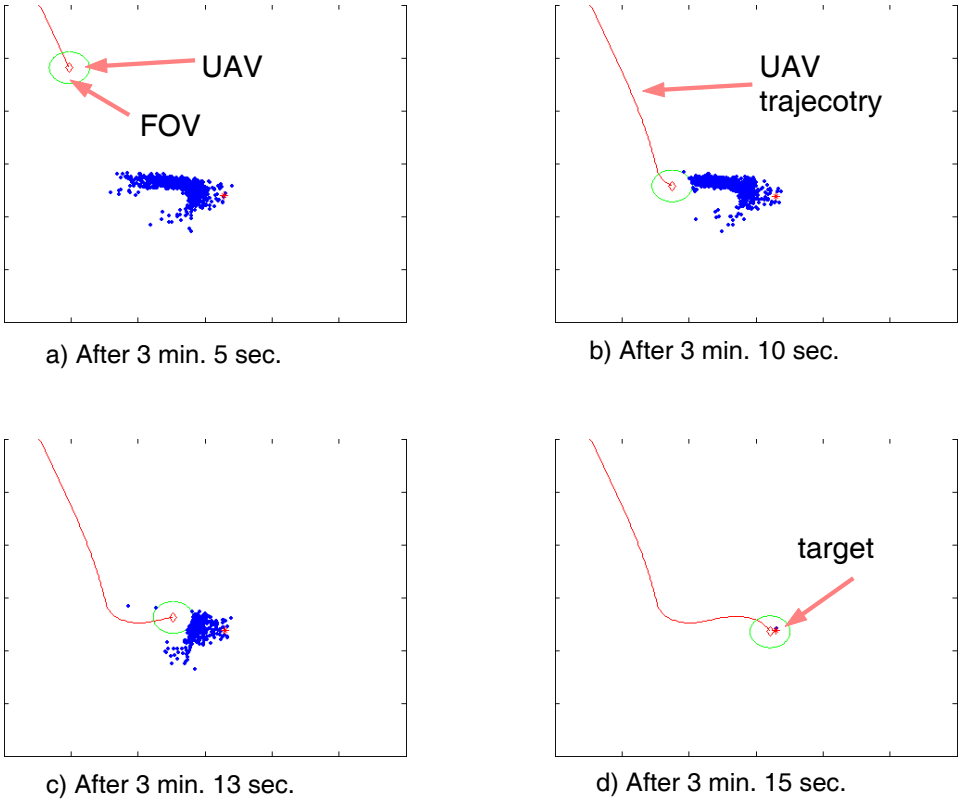


Figure 3.7: Online track maintenance for moving target search in scenario 2.

- Both algorithms are not only applicable to non-linear, non-Gaussian, multi-modal target systems, but also capable of using non-analytic terrain information to improve prediction performance.

- Both approaches can also handle complex measurement models caused by the limited view of a mobile sensor platform.

Due the above two features, the new Bf/PF-HMap method yields better performance in comparison with other existing methods, especially in dealing with *intermittent* and *regional* measurements obtained by MSAs. Meanwhile, Since the entire

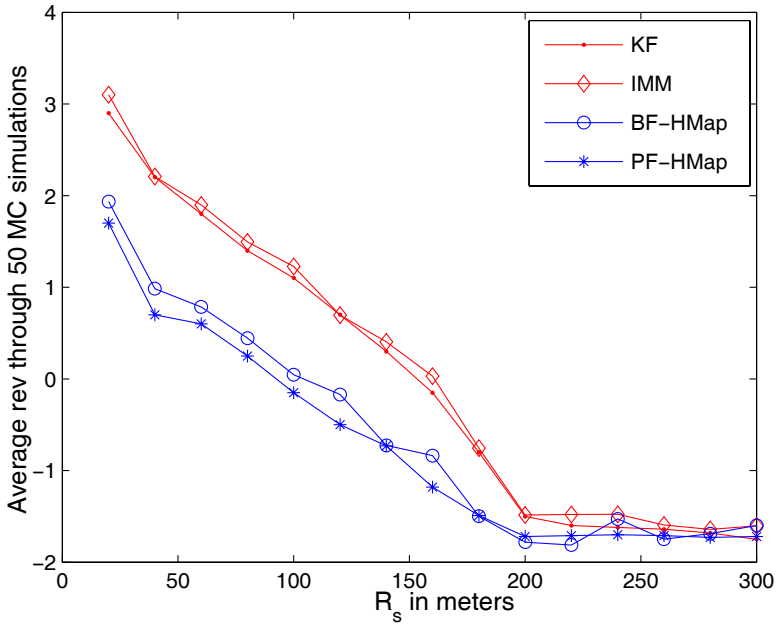


Figure 3.8: Performance comparison in scenario 3.

PDF (in terms of particles) is maintained by the BF/PF-HMap, both algorithms can be a very useful tool to provide online target information to the motion controller of mobile sensors in various applications, such as target search and target following. In particular, the PF-HMap approach has a much lower computational complexity than the BF-HMap algorithm, which makes it more feasible to be applied to real-time sensor systems.

Through the experiments shown in this chapter, we can see that there is not a distinct line between search and tracking when non-detection reports are also absorbed into the track maintenance scheme. Target search can then be considered as a special case of tracking. Thus, the BF/PF-HMap approach here can be further applied to the scenarios with MSAs performing both search and tracking tasks at the same time.

This chapter also introduces a new performance measure based on *relative entropy* for non-Gaussian tracking problems. Thus, the whole posterior PDF of the target system is evaluated, which can then used as a general, consentient criterion to compare KF-, BF-, PF-based and other tracking algorithms.

CHAPTER 4

DISTRIBUTED PARTICLE FILTERS FOR NETWORKED MOBILE SENSOR AGENTS

4.1 Introduction

The previous chapter mainly focuses on the target track maintenance problem in the single-MSA case. In many real world applications, however, one has to fuse observations from multiple MSAs in order to effectively track the evolution of a track system. Although a great amount of muli-sensor fusion approaches for target tracking have been proposed in the last two decade, most of them assumes that the measurements come in sequence, which requires instant sensor-to-senor or sensor-to-fusion-center communication. However, in the case of networked MSAs, no matter the sensor fusion procedure is achieved in a centralized way or a distributed way, communication delays among different MSAs (or between a fusion center and MSAs) are almost inevitable. As result, the corresponding track maintenance approach has to be able to deal with measurements that do not come sequentially. Such a non-standard tracking problem is also called the *out-of-sequence measurements* (OOSM) problem in literature [99]. Recently, a few approaches dealing with OOSM problems have

been proposed be researchers [99–101], most of which are mainly designed for linear systems.

In this chapter, we studied the OOSM problem in the context of tracking non-linear target systems with a *particle filter* (PF), which is in accord with our choice of PF over KF-based methods in the previous chapter. A generic statement of this OOSM problem is given in section 4.2 first. Then, a novel target tracking algorithm for that is capable of dealing with both in-sequence and out-of-sequence measurements, called the *Universal Particle Filter* (UPF), is developed in section 4.3. After that, some practical issues on the application of the UPF in real-world tracking problems is discussed (section 4.3), along with experiment results and performance analysis being presented (section 4.4). In the conclusion section of this chapter (section 4.5), possible directions of future work on multi-sensor fusion in distributed sensor networks based on the UPF method are discussed.

4.2 Problem Statement

The *particle filtering* method is essentially an approximation of the *Bayesian filtering* method, which approximates the posterior PDF of a target track by a set of samples and their weights $\{X^i_{t_1:t_k}, w^i_{t_k}\}_{i=1..N}$:

$$p(X(t_k)|Y(t_k)) \approx \sum_{i=1}^{N} w^i_{t_k} \delta(X(t_k) - X^i_{t_1:t_k}), \tag{4.1}$$

where $X(t_k) = \{x(t_1), ..., x(t_k)\}$; $Y(t_k) = \{y(t_1), ..., y(t_k)\}$, $X^i_{t_1:t_k} = \{x^i_{t_1}, ..., x^i_{t_k}\}$; $X(t_k), X^i_{t_1:t_k} \in R^{n_x \times (k)}$ and $Y(t_k) \in R^{m \times k}$.

The general OOSM problem in the context of particle filters can be described as the following:

Given a sequential particle set at time t_k as $\{X^i_{t_1:t_k}, w^i_{t_k}\}_{i=1..N}$ and a delayed measurement $y(t')$ where $t_l < t' \leq t_{l+1} \leq t_k$, we want to get a new particle set and the weights $\{\tilde{X}^i, \tilde{w}^i\}_{i=1..N}$ such that

$$p(X(t_k)|Y(t_k), y(t')) \approx \sum_{i=1}^{N} \tilde{w}^i \delta(X(t_k) - \tilde{X}^i), \tag{4.2}$$

where $\tilde{X}^i = \bigcup(X^i_{t_1:t_k}, \tilde{x}^i_{t'})$ is the updated sample sequence; $\{\tilde{x}^i_{t'}\}_{i=1..N}$ is the new sample set for time t'; $\tilde{w}^i$ is the updated weight for the new sample sequence $\tilde{X}^i$. Without loss of generality, let us assume that $t_l \leq t_m$ for all $l < m$. In other words, we assume that the particles are stored sequentially before $y(t')$ comes.

4.3 Universal Particle Filter (UPF): A Particle Filter for the OOSM Problem

The key procedures in a particle filter are drawing new samples and updating weights. When the measurements come in an ideal time order, (2.31) provides us a convenient way to draw new samples for a standard particle filter. However, in the case of OOSM, (2.30) does not hold in general and either does (2.31). Therefore, a different treatment has to be made in dealing with delayed measurements.

When a new measurement $y(t')$ comes, there are three possibilities:

Case 1: $\forall l = 1..k, t' > t_l$

In this case, $y(t')$ is the most recent measurement. We can just use the standard PF (2.28) to draw new samples and update the weights (2.29).

Case 2: $\exists l$**, such that** $t' = t_l$ **and** $1 \leq l \leq k$

In this case, there are already a set of samples drawn at time t_l, which means we do not have to draw a new sample set $\tilde{x}^i_{t'}$ for time t'. Therefore, we have $\tilde{X}^i = X^i_{t_1:t_k}$.

Based on the Bayes's rule, we have:

$$\begin{aligned} p(\tilde{X}^i|Y(t_k),y(t')) &= p(X^i_{t_1:t_k}|Y(t_k),y(t')) \\ &= \frac{p(X^i_{t_1:t_k},Y(t_k),y(t'))}{p(Y(t_k),y(t'))} \\ &= \frac{p(y(t')|X^i_{t_1:t_k},Y(t_k))p(X^i_{t_1:t_k}|Y(t_k))}{p(y(t')|Y(t_k))} \\ &= \frac{p(y(t')|x^i_{t_l})p(X^i_{t_1:t_k}|Y(t_k))}{p(y(t')|Y(t_k))} \\ &\propto p(y(t')|x^i_{t_l})p(X^i_{t_1:t_k}|Y(t_k)) \end{aligned} \tag{4.3}$$

Here we assume that the measurement noises are independent of each other. Eq. (4.3) indicates that, in order to satisfy (4.2), we only have to update the weights as:

$$\tilde{w}^i \propto p(y(t')|x^i_{t_l})w^i_{t_k}. \tag{4.4}$$

Case 3: $\exists l$, such that $t_l < t' < t_{l+1}$ and $1 \leq l < k$

In this case, no samples are available to utilize the information that the new measurement $y(t')$ provides. It is necessary to draw an additional sample set $\tilde{x}^i_{t'}$ for time t'. By doing so, we can then insert $\tilde{x}^i_{t'}$ into the previous particle sequences $\{X^i_{t_1:t_k}\}_{i=1..N}$ to generate a new set of particle sequences $\tilde{X}^i_{i=1..N}$, as illustrated in Fig. 4.1.

Base on the Bayes' rule, we have:

$$\begin{aligned} p(\tilde{X}^i|Y(t_k),y(t')) &= \frac{p(X^i_{t_1:t_k},\tilde{x}^i_{t'},y(t')|Y(t_k))}{p(y(t')|Y(t_k))} \\ &= \frac{p(\tilde{x}^i_{t'},y(t')|X^i_{t_1:t_k},Y(t_k))p(X^i_{t_1:t_k}|Y(t_k))}{p(y(t')|Y(t_k))} \\ &= \frac{p(y(t')|\tilde{x}^i_{t'})p(\tilde{x}^i_{t'}|X^i_{t_1:t_l},X^i_{t_{l+1}:t_k},Y(t_k))p(X^i_{t_1:t_k}|Y(t_k))}{p(y(t')|Y(t_k))} \\ &\propto \frac{p(y(t')|\tilde{x}^i_{t'})p(\tilde{x}^i_{t'}|x^i_{t_l})p(x^i_{t_{l+1}}|\tilde{x}^i_{t'})}{p(x^i_{t_{l+1}}|x^i_{t_l})}p(X^i_{t_1:t_k}|Y(t_k)) \end{aligned} \tag{4.5}$$

Combining (3.24) with (4.5), we can obtain the new wights $\tilde{w}^i$ by:

$$\tilde{w}^i \propto \frac{p(y(t')|\tilde{x}^i_{t'})p(\tilde{x}^i_{t'}|x^i_{t_l})p(x^i_{t_{l+1}}|\tilde{x}^i_{t'})}{q(x^i_{t'}|X^i_{t_1:t_k},Y(t_k),y(t'))p(x^i_{t_{l+1}}|x^i_{t_l})}w^i_{t_k}. \tag{4.6}$$

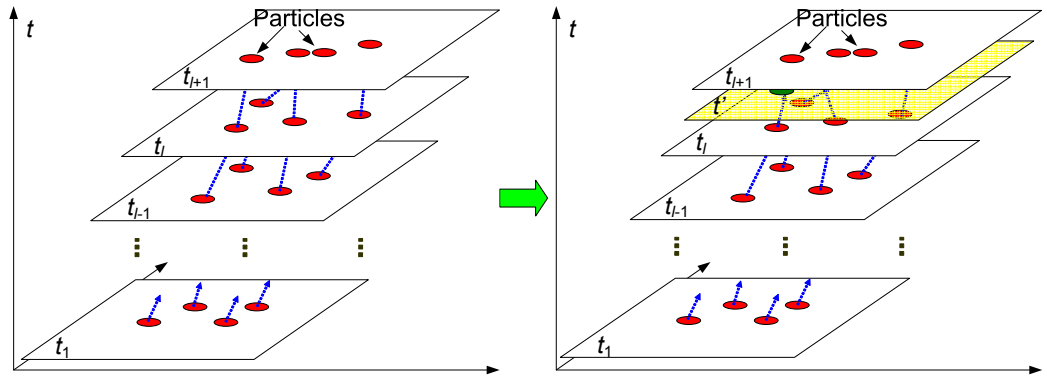


Figure 4.1: Case 3 of the UPF: to generate a new sample set and insert them into the previous particle sequences.

In this approach, we choose the *importance density* as:

$$q(\tilde{x}^i_{t'}|X^i_{t_1:t_k}, Y(t_k), y(t')) = p(\tilde{x}^i_{t'}|X^i_{t_1:t_l}), \tag{4.7}$$

the corresponding weights update equation is

$$\tilde{w}^i \propto \frac{p(y(t')|\tilde{x}^i_{t'})p(x^i_{t_{l+1}}|\tilde{x}^i_{t'})}{p(x^i_{t_{l+1}}|x^i_{t_l})} w^i_{t_k}. \tag{4.8}$$

Eq. (4.7) and (4.8) imply that in order to deal with the OOSM problem, the tracking system has to be able to retrieve the particles in past time. If the communication delay can be arbitrarily large, each MSA will need an infinite memory space to store the particle sequences, which is impossible to achieve in real world implementation.

4.4 The Limited-memory UPF

The idea of the limited-memory UPF is very straightforward, which only keeps most recent particles no elder than ΔT from the current time. Considering the fact that the communication delay between sensor nodes is usually bounded in reality, it

is reasonable to store the most recent segments of each particle sequence only. The question is how to choose such an appropriate ΔT in practice.

Denote the minimum sampling time unit as Δt. The actually memory requirement on each MSA for one target is of size $O(\frac{N\Delta T}{\Delta t})$, where N is the size of the sample set. Meanwhile, the computational load of each cycle of a particle filter is $O(N)$. Thus, the choice of ΔT and N is a tradeoff among system performance, the memory cost and the computational load.

4.4.1 The effect of the sample size (N) on the tracking performance

Investigations done by Geweke [107] have shown that as the sample size (i.e. N) increases, the approximation error introduced by a particle filter is loosely decreasing by a fact of $1\sqrt{N}$ as follows:

$$\sqrt{N}(\hat{X}_{t_1:t_k} - X_{t_1:t_k}) \rightarrow N(0, \sigma^2), \tag{4.9}$$

where

$$\hat{X}_{t_1:t_k} = \frac{1}{N}\sum_{i}^{N} X^i_{t_1:t_k}, \tag{4.10}$$

and

$$\sigma^2 = E\left[(X_{t_1:t_k} - E[X_{t_1:t_k}])^2 \frac{p(Y(t_k)|X_{t_1:t_k})p(X_{t_1:t_k})}{q(X_{t_1:t_k}|Y(t_k))}\right], \tag{4.11}$$

where $q(X_{t_1:t_k}|Y(t_k))$ is the importance density function.

4.4.2 The effect of the memory size (ΔT) on the tracking performance

The relationship between ΔT and the tracking performance is more complicated. In KF-based methods, additional measurement information always leads to superior

estimation performance. However, it may not be the case in the context of particle filters. As shown in (4.1), what a PF approximates is essentially the joint PDF of the target states at a sequence of time instant from t_1 to t_k. Therefore, inserting a new sample set at some past time instant will actually increase the dimensionality of the joint space of $X(t_k)$ in (4.1) and thus increase the approximation uncertainty. In the meantime, due to the re-sampling procedure, most of the particles at current time instant may actually come from the same "parent particle". In this case, the weights as well as the estimates will not change at all when a delayed measurement that is elder than the "parent particle" runs through the UPF algorithm.

4.5 Distributed Sensor Fusion for Target Tracking based on the UPF

Based on the UPF technique developed above, we can build a framework of distributed sensor fusion for target tracking, as illustrated in Fig. 4.2. In this frame work, each MSA has its own local sensor system. The measurements obtained by the local sensor system are called *internal measurements*. Meanwhile, we assume that there is a communication network among the MSAs, and the MSAs are sharing their local measurements with each other trough the communication network. The measurement reports that one MSA receives from other MSAs are called *external measurements*. Thus, each MSA runs its own UPF based on both the internal and external measurements. Thus, the sensor fusion here is a measurement-to-track fusion.

When one MSA sends out its newly obtained measurement, the message package also includes the following critical information besides the measurement.

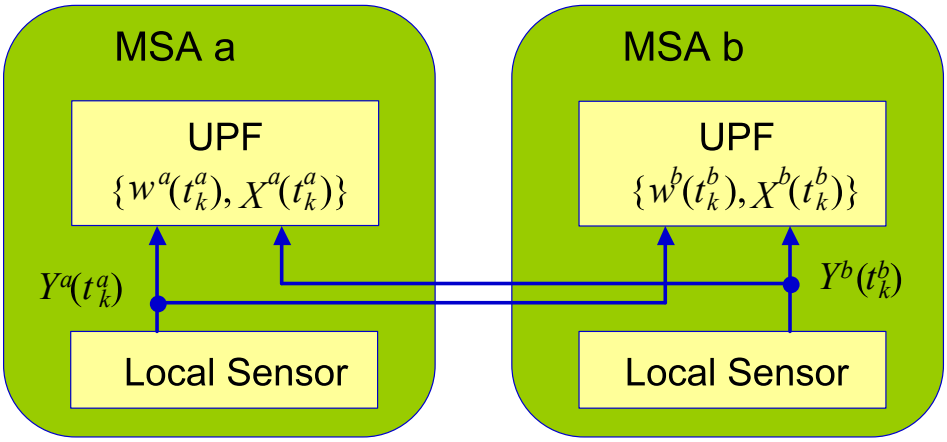


Figure 4.2: The diagram of distributed sensor fusion for target tracking using UPFs

- The sender's ID;
- The global time stamp of the measurement;
- The sensor model of the sender.

The global time stamps is used to align measurements from different sensor sources into the same time frame, which can be easily obtained with the assumption that a reliable GPS clock is available to each sensor node. The acknowledgement of sensor model of the sender is also a prerequisite for successful sensor fusion. A convenient way the transmit sensor models among MSA's is to pre-code the type of sensors and put a lookup table of sensor models at each MSA.

4.6 Experiments and Results

4.6.1 UPF vs. Standard PF without any Delay

To evaluate the performance of the proposed UPF, we choose the follow non-linear discrete time system:

$$x(k+1) = \frac{x(k)}{2} + \frac{25x(k)}{1+x^2(k)} + 8\cos(1.2k) + v(k), \tag{4.12}$$

where $v(k) \sim N(0, Q(k))$.

The sensor model is also non-linear:

$$y(k) = \frac{x^2(k)}{20} + n(k), \tag{4.13}$$

where $n(k) \sim N(0, R(k))$.

The system above has been widely used as a benchmark example to compare the performance of the PF and the conventional KF-based methods [102, 104]. Thus, in this experiment, we only compare the performance of a standard PF without any communication delay with the proposed UPF with communication delay.

Fig. 4.3 shows the result of one simulation, in which MSE and MSE_1 are the tracking errors of the standard PF and the UPF, respectively. In the case of UPF, a 2-step delay is simulated on every other measurement. The result shows that the performance of the UPF is pretty close to the ideal one (i.e. the standard PF without any communication delay).

Fig. 4.4 shows the average tracking errors of 50 Monte Carlo simulations, MSE and MSE_1 are the mean square errors of the standard PF and the UPF. The results demonstrate that the tracking performance of the UPF is very close to the standard PF in spite of the presence of communication delays.

4.6.2 UPF vs. Limited-memory UPF

In this experiment, we study the contribution of delayed measurements to tracking errors. Fig. 4.5 shows the performance comparison between the UPF and an LUPF in tracking a simple random-walk target as the following:

$$x(k+1) = x(k) + w(k), \tag{4.14}$$

$$y(k) = x(k) + v(k). \tag{4.15}$$

As Fig. 4.5 indicates, measurements being delayed more than one step is not that helpful in this example. The reason probably is that the information gained by such a delayed measurement is overwhelmed by additional variations introduced by new particles.

4.7 Conclusions

In this chapter, a novel particle filtering method, the Universal Particle Filter (UPF) is developed, which is capable of tracking non-linear target systems with out-of-sequence measurements. Based on the UPF, a generic framework of distributed sensor fusion for target tracking is also introduced. Some practical issues on the application of the UPF in real-world tracking problems is also discussed. Since the UPF requires historical information to generate new samples and update the weights, additional memory space is needed. Fortunately, both theoretical analysis and experiments show that a large memory load is often neither affordable nor necessary in practice.

It is worth mentioning that there are other technical issues in distributed mobile sensor networks that are left for further investigations. For example, many practical tasks require cooperations among multiple MSAs. Significant differences between

the target tracks maintained by different MSAs can easily lead to the collapse of a cooperative sensing task. Unlike conventional KF-based methods, the estimation difference between two particle filters is inevitable even under a perfect communication condition, thanks to its simulation-based nature. Further theoretic analysis on the relationship between the variance of estimation and the choice of key parameters such as the size of the sample size N and the communication channel configurations has to be conducted before the UPF and the PF-HMap approach proposed in the previous chapter can be applied to fully distributed mobile sensor networks.

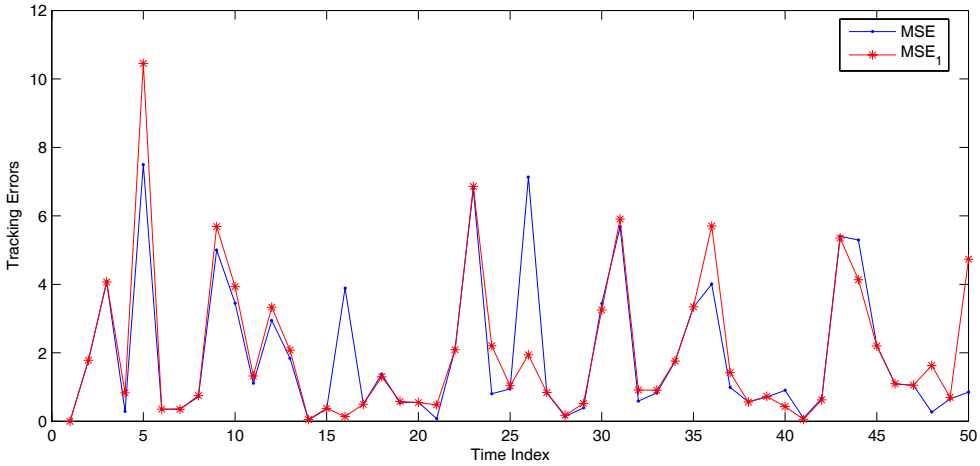


Figure 4.3: Tracking errors of a standard PF without measurement delay (MSE) and the UPF (MSE_1) with a 1-step lag on every other measurement: MSE and MSE_1 are the tracking errors of the standard PF and the UPF, respectively.

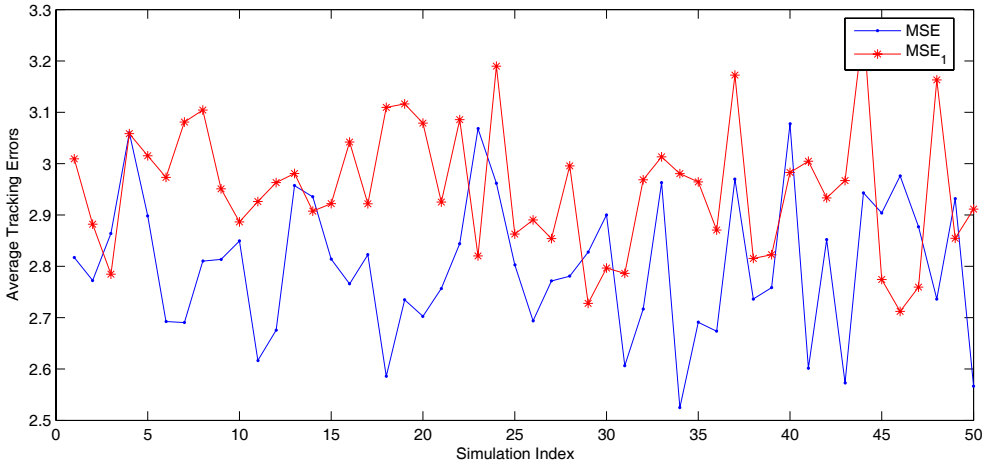


Figure 4.4: Performance comparison between a standard PF without measurement delay (MSE) and the UPF (MSE_1) with a 1-step lag on every other measurement: MSE and MSE_1 are the mean square errors of the standard PF and the UPF (across 50 MC simulations).

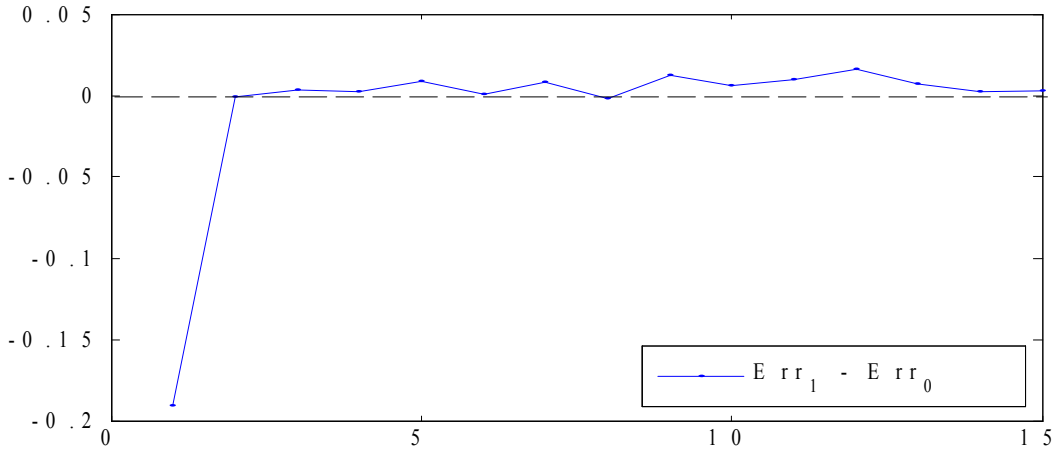


Figure 4.5: Effect of delayed measurements on tracking performance (Err_0 and Err_1 are the mean square errors of the standard PF and the UPF (across 50 MC simulations).

CHAPTER 5

FOUNDATIONS OF ENTROPY-BASED MANAGEMENT OF MSAS

5.1 Introduction

The main task of *sensor management* is to control the sensor resources to achieve and maintain a good representation of the subject of interest, which is the kinematic state of one or multiple moving objects in the MSA-target scenario. In other word, *sensor management* shares the same system goal with the information processing module. Thus, it is natural to consider the management problem of MSAs from the perspective of information processing.

Perhaps the first attempt to modeling a sensor management problem in an information-theoretic way is the optimal sensor-to-target assignment approach proposed by Nash [5]. In this approach, the target state vector, $x(t) \in \mathcal{R}^{n_x}$, is assumed to be maintained by a Kalman filter, and the objective function of sensor assignment is defined with respect to the trace of the co-variance matrix of $x(t)$. During the last two decades, a more general information measure, has been widely used in information-theoretic approaches for a variety of sensor scheduling problems [6, 7, 15], which is the entropy

of the posterior track PDF $p(x,t|Y(t))$ as follows:

$$\varepsilon(x,t|Y(t)) = -\int p(x,t|Y(t)) \log p(x,t|Y(t)) dx, \tag{5.1}$$

where $Y(t) = \{y(t_1), y(t_2), ..., y(t_k)\}$, $t_1 \leq t_2 \leq ... \leq t_k \leq t$ and $y(t_1) \in \mathcal{R}^m$. Note that the integration in (5.1) will be replaced by a summation in the discrete-space case.

Since the entropy above is essentially a measure of uncertainty $x(t)$, by controlling the evolution of the entropy, one can keep the uncertainty of $x(t)$ from increasing, or reduce it to a desired level. Among existing information-theoretic sensor management approaches, many are devoted to a specific target/sensor model, mostly in a KF/EKF-based framework [6, 7, 10, 12–15, 17]. As a result, there is very few theoretical study on how the entropy evolves over time and how entropy changes with respect to sensor actions, especially for the general non-linear case. However, due to the physical constraints on the motion and coverage of MSAs, the evolution of the entropy along with the motion of MSAs is a crucial factor in both modeling the MSA control problem and developing the corresponding motion control algorithms.

The goal of this chapter is to develop general foundations for entropy-based MSA management by studying the evolution of the entropy. Similar to $p(x,t|Y(t))$, the dynamics of the entropy $\varepsilon(x,t|Y(t))$ is driven by the joint effect of $x(t)$ and $Y(t)$. Thus, the evolution of $\varepsilon(x,t|Y(t))$ consists of two parts: the *time evolution*, which is governed by (2.1), and the *measurement evolution*, which describes the change of $\varepsilon(x,t|Y(t))$ with respect to newly obtained measurements. In what follows, we shall study the *time evolution* and the *measurement evolution* of the entropy in section 5.2 and 5.3, respectively. Without loss of generality, let us assume that a discrete-time Kalman filter is used to maintain the target information for the linear-Gaussian

case and the BF-HMap for the non-linear case. Some key properties of the evolution of the entropy are identified along the study, and are used as strategic guidelines to formulate the *sensor management* problem for different MSA-target scenarios in section 5.4.

5.2 The Time Evolution of the Entropy

The *time evolution* basically takes place between two consecutive sensor observations (denoted as $y(t_{k-1})$ and $y(t_k)$). Since no measurement is involved at this stage, $Y(t) \equiv Y(t_{k-1})$, $t_{k-1} \leq t < t_k$. Thus, the entropy here can be directly computed from the predictive PDF $p(x, t|Y(t_{k-1}))$, which is determined by the characteristics of $x(t)$ (i.e. the target model (2.1)) only.

5.2.1 The Linear-Gaussian Case

In the case that both the target model and the sensor model are linear and Gaussian, the Kalman filter is an efficient tool to maintain the target information. Since the Kalman filter is often implemented in discrete time in practice, here we use the discrete-time version of the Kalman filter to study the *time evolution* of the entropy for the linear-Gaussian case.

Let the following equation be the target model in a discrete time Kalman filter:

$$x(t_k) = F(t_k)x(t_{k-1}) + G(t_k)\beta(t_k), \tag{5.2}$$

which is usually obtained by discretizing a continuous-time target model:

$$\dot{x}(t) = f(t)x(t) + g(t)\beta(t), \quad t_{k-1} \leq t < t_k. \tag{5.3}$$

, where $F(t_k) = f(t_k)\Delta T + I$, $G(t_k) = g(t_k)\Delta T$, and $\Delta T = t_k - t_{k-1}$.

It has been known that for a linear Gaussian system, its entropy is linearly-proportional to the log of the determination of the co-variance matrix [6]. Thus, we have

$$\varepsilon(x,t|Y(t_{k-1})) \propto \log(\det P(t|t_{k-1})), \quad t \geq t_{k-1}, \tag{5.4}$$

where $P(t|t_{k-1})$ is the co-variance matrix of the predictive PDF $p(x,t|Y(t_{k-1}))$ maintained by a Kalman filter.

According to (2.7), $P(t_k|t_{k-1})$ can be obtained from (5.3) as follows:

$$P(t_k|t_{k-1}) = F(t_k)P(t_{k-1}|t_{k-1})F^T(t_k) + G(t_k)Q(t_k)G^T(t_k). \tag{5.5}$$

The linear algebra theory [97] indicates that for any matrix $F \in \mathcal{R}^{n_x \times n_x}$, there exists an orthogonal matrix $D\mathcal{R}^{n_x \times n_x}$ such that

$$F = D^{-1}\Lambda D, \tag{5.6}$$

where $\Lambda\mathcal{R}^{n_x \times n_x}$ is a upper triangular matrix. Note that, the diagonal element (λ_i,$i = 1..n_x$) of Λ are actually the eigenvalues of F.

Denote $D(t)$ as the orthogonal matrix that transform $f(t)$ to a upper triangular matrix $\Lambda(t)$ in (5.6). We then have

$$\begin{aligned} F(t_k) &= f(t_k)\Delta T + I \\ &= D^{-1}(t_k)\Lambda(t_k)D(t_k) + I \\ &= D^{-1}(t_k)(\Lambda(t_k) + I)D(t_k), \end{aligned} \tag{5.7}$$

and

$$\begin{aligned} P(t_k|t_{k-1}) &= D^{-1}(t_k)(\Lambda(t_k)\Delta T + I)D(t_k)P(t_{k-1}|t_{k-1})D^{-1}(t_k)(\Lambda^T(t_k)\Delta T + I)D(t_k) \\ &+ g(t_k)Q(t_k)g^T(t_k)\Delta T^2. \end{aligned} \tag{5.8}$$

Combining (5.8) with (5.4), we get

$$\begin{aligned}
& \varepsilon(x, t_k|Y(t_{k-1})) \\
\propto\ & \log(\det P(t_k|t_{k-1})) \\
\geq\ & \log\left[\det\left(D^{-1}(t_k)(\Lambda(t_k)\Delta T + I)D(t_k)P(t_{k-1}|t_{k-1})D^{-1}(t_k)(\Lambda^T(t_k)\Delta T + I)D(t_k)\right)\right] \\
=\ & \log\left[\prod_i^{n_x}(\lambda_i(t_k)\Delta T + 1)^2 \det P(t_{k-1}|t_{k-1})\right] \qquad (5.9)
\end{aligned}$$

where $\lambda_i(t_k)$ is the i^{th} eigenvalue of $\Lambda(t_k)$.

Eq. (5.9) above directly leads the following conclusion.

Theorem 5.2.1: If the transfer function $f(t)$ in the continuous-time model (5.3) is semi-positive definite for $\forall t \geq t_k$, we have

$$\varepsilon(x, t_2|Y(t_k)) > \varepsilon(x, t_1|Y(t_k)), \quad \forall t_k \leq t_1 < t_2. \qquad (5.10)$$

(5.10) indicates that the entropy of the open-loop (i.e. without any measurement) system (5.3) is monotonically increasing, which means that the status of $x(t)$ becomes more and more uncertain if there is no further measurements.

Remark 5.2.1: A positive definite $f(t)$ also implies that the target model (5.3) is a unstable system, which is totally make sense in a sensing problem. Otherwise, if the target system is stable, there will be no need to use a sensor to monitor $x(t)$ since it will converge to some equilibrium point eventually no matter what happens. In fact, many widely-used target models such as the random walk model, the constant velocity model and the constant acceleration model have a semi-positive definite $f(t)$.

Theorem 5.2.1 also leads to the following corollary.

Corollary 5.2.2: Consider two linear-Gaussian systems governed by the same model (5.3). Denote $\varepsilon_1(x, t|Y(t))$ and $\varepsilon_2(x, t|Y(t))$ as the entropies of the two systems,

respectively. It is true that

$$\varepsilon_1(x,t|Y(t_k)) \geq \varepsilon_2(x,t|Y(t_k)), \quad \forall t \geq t_k, \tag{5.11}$$

if $f(t)$ in (5.3) in semi-positive definite and

$$\det P_1(t_k|t_k) \geq \det P_2(t_k|t_k), \tag{5.12}$$

where $P_1(t_k|t_k)$ and $P_2(t_k|t_k)$ are the initial covariance matrix of the two systems at t_k, respectively.

5.2.2 The Non-linear Case

For a non-linear target system, if the target information can be adequately maintained by an EKF or an IMM filter, then the result in the linear-Gaussian case presented in the previous subsection can still be employed. The reason is that the EKF assumes that $x(t)$ is locally linearizable, and the IMM filter basically assumes that $x(t)$ is switching among multiple linear models. However, for the general non-linear target track maintenance problems, the *time evolution* of the entropy is much more complicated, especially when non-analytic environmental information (e.g. the *hospitability map* in Chapter 3) is absorbed into the target model. Nevertheless, as *Remark 5.2.1* indicates, it is still true in most sensing problems to assume that the entropy will keep increasing when there is no measurement input.

On the other hand, it is often true in real-world applications that the track PDF is only distributed in a finite subset $S(t)$ of the information space, which means

$$A(t) = \int_{x \in S(t)} dx < \infty, \tag{5.13}$$

where $S(t) = \bigcup\{x; x \in \mathcal{R}^{n_x}, p(x,t|Y(t)) > 0\}$.

According to the definition of entropy, it can be easily shown that [106]

$$\varepsilon(x, t|Y(t)) \leq \log A(t). \tag{5.14}$$

Note that the equality holds if and only if $x(t)$ is uniformly distributed in $S(t)$.

Remark 5.2.2: One can consider $S(t)$ as a "cloud" of track PDF, which expands and shrinks along with the evolution of the entropy. Since the extent of the PDF cloud (i.e. $A(t)$) is an upper bound of the entropy, one can indirectly control the entropy by controlling the evolution of $S(t)$, which can possibly simplify the control design problem in practice.

5.3 The Measurement Evolution of the Entropy

As the *time evolution* of the entropy being governed by the target system's own characteristics, the *measure evolution* of the entropy shows how sensor actions affect the uncertainty of the target. Similar to other entropy-based sensor management approaches, here we use the change in entropy to quantify the *information gain* of an sensor action as follows.

Definition 5.3.1: The *information gain* (IG) of taking a sensor control action u, which attains a measurement $y(t_k)$, is defined as:

$$\Delta\varepsilon(x, t_k|u, Y(t_k)) = \varepsilon(x, t_k|Y(t_{k-1})) - \varepsilon(x, t_k|u, Y(t_k)). \tag{5.15}$$

Clearly, the measurement $y(t_k)$ is unknown until the control action u is taken. To evaluate the expected performance of a sensor action, we further introduce the *Expected Information Gain* as follows.

Definition 5.3.2: Let $\mathbf{Y}(u) \subset \mathcal{R}^m$ be the set of all possible measurements coming from a sensor action u. The *expected information gain* (EIG) of u is defined

as:

$$\begin{aligned}\Delta\varepsilon(x,t|u) &= \sum_{y\in\mathbf{Y}(u)} p(y(t_k)=y)\Delta\varepsilon(x,t_k|\bar{Y}(t_k)) \\ &= \varepsilon(x,t_k|Y(t_{k-1})) - \sum_{y\in\mathbf{Y}(u)} p(y(t_k)=y)\varepsilon(x,t_k|u,\bar{Y}(t_k)). \quad (5.16)\end{aligned}$$

where $\bar{Y}(t_k) = \{Y(t_{k-1}), y(t_k) = y\}$. The EIG is essentially the expectation of the IG with respect to $y(t_k)$. Note that in the case of continuous detection results, the summation in (5.16) should be replaced by an integration.

5.3.1 The Linear-Gaussian Case

The measurement model for the linear-Gaussian case can be simply described by the following equation:

$$y(t_k) = H(t_k)x(t_k) + \nu(t_k), \quad (5.17)$$

where $\nu(t_k) \sim N(0, R(t_k))$.

Similar to the *time evolution*, the *measurement evolution* of the entropy of a linear-Gaussian target system can also be represented by means of the covariance matrix $P(\cdot)$ using the Kalman filter. In this case, (5.15) will simply reduce to

$$\begin{aligned}\Delta\varepsilon(x,t_k|u,Y(t_k)) &= \varepsilon(x,t_k|Y(t_{k-1})) - \varepsilon(x,t_k|u,Y(t_k)) \\ &= \log(\det P(t_k|t_{k-1})) - \log(\det P(t_k|t_k)). \quad (5.18)\end{aligned}$$

Recall that [62]

$$P(t_k|t_k) = [I - K_{t_k}H(t_k)]P(t_k|t_{k-1}), \quad (5.19)$$

where $K_{t_k} = P(t_k|t_{k-1})H^T(t_k)[H(t_k)P(t_k|t_{k-1})H^T(t_k) + R(t_k)]^{-1}$.

We then have

$$\Delta\varepsilon(x,t_k|u,Y(t_k)) = \log(\det(I - K_{t_k}H(t_k))). \quad (5.20)$$

Remark 5.3.1: It is worth noting that the *information gain* in the linear-Gaussian case (5.20) is only determined by one's prior knowledge of the target system and the sensor, which is independent of the measurement $y(t_k)$.

Eq. (5.19) can also be re-written as [62]

$$P^{-1}(t_k|t_k) = P^{-1}(t_k|t_{k-1}) + H(t_k)^T R^{-1}(t_k) H(t_k), \tag{5.21}$$

which lead to the following conclusion.

Theorem 5.3.1: Let a linear-Gaussian system (5.3) be tracked independently by two sensor systems with the same configuration (i.e. identical $H(\cdot)$ and $R(\cdot)$ in (5.17)), except for their measurement sampling rate. Denote $\varepsilon_1(x,t|Y(t))$ and $\varepsilon_2(x,t|Y(t))$ as the entropies of the track PDFs maintained by two sensor systems. It is true that

$$\sup_{t_0 \le t \le \infty} \varepsilon_1(x,t|Y(t)) \ge \sup_{t_0 \le t \le \infty} \varepsilon_2(x,t|Y(t)), \tag{5.22}$$

if $f(t)$ in (5.3) in semi-positive definite and

$$\Delta T_1 \le \Delta T_2, \tag{5.23}$$

where $\frac{1}{\Delta T_1}$ and $\frac{1}{\Delta T_2}$ are the measurement sampling rates of the two sensor systems, respectively.

Proof: Let $Y_1(t)/Y_2(t)$ be the sets of measurements obtained by the two sensor systems, and $P_1(t|t)/P_2(t|t)$ be the covariance matrices maintained by the two sensor systems respectively.

Consider the first estimation cycle of the Kalman filter in each sensor system (i.e $k = 0$). According to *Theorem 5.2.1* and (5.23), we have

$$\sup_{t_0 + k\Delta T_1 \le t < t_0 + (k+1)\Delta T_1} \varepsilon_1(x,t|Y_1(t))$$

$$
\begin{aligned}
&= \varepsilon_1(x, t_0 + (k+1)\Delta T_1 | Y_1(t_0 + k\Delta T_1)) \\
&\propto \log(\det(P_1(t_0 + (k+1)\Delta T_1 | t_0 + k\Delta T_1))) \\
&\leq \log(\det(P_2(t_0 + (k+1)\Delta T_2 | t_0 + k\Delta T_2))) \\
&\propto \varepsilon_2(x, t_0 + (k+1)\Delta T_2 | Y_2(t_0 + k\Delta T_2)) \\
&= \sup_{t_0 + k\Delta T_2 \leq t < t_0 + (k+1)\Delta T_2} \varepsilon_2(x, t | Y_2(t)).
\end{aligned} \tag{5.24}
$$

Since the two sensor systems have the same measurement model $H(\cdot)$ and noise model $R(\cdot)$, we have the following inequality based on (5.21):

$$
\det(P_1(t_0 + \Delta T_1 | t_0 + \Delta T_1)) \leq \det(P_2(t_0 + \Delta T_2 | t_0 + \Delta T_2)). \tag{5.25}
$$

Note that $P_1(t_0 + \Delta T_1 | t_0 + \Delta T_1)$ and $P_2(t_0 + \Delta T_2 | t_0 + \Delta T_2)$ are also the initial co-variance matrices for the two system in the next estimation cycle. Combing the result above with *Corollary 5.2.2*, we now know that if (5.24) holds for the k^{th}, $k \geq 0$ estimation cycle, it also holds for the $(k+1)^{th}$ estimation cycle, which proves (5.22).

Remark 5.3.2: *Theorem 5.3.1* indicates that the maximum entropy of a target system is inversely proportional to the measurement sampling rate. It is worth noting that in the MSA-target scenario, the measurement sampling rate is not merely a intrinsic parameter of the on-board sensor system. Since there is possibly a non-trivial transition time between two measurements duo to the physical constraints on MSAs, the measurement sampling rate is dependent on how the MSAs move.

5.3.2 The Non-linear Case

Since the BF-HMap method introduced in Chapter 3 applies to general non-linear target systems and MSAs, which also takes the effect of a limited FOV and false

alarm/miss detection into account, we use this method in this study to analyze the *measurement evolution* of the entropy in the non-linear case.

Let $p^-(x,t_k)$ and $p^+(x,t_k)$ be the target track PDF before and after the arrival of measurement $y(t_k)$. We can then rewrite the measurement update equation in the BF-HMap algorithm derived in Chapter 3 as follows:

$$p^+(x,t_k) = \begin{cases} \frac{P_0}{A_F C_D} p^-(x,t_k), & \text{if } y(t_k) \neq \emptyset \text{ and } x \notin F(u); \\ \frac{(1-P_1)}{C_D} p^-(x,t_k) p^*(y(t_k)|x), & \text{if } y(t_k) \neq \emptyset \text{ and } x \in F(u); \\ \frac{P_1}{C_U} p^-(x,t_k), & \text{if } y(t_k) = \emptyset \text{ and } x \in F(u); \\ \frac{1-P_0}{C_U} p^-(x,t_k), & \text{if } y(t_k) = \emptyset \text{ and } x \notin F(u); \end{cases} \tag{5.26}$$

where

$$C_D = \frac{P_0}{A_F}[1 - \int_F p^-(x,t_k)dx] + (1-P_1)\int_F p^-(x,t_k)p^*(y(t_k)|x)dx, \tag{5.27}$$

$$C_U = P_1 \int_F p^-(x,t_k)dx + (1-P_0)[1 - \int_F p^-(x,t_k)dx], \tag{5.28}$$

Here $F(u)$ denotes region covered by the FOV of the MSA, which is resulted from a motion-control decision u. $x \in F(u)$ means that the positional components of the target state vector x is inside $F(u)$. A_F is the area of $F(u)$, which is a constant. P_0 and P_1 are the pre-known probabilities of false alarm and miss detection as defined in (3.9) and (3.10), respectively. $p^*(y(t_k)|x)$ is the probability density function of a non-empty measurement $y(t_k)$ given the true state of the target $x(t_k) = x$, which is basically the distribution of the measurement noise $\nu(t_k)$.

Let $\varepsilon^-(x,t_k|u,y(t_k))$ and $\varepsilon^+(x,t_k|u,y_k)$ be the entropy of the target track PDF before and after the arrival of measurement $y(t_k)$ (which is resulted from sensor action u). For the sake of simplification, we will use the discrete-space version of the BF-HMap method here to calculate the change of the entropy. We will also omit the time stamp t_k in the rest of this section for the purpose of notation convenience.

Eq. (5.26) shows that $p^+(x)$ looks very different whether there is measurement is an empty set or not. Thus, we treat these two cases separately as follows.

Case 1: $y \neq \emptyset$

In this case, we can utilize the first two rows of (5.26) and get

$$\begin{aligned}
& \varepsilon^+(x|u, y, y \neq \emptyset) \\
= & -\int P^+(x) \log P^+(x) dx \\
= & -\int_{F(u)} \frac{1-P_1}{p(y, y \neq \emptyset)} P^-(x) p(y|x) \log \left[\frac{1-P_1}{p(y, y \neq \emptyset)} P^-(x) p(y|x) \right] dx \\
& -\int_{F(\bar{u})} \frac{P_0 P^-(x)}{p(y, y \neq \emptyset) A_F} \log \frac{P_0 P^-(x)}{p(y, y \neq \emptyset) A_F} dx \\
= & -\frac{1-P_1}{p(y, y \neq \emptyset)} \int_F P^-(x) p(y|x) \log[P^-(x) p(y|x)] dx \\
& -\frac{1-P_1}{p(y, y \neq \emptyset)} \log \frac{1-P_1}{p(y, y \neq \emptyset)} \int_{F(u)} P^-(x) p(y|x) dx \\
& -\frac{P_0}{p(y, y \neq \emptyset) A_F} \int_{F(\bar{u})} P^-(x) \log P^-(x) dx \\
& -\frac{P_0[1-P_F(u)]}{p(y, y \neq \emptyset) A_F} \log \frac{P_0}{p(y, y \neq \emptyset) A_F}.
\end{aligned} \tag{5.29}$$

where $P_F(u) = \int_{F(u)} P^-(x)$.

Thus, we can further get the expected entropy resulted from a non-empty measurement as follows.

$$\begin{aligned}
& \varepsilon^+(x|u, y \neq \emptyset) \\
= & -\int p(y|y \neq \emptyset) \varepsilon^+(x|y, y \neq \emptyset) dy \\
= & \int_{F(u)} \int_{F(u)} \frac{1-P_1}{P_D} P^-(x) p(y|x) \log \left[\frac{1-P_1}{p(y, y \neq \emptyset)} P^-(x) p(y|x) \right] dx dy \\
& -\int_{F(u)} \int_{F(\bar{u})} \frac{P_0 P^-(x)}{P_D A_F} \log \frac{P_0 P^-(x)}{p(y, y \neq \emptyset) A_F} dx dy \\
= & -\frac{1-P_1}{P_D} \int_{F(u)} \int_{F(u)} P^-(x) p(y|x) \log p(y|x) dx dy
\end{aligned}$$

$$
\begin{aligned}
&-\frac{1-P_1}{P_D}\int_{F(u)} P^-(x)\log P^-(x)dx \\
&+\frac{1-P_1}{P_D}\int_{F(u)}\int_{F(u)} P^-(x)p(y|x)\log p(y|D)dxdy \\
&+\frac{(1-P_1)P_F(u)}{P_D}\log P_D - \frac{P_F(u)(1-P_1)}{P_D}\log(1-P_1) \\
&-\frac{P_0}{P_D}\int_{F\bar{(}u)} P^-(x)\log P^-(x)dx - \frac{P_0[1-P_F(u)]}{P_D}\log\frac{P_0}{A_F} \\
&+\frac{P_0[1-P_F(u)]}{P_D A_F}\int_{F(u)}\log p(y|D)dy + \frac{P_0[1-P_F(u)]}{P_D}\log P_D, \qquad (5.30)
\end{aligned}
$$

where

$$
P_D \equiv p(y\neq\emptyset) = P_0[1-P_F(u)] + (1-P_1)P_F(u). \qquad (5.31)
$$

Case 2: $y=\emptyset$

In this case, we have to use the last two rows of (5.26) to compute $\varepsilon^+(x|u,y=\emptyset)$ as follows:

$$
\begin{aligned}
& \varepsilon^+(x|u,y=\emptyset) \\
= & -\int P^+(x)\log P^+(x)dx \\
= & -\int_{F(u)}\frac{P_1}{P_U}P^-(x)\log\frac{P_1P^-(x)}{P_U}dx \\
& -\int_{F\bar{(}u)}\frac{1-P_0}{P_U}P^-(x)\log\frac{(1-P_0)P^-(x)}{P_U}dx \\
= & -\frac{P_1}{P_U}\int_{F(u)} P^-(x)\log P^-(x)dx - \frac{P_1P_F(u)}{P_U}\log\frac{P_1}{P_U} \\
& -\frac{1-P_0}{P_U}\int_{F\bar{(}u)} P^-(x)\log P^-(x)dx \\
& -\frac{(1-P_0)[1-P_F(u)]}{P_U}\log\frac{1-P_0}{P_U}, \qquad (5.32)
\end{aligned}
$$

where

$$
P_U \equiv p(y=\emptyset) = 1-P_D = P_0[1-P_F(u)] + (1-P_1)P_F(u). \qquad (5.33)
$$

Based on (5.30) and (5.32), we can then obtain the EIG of control action u:

$$\begin{aligned}
\Delta\varepsilon(x|u) &= \varepsilon^-(x) - P_D\varepsilon^+(x|u, y \neq \emptyset) - P_U\varepsilon^+(x|y = \emptyset) \\
&= (1 - P_1)\int_{F(u)}\int_{F(u)} p^-(x)p(y|x)\log p(y|x)dxdy - P_0[1 - P_F(u)]\log A_F \\
&\quad -P_D\int_{F(u)} p(y|D)\log p(y|D)dy \\
&\quad +\mathbf{E}(\{P_D, P_U\}) - P_F(u)\mathbf{E}(\{P_1, 1 - P_1\}) + [1 - P_F(u)]\mathbf{E}(\{P_0, 1 - P_0\})\} \\
&= \Delta\varepsilon_m(u) + \Delta\varepsilon_d(u) - \Delta\varepsilon_{miss}(u) - \Delta\varepsilon_{false}(u). \qquad (5.34)
\end{aligned}$$

where $\mathbf{E}(\cdot)$ stands for the entropy function, and

$$\begin{aligned}
\Delta\varepsilon_m(u) &= (1 - P_1)\int_{F(u)}\int_{F(u)} p^-(x)p(y|x)\log p(y|x)dxdy \\
&\quad -P_0[1 - P_F(u)]\log A_F - P_D\int_{F(u)} p(y|D)\log p(y|D)dy, \qquad (5.35) \\
\Delta\varepsilon_d(u) &= \mathbf{E}(\{P_D, P_U\}), \qquad (5.36) \\
\Delta\varepsilon_{miss}(u) &= P_F(u)\mathbf{E}(\{P_1, 1 - P_1\})\}, \qquad (5.37) \\
\Delta\varepsilon_{false}(u) &= [1 - P_F(u)]\mathbf{E}(\{P_0, 1 - P_0\})\}. \qquad (5.38)
\end{aligned}$$

We call the above four components on the right side of (5.34) as the the information gain from the measurement, the information gain from detection, and the information losses due to miss-detection and false alarm, respectively.

In particular, if the sensor system is extremely noisy, the measurement probability density function can be simplified as uniformly distributed in the FOV, i.e. $p(y|x) = \frac{1}{A_F}$. In this case, it can be easily shown that $\varepsilon_m(u) = 0$, and (5.34) becomes

$$\begin{aligned}
&\Delta\varepsilon(x|u) \\
= \;& \Delta\varepsilon_d(u) - \Delta\varepsilon_{miss}(u) - \Delta\varepsilon_{false}(u) \\
= \;& \mathbf{E}(\{P_D, P_U\}) - P_F(u)\mathbf{E}(\{P_1, 1 - P_1\}) + [1 - P_F(u)]\mathbf{E}(\{P_0, 1 - P_0\})\}. \qquad (5.39)
\end{aligned}$$

Remark 5.3.3: Eq. (5.39) is essentially the EIG of a binary detection event, in which the sensor can only tell whether a target is inside its FOV or not. This simplified version of EIG can be used in applications that an accurate localization is not the first priority of sensor management. For instance, in a target search problem, the outcome of a sensor action is commonly simplified as a binary event, which means the target is either found or not. Therefore, (5.39) is sufficient enough to quantify the information gain in this case.

Another extreme case is that the sensor is perfect, which means there is no measurement noise, no miss-detection or false alarm (i.e. $P_0 = P_1 = 0$). Thus, $p(y|x)$ can be simplified as a pulse function, i.e. $p(y|x) = \delta(y - x)$. The corresponding EIG in this case will reduces to:

$$\begin{aligned} \Delta\varepsilon(x|u) &= \Delta\varepsilon_m(u) + \Delta\varepsilon_d(u) \\ &= -\int_{F(u)} p^-(x) \log p^-(x) dx - (1 - P_F(s)) \log(1 - P_F(s)). \end{aligned} \quad (5.40)$$

Remark 5.3.4: In reality, the sensor is neither extremely noisy, nor perfect, which requires a huge computational load to compute the EIG of an sensor action. Nevertheless, (5.39) and (5.40) can be considered as the lower bound and the upper bound of the EIG for an MSA, which can be helpful in developing MSA control methods in practice.

Eq. (5.34), (5.39) and (5.40) give us a general picture of how the EIG evolves with respect to the control actions of an MSA. As one may expect, the EIG here is a function of both the accuracy and coverage of the sensor (e.g. noise level, the size of FOV) and the MSA's kinematic state. Based on these results, one can mathematically quantify the effect of both the intrinsic and the extrinsic configurations of the MSAs the overall sensing performance, as well as evaluate the feasibility of the heuristics

that is used to formulate the objective function of MSA management. In fact, there are some interesting properties of the EIG that can be derived from (5.34), (5.39) and (5.40), which are not quite the same as one's intuitions.

For instance, in a target search problem, the "gain" of a search action is often formulated to be proportional to the probability of the existence of the target in the FOV, i.e. $P_F(u)$. Many optimal search methodologies have been developed by researchers based on this heuristic. However, this is not always the case in reality. One simple counterexample is shown in Fig. 5.1. In this example, the target state $x(t)$ is assumed to be distributed in a one-dimensional, discrete space with

$$p^-(x) = \begin{cases} 0.08 & x = 1, 2, 3, 4, 5; \\ 0.3 & x = 9, 10; \\ 0 & \text{otherwise.} \end{cases} \tag{5.41}$$

Assume that an MSA, equipped with a perfect sensor, is searching for the target with a FOV of width 5.1. If we choose objective function as the detection probability ($P_F(u)$), the maximum detection probability that one can achieve through a one step action is 0.6, which means that the FOV should be put somewhere around 9 and 10, as shown in Fig. 5.1. Based on (5.40), the EIG of such an action is 0.085 in this case. However, if the MSA put its FOV on top of region [1, 5] as shown Fig. 5.1, the resulting EIG is 0.313, which is much larger than the previous one. This example shows that even with a perfect sensor, the maximum detection probability criterion does not necessary lead to the optimal sensing result. From the information-theoretic point of view, this phenomenon does make sense. Although it is more likely to detect target by looking at the region with a larger detection probability, the stake is that if the target is not detected, the resulting uncertainty of the target state will still be quite significant, which is uniformly distributed in region [1, 5] in this example. On

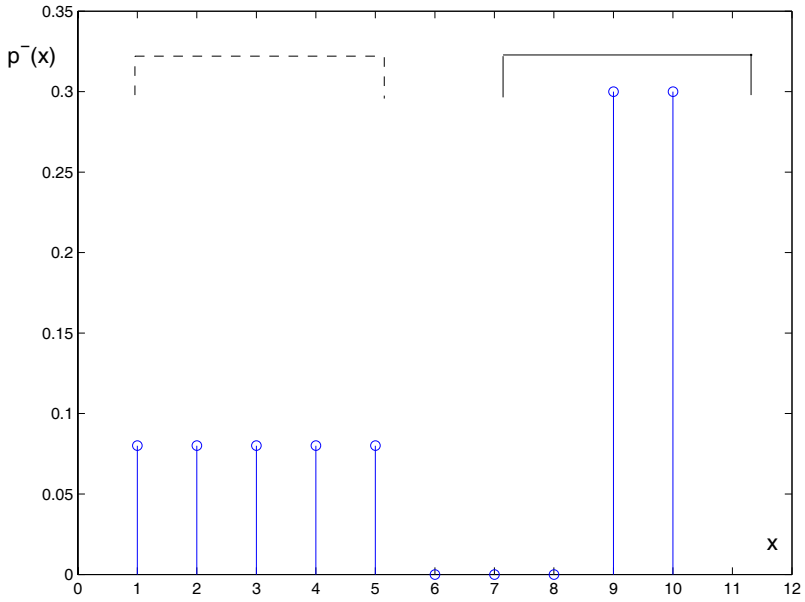


Figure 5.1: A simple example against the maximum detection probability criterion.

the other hand, by putting its FOV in region [1, 5], an MSA will be quite certain about where the target is no matter the target is inside its FOV or not.

5.4 Entropy-based Formulations of the Sensor Management Problems for MSAs

The analysis above shows that the evolution of the entropy in most cases follows a saw-toothed pattern, which tends to increase over time until it is dragged down by new measurements obtained by the MSAs as shown in Fig. 5.2. Since the entropy here is essentially a measure of the uncertainty of a target system, by moving around and collecting measurements, the MSAs can control the evolution of the entropy, and thus to control the the uncertainty of the targets of interest.

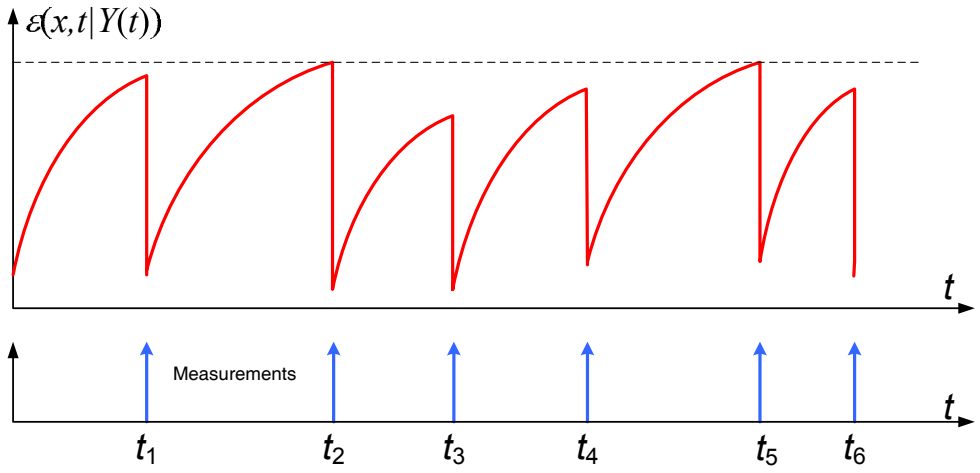


Figure 5.2: An illustration of the evolution of the entropy.

Therefore, we can interpret various MSA management problems into the language of entropy evolution. In this section, we will take target search and target tracking as two examples to study how to formulate the MSA motion control problem based on the evolution of the entropy.

5.4.1 Case of Study 1: Target Search

The main task of target search is usually to locate the position of a target. If we temporarily ignore the measurement noise of the sensor, $p(x,t|Y(t))$ becomes a pulse function when the target is found by the MSAs. Consequently, $e^{\varepsilon x,t|Y(t)}$ goes to zero according to (5.1). Thus, from the control point of view, to search for a moving target by a team of MSAs is equivalent to **stabilizing the system** $e^{\varepsilon(x,t|Y(t))}$. In a similar way, the search problem in the discrete-space case can be translated to a stabilization problem for $\varepsilon(x,t|Y(t))$.

As pointed out in Chapter 2, due to the effect of the limited FOVs of the MSAs, the evolution of $p(x,t|Y(t))$ as well as that of the entropy $\varepsilon(x,t|Y(t))$ is non-linear and non-Gaussian even both the target model and the measurements are linear and Gaussian. As a result, it is almost impossible to solve the control problem for the MSAs with respect to $\varepsilon(x,t|Y(t))$ directly. On the other hand, we have shown in section 5.2 that the entropy is upper-bounded by the log of the area of the region with non-zero $p(x,t|Y(t))$, which can be considered as an alternative metric of uncertainty especially in the target search problem. Obviously, $A(S(t|t))$ also converges to zero when $e^{\varepsilon(x,t|Y(t))}$ is stabilized. Therefore, we can further translate the search problem to a stabilization problem of the system $A(S(t|t))$.

Now, let us take the imperfection of sensors into account. With sensor error and noise, $e^{\varepsilon x,t|Y(t)}$ (or $\varepsilon x,t|Y(t)$ for the discrete-space case) will not be zero when the target is detected. Nevertheless, the objective of MSA control still is to reduce the entropy as much as possible. The desired evolution pattern of the entropy is like a diminishing saw-toothed curve, as shown in Fig. 5.3. If the measurement sampling rate is high enough, the search problem can still be treated as a stabilization problem of the entropy (the equilibrium is not zero, though).

In Chapter 6, we will study the target search problem based on this information-theoretic formulation in more detail.

5.4.2 Case of Study 2: Target Surveillance

Unlike a target search task, the job of target surveillance does not end once the target is detected. In addition to cut down the entropy as much as possible at each prediction-and-estimation cycle, we often want to keep the entropy below a given

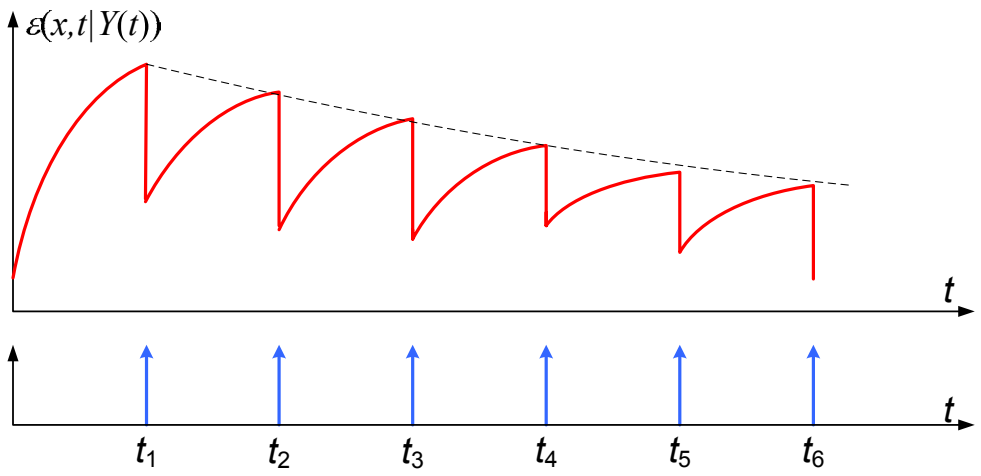


Figure 5.3: The desired evolution pattern of the entropy in target search.

threshold as well. Such a bounded entropy not only keeps the tracking errors at a desired level, but also allows the motion controller of an MSA to simplify its motion planning strategies. For example, by keeping the extent of non-zero $p(x,t|Y(t))$ smaller than the FOV of an MSA (which also means $\varepsilon(x,t|Y(t)) \leq \log A_F$), what the motion controller has to do to take a non-empty measurement is just to drive the MSA on top of the cloud of $p(x,t|Y(t))$. Thus, the motion planning problem in between two measurement cycles reduces to a conventional point-to-point trajectory design problem. It is worth noting that, if the the cloud of the target PDF is smaller than the FOV of an MSA, $\Delta\varepsilon_b$, $\Delta\varepsilon_{miss}$ and $\Delta\varepsilon_{false}$ in (5.34) will cancel with other. Thus, the EIG in the non-linear case will reduce to the information gain from the measurement ($\Delta\varepsilon_m$) only, which is very similar to the linear-Gaussian case. As a result, the argument that the maximums of the entropy (i.e. the peaks in Fig. 5.2) are determined by the measurement sampling rate still holds for the non-linear case.

Thus, in order to keep the maximums of the entropy as low as possible, an MSA has to make the measurement sampling rate as high as possible. Considering the transition time between two consecutive measurements caused by motion and coverage constraints on an MSA, the best motion plan is the one that generates the time-optimal path for the MSA between two observations. In the case of monitoring multiple targets, the desired motion plan becomes a time-optimal path that traverses the targets. We will apply this entropy-based formulation to the more general multi-MSA-multi-target (MMMT) case and study this multi-MSA motion planning problem for multi-target surveillance in more detail in Chapter 7.

CHAPTER 6

COOPERATIVE SEARCH FOR A MOVING TARGET BY MULTIPLE MOBILE SENSOR AGENTS: THE NON-ESCAPE CASE

6.1 Introduction

Target search is a fundamental problem in automatic control and operations research. The mathematical foundations of modern search theory originate from the early work of Koopman [71], Stone [66], Brown [73], Tierney [74] Washburn [75] and others. In these approaches, the search problem was formulated as an optimization problem, whose objective is to maximize the payoff in a given finite period of time by distributing the sensing effort across the field of interest. A major disadvantage of search theory is that very few physical characteristics of the sensors have been taken into consideration. The sensing effort was assumed to be both infinitely divisible and arbitrarily allocatable in the space domain, which makes it very difficult to implement those early search algorithms in practice. In particular, there has been increasing interest to performing target search or other similar tasks using a group of mobile sensor agents (MSA) such as Unmanned Air Vehicles (UAVs) in recent years. Because of the physical constraints on the MSAs, the sensing effort is neither infinitely divisible, nor arbitrarily allocatable in the space domain. As a result, the search problem is not

merely a mathematical optimization problem, but rather a motion control problem. During the last few years, numerous cooperative control strategies and methods have been reported by researchers for target search by multiple MSAs [23,24,77,78,81–88]. Most of these attempts are seeking optimal control solutions to variations of the search problem addressed in search theory. In the meantime, there are several other intriguing questions that have not been paid the attentions that they deserve. For instance, it is often crucial to assess the load of a search task and the sufficiency of a given amount of MSA resources first. Subsequently, one can then decide whether the search strategy should be greedy or conservative, short-term or long-term.

In this chapter, we investigate the search problem from the resource management point of view. Unlike most related work, which often focuses on maximizing a payoff function (e.g. probability of detection) within a fixed amount of time, we seek theoretic boundaries for non-escape search with respect to the configuration of the search problem and the characteristics of the MSAs.

The study in the previous chapter shows that searching for a moving target is equivalent to stabilize the entropy of the target track PDF, or to stabilize the area of the non-zero portion of the PDF in the case of perfect sensors. Based on this consideration, a necessary condition in terms of the number of MSAs to possibly fulfill a search task in finite time is derived. In other words, we provide a threshold M_{min}, such that there is always a non-zero probability of escape if the number of MSAs is less than M_{min}. Meanwhile, a sufficient condition in terms of the number of MSAs 1to guarantee a non-escape search is also derived. In addition, a cooperative search formation, the Progressively-Spiral-In (PSI) algorithm, for a multi-MSA team

to realize a non-escape search in finite time is presented. Both the upper bound and the lower bound for a non-escape search are extended to heterogenous MSAs as well.

The rest of the chapter is organized as follows: section 6.2 lists the notation that is used in this chapter and the assumptions that have been made. Then, some basic concepts in curve expansion and curve contraction are introduced in section 6.3. After that, an information-theoretic formulation of the search problem is proposed in section 6.4. The major results of this chapter, a lower bound a team of MSAs to fulfill a non-escape search task is presented in section 6.5, followed by the PSI algorithm and a upper-bound of a non-escape search in section 6.6. Section 6.7 discusses the applications of the main results of this paper in different search scenarios. Finally, conclusions are given in section 6.8.

6.2 Notations and Assumptions

In this section, some basic notations and assumptions on the MSAs and the targets that are frequently used throughout this chapter are introduced.

6.2.1 Notations

- $\mathcal{R}(\mathcal{R}^+)$: Set of real (positive real) numbers;
- $\mathcal{Z}$: Set of integers;
- $\lceil\cdot\rceil : \mathcal{R} \to \mathcal{Z}$: $\forall r \in \mathcal{R}$, $\lceil r \rceil = \min_z\{z \geq r, z \in \mathcal{Z}\}$;
- $\|\cdot\| : \mathcal{R}^2 \to \mathcal{R}$: $\forall o = [o_x, o_y]^T \in \mathcal{R}^2$, $\|o\| = \sqrt{o_x^2 + o_y^2}$;
- $A(S)$: The area of S, $S \subset \mathcal{R}^2$;
- $C(S)$: The boundary of S, $S \subset \mathcal{R}^2$;

- $L(C)$: The length of curve C;
- $d(x_1, x_2)$: The Euclid distance between two points x_1, x_2, where $x_1, x_2 \in \mathcal{R}^2$;
- $d(x, S)$: The distance between a point $x \in \mathcal{R}^2$ and a point set $S \subset \mathcal{R}^2$, where

$$d(x, S) = \min_{x' \in S}\{d(x, x')\}. \tag{6.1}$$

- $\tilde{d}_C(x_1, x_2)$: The distance between two points along curve C. In other words, it is the length of the segment of curve C between x_1 and x_2. In particular, if C is a closed curve, the clockwise curve segment from x_1 to x_2 is taken into account.
- $\overline{d}_{C_1}(C_2)$: The maximum of the minimum distance from C_2 to C_1:

$$\overline{d}_{C_1}(C_2) : \max_{x_1 \in C_1}\{\min_{x_2 \in C_2} d(x_1, C_2)\}, \quad C_1, C_2 \subset \mathcal{R}^2. \tag{6.2}$$

- $\phi_l(C, x)/\phi_r(C, x)$: The left-hand/right-hand tangent vector of curve C, given that C is at least directionally differentiable at x. Note that if C is smooth at x, $\phi_l(C, x) = \phi_r(C, x)$.
- $\kappa(C, x)$: The curvature of curve C at point x, given that C is smooth at x.
- $\perp$: Consider a straight line $C_a \subset \mathcal{R}^2$ and a curve $C_b \subset \mathcal{R}^2$, $C_a \perp C_b$ if and only if C_a and C_b only has one intersection x, where C_b is smooth and C_a is perpendicular to the tangent of C_b at x.

6.2.2 The MSA Model

Denote $s^i = [x_s^i, y_s^i]^T \in \mathcal{R}^2$ and θ_s^i as the 2-D position and heading of the i^{th} MSA. The kinematics of each MSA is simply modeled as :

$$\begin{cases} \dot{x}_s^i = v_s^i \cos\theta_s^i \\ \dot{y}_s^i = v_s^i \sin\theta_s^i \\ \dot{\theta}_s^i = u^i \end{cases} \tag{6.3}$$

where u^i is the controller on i^{th} MSA and v_s^i is its the cruise speed.

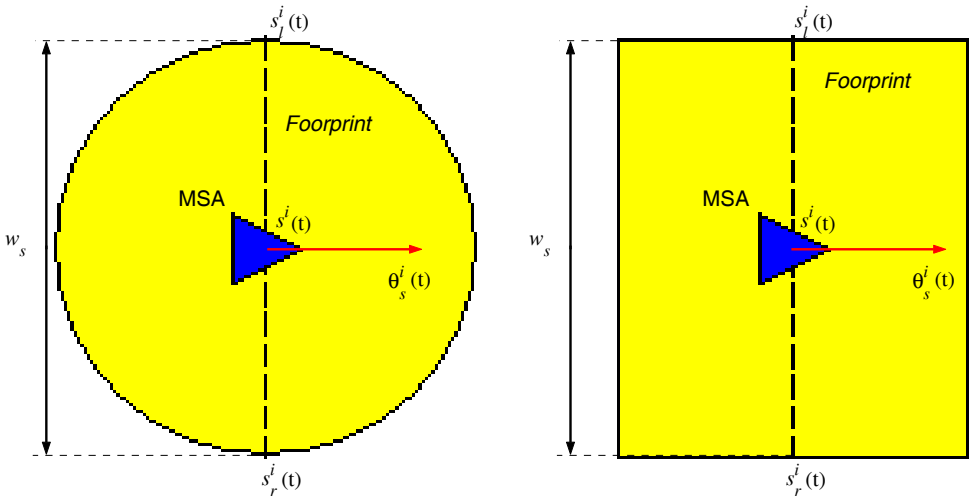
Each MSA is assumed to be equipped with an on-board sensor. The sensor has a limited field of view (i.e a footprint) centered at the MSA (Fig. 6.1). The following notation is used to characterize the MSAs.

- $F^i(t)$: The closed region covered by the footprint of the i^{th} MSA;
- $F(t)$: The union of the footprints of all the MSAs, i.e. $F(t) = \bigcup F^i(t), i = 0..M-1$;
- A_s^i: The area of the footprint of the i^{th} MSA;
- L_s^i: The perimeter of the footprint of the i^{th} MSA;
- v_s^i: The cruise speed of the i^{th} MSA;
- w_s^i: The width of the sensor footprint, or the maximum span of the footprint that is orthogonal to the heading of the i^{th} MSA (Fig. 6.1).
- $s_r^i(t)$ and $s_l^i(t)$: The right-hand and left-hand ends of the longitudinal span of the footprint of the i^{th} MSA (Fig. 6.1).

For the purpose of simplicity, it is assumed that $s^i(t)$, $s_l^i(t)$ and $s_l^i(t)$ are collinear (Fig. 6.1). It is worth noting that the traveling distances of $s^i(t)$, $s_r^i(t)$ and $s_l^i(t)$ have the following relations.

Lemma 6.2.1: Denote C, C_r, and C_l as the trajectories of $s^i(t)$, $s_r^i(t)$ and $s_l^i(t)$ from t_a to t_b, respectively. If C_r is convex and $\overrightarrow{s_r^i(t)s_l^i(t)}$ always points to its outward normal directions (as shown in Fig. 6.2), then

$$L(C) = L(C_r) + \frac{w_s^i}{2}(\theta_s^i(t_a) - \theta_s^i(t_b)), \tag{6.4}$$



(a) A circular footprint (b) A rectangular footprint

Figure 6.1: Examples of an MSA and its footprint.

and

$$L(C_l) = L(C_r) + w_s^i(\theta_s^i(t_a) - \theta_s^i(t_b)). \tag{6.5}$$

In particular, if C_r is a closed convex curve, we have

$$L(C) = L(C_r) + \pi w_s^i, \tag{6.6}$$

and

$$L(C_l) = L(C_r) + 2\pi w_s^i. \tag{6.7}$$

Proof: Consider an arbitrarily small segment of C and its counterparts on C_r, which are denoted as δC and δC_r, respectively (Fig. 6.2).

Since both segments are arbitrarily, they can be treated as two segments of arcs. Thus, we have

$$L(\delta C) = r\delta\theta = r_r\delta\theta + \frac{w_s^i}{2}\delta\theta = L(\delta C_r) + w_s^i\delta\theta + o(\delta\theta), \tag{6.8}$$

where r and r_r are the turning radius of arc δC and δC_r, respectively.

By integrating both sides of (6.8) from t_a to t_b, we get (6.4). In a similar way, we can get (6.5), (6.6) and (6.7).

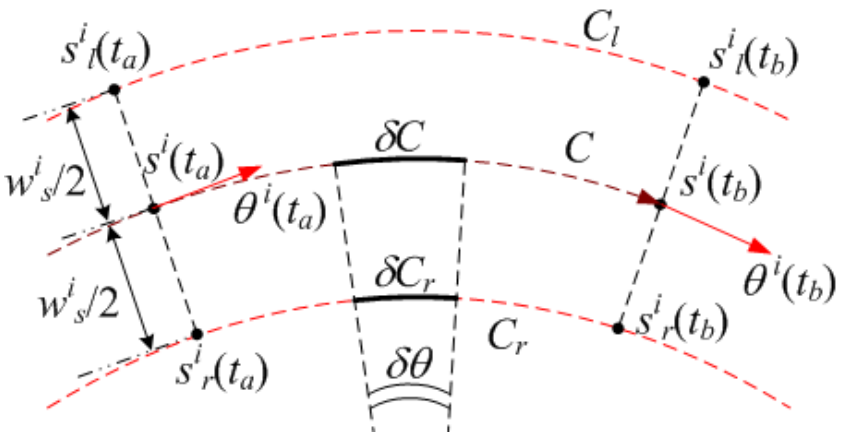


Figure 6.2: Trajectories traveled by an MSA and two longitudinal ends of its footprint.

The sensor is assumed to be reliable, i.e. a target is detected by an MSA if and only if it is inside the footprint of the MSA. In the majority of this chapter, it is further assumed that the MSAs are homogeneous, which means $A^i_s \equiv A_s$, $L^i_s \equiv L_s$, $w^i_s \equiv w_s$, and $v^i_s \equiv v_s$ for $i = 0..M-1$.

6.2.3 The Target Model

Let $x_{tgt} = [\mathrm{x}, \mathrm{y}]^T \in \mathcal{R}^2$ stands for the 2-D position of the target. The motion of the target is modeled as a plenary random process:

$$\begin{cases} \dot{\mathrm{x}} = \upsilon(t)\cos\vartheta(t) \\ \dot{\mathrm{y}} = \upsilon(t)\sin\vartheta(t) \end{cases} \tag{6.9}$$

where $\upsilon(t)$ is uniformly distributed in $[0, v_t]$ and $\vartheta(t)$ is uniformly distributed in $[0, 2\pi)$, which means the target can move in any direction with speed up to v_t.

We also use $p_{tgt}(x, t|Y(t'))$ to represent the posterior probability density function of $x_{tgt}(t) = x$ given $Y(t')$, where $Y(t')$ is the collection of all the observations of the MSAs up to time instant $t' \leq t$.

Definition 6.2.1: The *Survival Zone* is a region $S(t|t') \subset \mathcal{R}^2$ with non-zero $p_{tgt}(x, t|Y(t'))$:

$$S(t|t') = \bigcup \{x; x \in \mathcal{R}^2, p_{tgt}(x, t|Y(t')) > 0\}. \tag{6.10}$$

Thus, according to the analysis in Chapter 5, to search for a moving target is equivalent to stabilize the area of the *survival zone* of the target.

6.3 The Evolution of Plane Curves

The concept of curve expansion and contraction originates from the domain of differential geometry, and has been used in various areas such as physics, biology, chemical kinetics and computer vision [90–92, 94]. In this section, we will introduce some basic concepts on the expansion and contraction of 2-D curves, which will be useful in deriving the main results of this paper.

Definition 6.3.1: A continuous curve $C \subset \mathcal{R}^2$ is *smooth almost everywhere* if it is smooth except for a finite number of points. These non-smooth points are called the *corners* of C.

Note that a convex curve is at least *smooth almost everywhere* [95].

Definition 6.3.2: Consider a simple (i.e. no self-crossing), closed, continuous curve $C \subset \mathcal{R}^2$, which is smooth almost everywhere. A curve $C' \subset \mathcal{R}^2$ is called an *isotropic*

expansion of C by d', or $C' = \mathcal{IE}(C, d')$, if

$$\forall x \in C', \quad d(x, S) = d', \tag{6.11}$$

where S is the region enclosed by C. In the meantime, a curve $C'' \subset \mathcal{R}^2$ is called an *isotropic contraction* of C , or $C'' = \mathcal{IC}(c, d'')$, if

$$\forall x \in C'', \quad d(x, \bar{S}) = d'', \tag{6.12}$$

where $\bar{S}$ is complement set of S in $\mathcal{R}^2$.

Definition 6.3.3: Consider a simple, closed continuous curve $C \subset \mathcal{R}^2$ and its *isotropic expansion* $C'' = \mathcal{IE}(C, d)$. $\forall q_0'' \in C''$, a point $q_0 \in C$ is called an *expansion seed* of q_0'' on C, or $q_0 = \rho^{\mathcal{IE}}(q_0'', C)$, if $d(q_0, q_0'') = d$ (Fig. 6.3). We also call q_0'' an *expanding point* of q_0 on C''.

Definition 6.3.4: Consider a simple, closed continuous curves $C \subset \mathcal{R}^2$ and its *isotropic contraction* $C' = \mathcal{IC}(C, d)$. $\forall q_0' \in C'$, a point $q_0 \in C$ is called a *contraction seed* of q_0' on C, or $q_0 = \rho^{\mathcal{IC}}(q_0', C)$, if $d(q_0, q_0') = \min\{d(x, q_0'); x \in C\}$ (Fig. 6.3). We also call q_0' a *contracting point* of q_0 on C'.

In particular, an *isotropic expansion* or *contraction* of a closed, convex curve C has the following properties:

- **P1**: Any *isotropic extension* of C is smooth and convex.
- **P2**: Any *isotropic contraction* of C is convex;
- **P3**: Let $d = \frac{1}{2}\min\{d(q, C); \forall q \in S\}$, where S is the region enclosed by C. $C' = \mathcal{IE}(C, d)$ is either a single point or a straight line segment.

- **P4**: Let $C' = \mathcal{IE}(C, d)$, $\forall d \in \mathcal{R}^+$. $\forall q' \in C'$ only has a unique *expansion seed* $q = \rho^{\mathcal{IE}}(q', C)$. In addition, we have $\overline{qq'} \perp C'$ and $\phi_r(q, C) \leq \phi_r(q', C') = \phi_l(q', C') \leq \phi_l(q, C)$.

- **P5**: Let $C' = \mathcal{IE}(C, d)$, $\forall d \in \mathcal{R}^+$. $\forall q \in C$, if q is not a corner, it only has a unique *expanding point* on C'. Denote this point as q', it is true that $\overline{qq'} \perp C$.

- **P6**: $\forall q \in C$, it has only one *contracting point* on any *isotropic contraction* of C.

- **P7**: Let $C' = \mathcal{IC}(C, d)$, $\forall d \in \mathcal{R}^+$. $\forall q' \in C'$, $\phi_r(q', C') \leq \phi_r(q, C) \leq \phi_l(q, C) \leq \phi_l(q', C')$, where $q = \rho^{\mathcal{IC}}(q', C)$.

- **P8**: Let $C' = \mathcal{IC}(C, d)$, $\forall d \in \mathcal{R}^+$. $\forall q' \in C'$, if q' is not a corner of C', it only has a unique *contraction seed* $q = \rho^{\mathcal{IC}}(q', C)$, which is not a corner of C, either. In addition, line $\overline{qq'} \perp C$ and $\overline{qq'} \perp C'$.

- **P9**: Let $C' = \mathcal{IC}(C, d)$, $\forall d \in \mathcal{R}^+$. $\forall q'_1 \in C'$ and $\forall q_1, q_3 \in C$, s.t. q_1 and q_3 are both *contraction seeds* of q'_1, it is true that $\forall q_2 \in \Delta C$ is also a *contraction seed* of q'_1, where ΔC stands for the segment of C between q_1 and q_3 (Fig. 6.3).

- **P10**: Let $C' = \mathcal{IC}(C, d)$, $\forall d \in \mathcal{R}^+$. $\forall q'_1 \in C'$, let q_l and q_3 be the first and the last *contraction seed* of q'_1 on C in the clockwise order, respectively as shown in Fig. 6.3. It is true that $\phi_l(C, q_1) = \phi_l(C', q'_1)$ and $\phi_r(C, q_3) = \phi_r(C', q'_1)$.

- **P11**: If $C' = \mathcal{IE}(C, d)$,

$$A(S') = A(S) + \frac{L(C) + L(C')}{2} d, \tag{6.13}$$

where S and S' stand for the region enclosed by C and C', respectively.

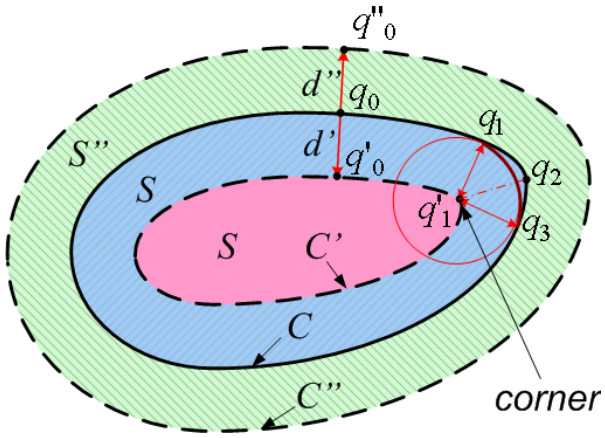


Figure 6.3: Expansion and contraction of a 2-D curve.

6.4 The Evolution of the Survival Zone $S(t|t)$ and its Area $A(S(t|t))$

The analysis in Chapter 5 indicates that the target search problem is equivalent to the stabilization problem of the area of the region with non-zero $p(x,t|Y(t))$, i.e. the *survival zone* $S(t|t)$ as defined in (6.10). In order to find an appropriate way to stabilize $A(S(t|t))$, we study how $S(t|t)$ and $A(S(t|t))$ evolves over time and changes with respect to MSA's motion in this section.

Compared with the entropy, $A(S(t|t))$ evolves in a much simpler way. Consider the change of the $S(t|t)$ of a target from any time instant t to $t+\delta t$ without any new measurement information. As (6.9) indicates, the *survival zone* expends in such a way that any $x \in S(t|t)$ will lead to a non-zero $p_{tgt}(x', t+\delta t|Y_t)$, as long as $d(x',x) \leq v_t \delta t$. In other words, $C(S(t+\delta t|t))$ is an *isotropic expansion* of $C(S(t|t))$ by $v_t \delta t$ (Fig. 6.5(a)). Based on (**P11**) in Section 6.3, we have

$$A(S(t+\delta t|t)) \;\; = \;\; A(S(t|t)) + \frac{L(C(S(t|t))) + L(C(S(t+\delta t|t)))}{2} v_t \delta t$$

$$= \quad A(S(t|t)) + L(C(S(t|t))v_t\delta t + o(\delta t). \tag{6.14}$$

Now, let us take the new observations of the MSAs from t to $t+\delta t$ into account. If the target is detected, $A(S(t+\delta t)) = 0$ and the search job is finished. Otherwise, the non-detection observations of the MSAs will eliminate the possibility of the existence of the target in the region covered by their footprints at current moment. In summary, we have:

$$\begin{aligned} & A(S(t+\delta t|t+\delta t)) \\ = & \begin{cases} 0 & \text{if the target is detected;} \\ A(S(t+\delta t|t)) - A(F(t+\delta t) \cap S(t+\delta t|t)) & \text{otherwise.} \end{cases} \end{aligned} \tag{6.15}$$

Since the focus of this paper is non-escape search, only the non-detection case (i.e. the worst case) is considered here. The resulting update equation of the *survival zone* then is

$$\begin{aligned} & A(S(t+\delta t|t+\delta t)) \\ = \quad & A(S(t|t)) + L(C(S(t|t)))v_t\delta t - A(F(t+\delta t) \cap \mathcal{IE}(S(t|t), v_t\delta t) + o(\delta t). \end{aligned} \tag{6.16}$$

Eq. (6.16) also lead to the following properties of $S(t|t)$ directly.

Lemma 6.4.1: $\forall t \geq t_0$,

$$F(t) \cap S(t|t) = \emptyset. \tag{6.17}$$

Lemma 6.4.2: $\forall x \in \mathcal{R}^2$, if $d(x, S(t|t)) > 0$,

$$x \notin S(t'|t) \quad \text{and} \quad x \notin S(t'|t'), \quad \forall t \leq t' < t + \frac{d(x, S(t|t))}{v_t}. \tag{6.18}$$

Lemma 6.4.3: $\forall x \in \mathcal{R}^2$, if $x \in S(t|t))$, it is true that $\forall t' \leq t, \quad \forall x' \in S(t'|t')$, there exists a feasible trajectory of the target $x^*(t)$ s.t. $x^*(t') = x'$ and $x^*(t) = x$.

Remark 6.4.1: It is worth noting that $A(S(t|t)) = 0$ does not necessarily indicate that the target is detected. A simple example for this extreme case is $S(t|t)$'s being a curve segment. Fortunately, since $F(t)$ is a closed set here, such a curve segment will not be a stable equilibrium for $S(t|t)$.

Lemma 6.4.4: Let an MSAs (denoted as MSA_0) travels along curve C with another MSA (denoted as MSA_1) chasing him through another trajectory C', as shown in Fig. 6.4. Denote $x^*(t)$, $t \in [t_a, t_b]$ as an arbitrary feasible trajectory of the target that crosses C through the gap between MSA_1 and MSA_0 (i.e. $x^*(t)$ intersects the segment of C between line $\overline{s_r^1(t)s_l^1(t)}$ and $s^0(t)$). It is true that

$$\begin{aligned} &\exists \tau \in [t_a, t_b], \quad s.t. \quad x^*(\tau) \in F(\tau), \\ &\text{if} \quad \frac{d(s^1(t), \hat{s}^1(t))}{v_t} + \frac{\tilde{d}_C(\hat{s}^1(t), s^0(t))}{v_s} \leq \frac{w_s}{v_t}, \quad \forall t \in [t_a, t_b], \end{aligned}$$

where $\hat{s}^1(t)$ is the intersection of C and line $\overline{s_r^1(t)s_l^1(t)}$.

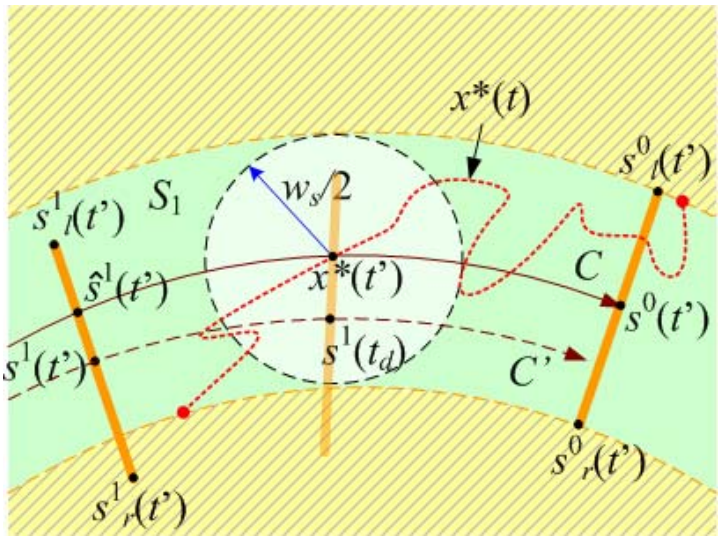


Figure 6.4: Illustration for *Lemma 6.4.4*.

Proof Let t' be any one of the time instants that $x^*(t)$ crosses C in between MSA_1 and MSA_0. Also, let t_c be the time instant that MSA_0 passes $x^*(t')$ (i.e. $s^0(t_c) = x^*(t') \in C$).

According to *Lemma 6.4.1* and *Lemma 6.4.2*, we know that

$$d(x^*(t_c), x^*(t')) \geq \frac{w_s}{2}, \tag{6.19}$$

i.e.

$$t' - t_c \geq \frac{w_s}{2v_t}. \tag{6.20}$$

Denote t_d as the time instant that line $\overline{s_r^1(t)s_r^i(t)}$ hits $x^*(t')$, i.e. $\hat{s}^1(t_d) = x^*(t')$. According to (6.19) and (6.20), we have

$$\begin{aligned} & \frac{d(s^1(t_d), \hat{s}^1(t_d))}{v_t} + t_d - t_c \leq \frac{w_s}{v_t} \\ \Rightarrow \quad & \frac{d(s^1(t_d), \hat{s}^1(t_d))}{v_t} + t_d - t' \leq \frac{w_s}{2v_t}. \end{aligned} \tag{6.21}$$

Thus, we have

$$\begin{aligned} d(s^1(t_d), x(t_d)) &\leq d(s^1(t_d), \hat{s}^1(t_d)) + d(\hat{s}^1(t_d), x(t_d)) \\ &= d(s^1(t_d), \hat{s}^1(t_d)) + (t_d - t')v_t \\ &\leq \frac{w_s}{2}, \end{aligned} \tag{6.22}$$

which means $x^*(t_d) \in F^1(t_d)$.

In particular, when MSA_1 follows the same trajectory as MSA_0 (i.e. $C' = C$), *Lemma 6.4.4* reduces to the following:

Lemma 6.4.5: Let MSA_1 travels along curve C following MSA_0. For any feasible target trajectory $x^*(t)$ that crosses C between $s^1(t)$ and $s^0(t)$, $t \in [t_a, t_b]$,

there exists a $\tau \in [t_a, t_b]$, s.t.

$$x^*(\tau) \in F(\tau), \quad \text{if} \quad \frac{\tilde{d}_C(s^1(t), s^0(t))}{v_s} \leq \frac{w_s}{v_t}, \quad \forall t \in [t_a, t_b]. \tag{6.23}$$

Lemma 6.4.4 can also be easily extended to the case with heterogenous MSAs as follows:

Lemma 6.4.6: Let MSA_0 travels along curve C with MSA_1 chasing him through curve C'. For any feasible target trajectory $x^*(t)$ that crosses C between $s^1(t)$ and $s^0(t)$, $t \in [t_a, t_b]$, there exists a $\tau \in [t_a, t_b]$, s.t. $x^*(\tau) \in F(\tau)$, if

$$\begin{aligned} &\frac{d(s^1(t), \hat{s}^1(t))}{v_t} + \frac{\tilde{d}_{C_c}(\hat{s}^1(t), s^0(t))}{v_s^1} \leq \frac{1}{2v_t}(w_s^1 + \frac{v_s^0}{v_s^1} w_s^0), \\ \text{and} \qquad &d(s^1(t), \hat{s}^1(t)) \leq \frac{w_s^1}{2}, \quad \forall t \in [t_a, t_b]. \end{aligned} \tag{6.24}$$

Remark 6.4.2: *Lemma 6.4.4* $\sim$ *6.4.6* indicate that any target that tries to cross C through the gap between MSA_1 and MSA_0 will be detected by MSA_1 at least once if (6.19), (6.23) or (6.24) holds. In other words, $S(t|t)$ will not propagate through the gap between MSA_1 and MSA_0.

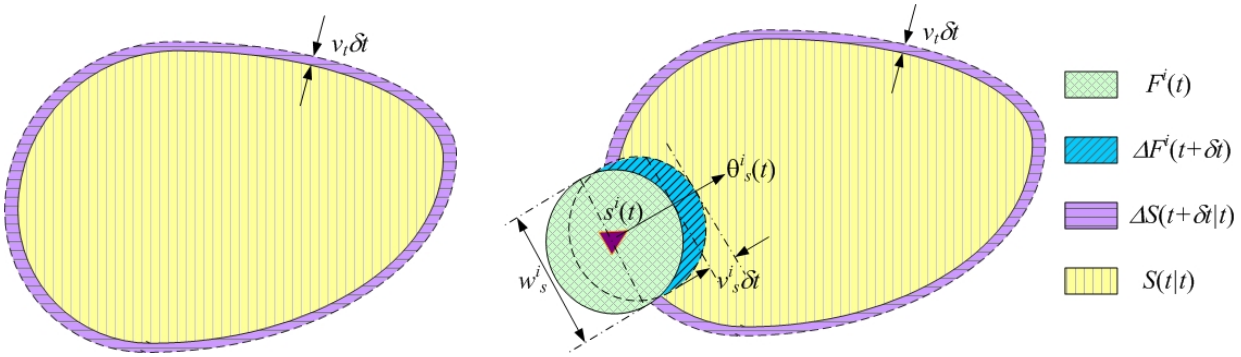
6.5 A Necessary Condition for Non-Escape Search

Theorem 6.5.1: There is always a non-zero probability of escape if the number of MSAs is less than M_{min}, where

$$M_{min} = \frac{2\sqrt{\pi A(S(t_0|t_0))} v_t}{w_s v_s + L_s v_t}. \tag{6.25}$$

Proof Consider the change of $F(t)$ from t to $t + \delta t$, where δt is an arbitrarily small time interval, as illustrated in Fig. 6.5(b).

$$F(t + \delta t) \quad \subset \quad F(t + \delta t) \cup F(t)$$



(a) $S(t+\delta t|t)$ (b) $S(t+\delta t|t+\delta t)$

Figure 6.5: The evolution of $S(t|t)$ and $A(S(t|t))$ for the worst case.

$$= F(t) \cup \Delta F \tag{6.26}$$

where $\Delta F = \bigcup\{x; x \in F(t+\delta t) \text{ and } x \notin F(t)\}$.

In addition, (6.17) implies that

$$A(F(t) \cap S(t|t)) = 0, \forall t \geq t_0. \tag{6.27}$$

Thus, we have

$$\begin{aligned}
& A\left(F(t+\delta t) \cap S(t+\delta t|t)\right) \\
\leq\ & A\left(F(t) \cap S(t+\delta t|t)\right) + A\left(\Delta F \cap S(t+\delta t|t)\right) \\
=\ & A\left(F(t) \cap S(t|t)\right) + A\left(F(t) \cap \Delta S\right) + A\left(\Delta F \cap S(t+\delta t|t)\right) \\
=\ & A\left(F(t) \cap \Delta S\right) + A\left(\Delta F \cap S(t+\delta t|t)\right) \\
\leq\ & ML_s v_t \delta t + M w_s v_s \delta t + o(\delta t)
\end{aligned} \tag{6.28}$$

where ΔS is the region enclosed by $C(S(t+\delta t|t+\delta t))$ and $C(S(t+\delta t|t))$.

Consequently, for the non-detection case in (6.15), we have

$$A(S(t+\delta t|t+\delta t))$$

$$
\begin{aligned}
&= A(S(t+\delta t|t) - A(F(t+\delta t) \cap S(t+\delta t|t)) \\
&\geq A(S(t|t)) + L(C(S(t|t)))v_t\delta t - Mw_s v_s \delta t - ML_s v_t \delta t + o(\delta t). \qquad (6.29)
\end{aligned}
$$

Recall that the perimeter of a 2-D region S and its area satisfies [96]:

$$
L(C(S)) \geq 2\sqrt{\pi A(S)}. \qquad (6.30)
$$

It is worth noting that (6.30) holds even if S consists of multiple disconnected regions. The combination of (6.25) (6.30) and (6.29) will lead to

$$
\begin{aligned}
& A(S(t_0+\delta t|t_0+\delta t)) \\
\geq\ & A(S(t_0|t_0)) + 2\sqrt{\pi \cdot A(S(t_0|t_0))}v_t\delta t - ML_s v_t\delta t - Mw_s v_s\delta t + o(\delta t) \\
>\ & A(S(t_0|t_0)) + o(\delta t), \qquad (6.31)
\end{aligned}
$$

and

$$
A(S(t+\delta t|t+\delta t)) > A(S(t|t)) + o(\delta t), \qquad (6.32)
$$

for any $t > t_0$, if $A(S(t)) \geq A(S(t_0))$.

As $\delta t \to 0$, we then obtain that

$$
A(S(t_1|t_1)) > A(S(t_2|t_2)), \quad \forall t_1 > t_2 \geq t_0. \qquad (6.33)
$$

In other words, $A(S(t|t))$ is unstablizable for the worst case.

The result in *Theorem 6.5.1* can be easily extended to the case of heterogenous MSAs as follows.

Theorem 6.5.2: There is always a non-zero probability of escape if

$$
\sum_{i=0}^{M-1} w_s^i v_s^i + v_t \sum_{i=0}^{M-1} L_s^i < 2\sqrt{\pi A(S(t_0|t_0))}v_t, \qquad (6.34)
$$

6.6 The Progressively-Spiral-In (PSI) Algorithm and a Sufficient Condition for Non-Escape Search

In this section, a multi-MSA search formation, called the *Progressively-Spiral-In algorithm* (PSI), is introduced first. Then a sufficient condition in terms of the number of MSAs to guarantee a non-escape search based on the PSI algorithm is presented. Without loss of generality, the FOV of each MSA is assumed to be a disk centered at $s^i(t)$ with a radius of $w_s/2$. It is worth noting that for MSAs with footprints of other shapes, one can treat the maximum disk enclosed by the footprint as the pseudo footprint. Thus, the results derived in this section can still be employed.

6.6.1 Algorithm Overview

Assume that the *survival zone* is a finite region initially (i.e. at $t = t_0$, $S(t_0|t_0)$ is a finite subset of $\mathcal{R}^2$). Let C_0 be the boundary of the convex hull of $S(t_0|t_0)$. In initialization, we arbitrarily pick one MSA (denoted as MSA_0) as the leader, which is placed at an arbitrary position along $\hat{C}_0$, where $\hat{C}_0 = \mathcal{IE}(C_0, w_s/2)$. The rest of the MSAs (i.e the followers) also start from $\hat{C}_0$. Without loss of generality, we let the followers be distributed counterclockwise along $\hat{C}_0$. The distances between two consecutive MSAs satisfy

$$\tilde{d}_{\hat{C}_0}(s^i(t_0), s^{i-1}(t_0)) = \frac{w_s}{v_t} v_s, \quad i = 1..M-1. \tag{6.35}$$

All the initial headings of the MSAs are along the clockwise tangent direction of $\hat{C}_0$ (i.e. $\theta^i(t_0) = \phi_l(\hat{C}_0, s^i(t_0))$,$i = 0..M-1$).

When the search starts, MSA_0 moves into the inside of C_0, whose right-hand end (i.e. $s_r^0(t)$) shifts from C_0 to its *isotropic contractions* by following a series of

clockwise *spiral-in cycles*, $C_r^{sp}(k)$. During each *spiral-in cycle*, MSA_0 will complete a 360 degree turn, which will be explained in detail the next subsection.

As for the rest of the MSA team, MSA_i $(i > 0)$ will chase MSA_{i-1} $\hat{C}_0$ from $s^i(t_0)$ to $s^{i-1}(t_0)$ first. Then it will follow the same trajectory that MSA_{i-1} passes. The search ends when the target is detected.

In the rest of this section, we will use $C^{sp}(k)$ to stand for the corresponding trajectory of $s^0(t)$ in the k^{th} *spiral-in cycle* $(k \geq 1)$. In the meantime, the time instant that MSA_0 finish its $C^{sp}(k)$ is denoted as t_k. In other words, $C_r^{sp}(k)$ (or $C^{sp}(k)$) starts at $s_r^0(t_{k-1})$ (or $s^0(t_{k-1})$) and ends at $s_r^0(t_k)$ (or $s^0(t_k)$).

6.6.2 The Construction of $C_r^{sp}(k)$ and $C^{sp}(k)$, $k \geq 1$

Without loss of generality, let us assume that $\theta_s^0(t_{k-1}) = 2\pi$. Each *spiral-in cycle* $C_r^{sp}(k)$ starts from $s^0(t_k^{sp})$ and extends in the following way.

For any $x \in C_r^{cp}(k)$, if x is not a *corner* of C',

$$\kappa(C_r^{cp}(k), x) = \begin{cases} \mu_c \kappa(C', x) + \kappa_0, & \text{if } \phi_l(C_r^{cp}(k), x) > (1 - \frac{\alpha_k}{2\pi})\phi_l(C', x); \\ 0, & \text{if } \phi_l(C_r^{cp}(k), x) \leq (1 - \frac{\alpha_k}{2\pi})\phi_l(C', x). \end{cases} \tag{6.36}$$

where

$$C_k = \mathcal{IC}(C_{k-1}, \tilde{d}(s_r^0(t_{k-1}), C_{k-1})) = \mathcal{IC}(C_0, \tilde{d}(s_r^0(t_{k-1}), C_0)), \tag{6.37}$$

$$C' = \mathcal{IC}(C_{k-1}, \tilde{d}(x, C_{k-1})) \tag{6.38}$$

$$\alpha_k = \arctan \frac{d_k}{L(C_{k-1})}. \tag{6.39}$$

Here d_k is a user chosen parameter, $\mu_c \geq 1$ and $\kappa_0 > 0$ are predefined constants.

In the case that x is a *corner* of C', $C_r^{cp}(k)$ will also make a sharp turn, whose right-hand tangent direction is chosen as:

$$\phi_r(C_r^{cp}(k), x) = (1 - \frac{\alpha_k}{2\pi})\phi_r(C', x). \tag{6.40}$$

where C', α_k are as defined in (6.38) and (6.39).

Based on (6.36) and (6.40), we can also get the corresponding controller on MSA_0 to generate such a spiral-in cycle $C^{cp}(k)$ as follows (for $t_{k-1} \leq t \leq t_k$):

$$u^0(t) = \begin{cases} \frac{\mu_c\kappa(C', s_r^0(t))+\kappa_0}{1+[\mu_c\kappa(C', s_r^0(t))+\kappa_0]w_s/2}v_s, & \text{if } s_r^0(t) \text{ is not a } \textit{corner} \text{ of } C' \\ & \text{and } \phi_l(C_r^{cp}, s_r^0(t)) > (1 - \frac{\alpha_k}{2\pi})\phi_l(C', s_r^0(t)); \\ 2v_s/w_s, & \text{if } s_r^0(t) \text{ is a } \textit{corner} \text{ of } C' \\ & \text{and } \theta_s^0(t) \geq (1 - \frac{\alpha_k}{2\pi})\phi_r(C', s_r^0(t)); \\ 0, & \text{otherwise;} \end{cases} \tag{6.41}$$

where C', α_k are as defined in (6.38) and (6.39).

It is worth noting that (6.36) and (6.40) implies that both $C_r^{sp}(k)$ and $C^{sp}(k)$ only make right turns. Thus, both of them are always convex.

6.6.3 Properties of Spiral-in Cycles

Lemma 6.6.1: For any $x \in C_r^{sp}(k)$,

$$d(x, C_{k-1}) \leq d_k, \tag{6.42}$$

Proof Consider a very small segment ΔC_r^{sp} of $C_r^{sp}(k)$ from q_0 to q_1 with its length $L(\Delta C_r^{sp}) \ll 1$ (as shown in Fig. 6.6). Let $q_0' = \rho_r^{\mathcal{E}}(q_0, C_{k-1})$ and $q_0' = \rho_l^{\mathcal{E}}(q_1, C_{k-1})$. The segment of C_{k-1} between q_0' and q_1' is then denoted as ΔC_{k-1}.

Since ΔC_r^{sp} is arbitrarily small, we can guarantee that there is no *corners* of ΔC_r^{sp} in between q_0(or q_0') and q_1(or q_1'). Thus, both ΔC_r^{sp} and ΔC_{k-1} can be treated as

straight line segments. According to the PSI algorithm, we have

$$\begin{aligned} \Delta d &\leq L(\Delta C') \tan \alpha_k \\ &= \frac{L(\Delta C'') d_k}{L(C_{k-1})} \\ &\leq \frac{L(\Delta C_{k-1}) d_k}{L(C_{k-1})} \qquad \text{(due to the convexity of } C_{k-1} \text{ and } C'\text{)}, \end{aligned} \tag{6.43}$$

where $\Delta C'$ is the corresponding segment of ΔC_r^{sp} on $\mathcal{IC}(C_{k-1}, d(q_0, C_{k-1}))$.

Let $L(\Delta C_r^{sp})$ goes to zero and take integrations on both sides of (6.43) and we get (6.42).

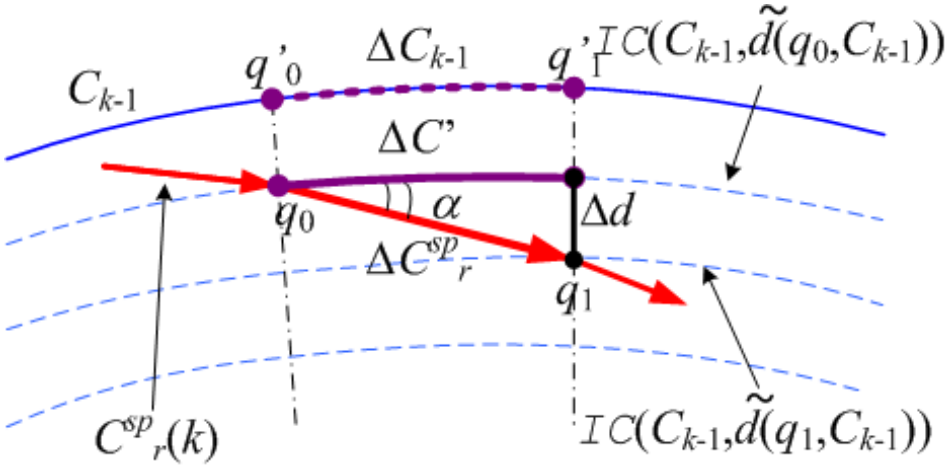


Figure 6.6: Illustration of *Lemma 6.6.1*.

Lemma 6.6.2: Consider a segment of a *spiral-in cycle* $C^{sp}(k)$ from $s^0(t_a)$ to $s^0(t_b)$, $t_{k-1} \leq t_a < t_b \leq t_k$, $k \geq 1$. We have

$$\tilde{d}_{C^{sp}(k)}(s^0(t_a), s^0(t_b)) \leq \tilde{d}_{\hat{C}_{k-1}}(\hat{s}^0(t_a), \hat{s}^0(t_b)), \tag{6.44}$$

where $\hat{C}_k = \mathcal{IE}(C_k, w_s/2)$; $\hat{s}^0(t_a)$ and $\hat{s}^0(t_b)$ are the intersections of line $\overline{s_r^0(t) s_l^0(t)}$ and $\hat{C}_{k-1}$ at t_a and t_b, respectively (as shown in Fig. 6.7).

Proof Consider an arbitrarily small segment δC of the $C^{sp}(k)$ from $s^0(t) = q_0$ to $s^0(t+\delta t) = q_1$ as shown in Fig. 6.7. Let q_0' and q_1' as the intersections of line $\overline{s_r^0(t)s_l^0(t)}$ and $\hat{C}_{k-1}$ at t and $t + \delta t$, respectively.

Because both $\hat{C}_{k-1}$ and $C^{sp}(k)$ are *isotropic expansions* of convex curves, they are both smooth. Thus, we can approximate δC and the segment of C_{k-1} between q_0' and q_1' by two line segments, as illustrated in Fig. 6.7. Due to the convexity of $\hat{C}_{k-1}$ and $C^{sp}(k)$, we know that $\angle q_0' q_0 q_1 \geq \pi/2$ and $\angle q_1' q_1 q_0 \geq \pi/2$. Without loss of generality, let $d(q_0, q_0') \geq d(q_1, q_1')$. Thus, it is true that

$$\begin{aligned} L(\delta C) &= d(q_0, q_1) + o(\delta t) \\ &\leq d(q_0, q_1'') + o(\delta t) \\ &\leq d(q_0', q_1') + o(\delta t) \\ &\leq \tilde{d}_{C_{k-1}}(q_0', q_1') + o(\delta t), \end{aligned} \tag{6.45}$$

where q_1'' is the intersection of line $\overline{q_1 q_1'}$ and the auxiliary line that is parallel to line $\overline{q_0' q_1'}$.

Letting δt goes to zero and integrating both sides of (6.45), we will get (6.44).

In a similar way, we can prove that

Lemma 6.6.3: $\forall t_{k-1} \leq t_a < t_b \leq t_k$, $k > 1$,

$$\tilde{d}_{C^{sp}(k)}(s^0(t_a), s^0(t_b)) < \tilde{d}_{C^{sp}(k-1)}(\hat{s}^0(t_a), \hat{s}^0(t_b)), \tag{6.46}$$

where $\hat{s}^0(t_a)$ and $\hat{s}^0(t_b)$ are the intersections of line $\overline{s_r^0(t)s_l^0(t)}$ and $C^{sp}(k-1)$ at t_a and t_b, respectively.

In particular, if we choose $t_a = t_{k-1}$ and $t_b = t_k$ in (6.44) and (6.47), we will get the following conclusion:

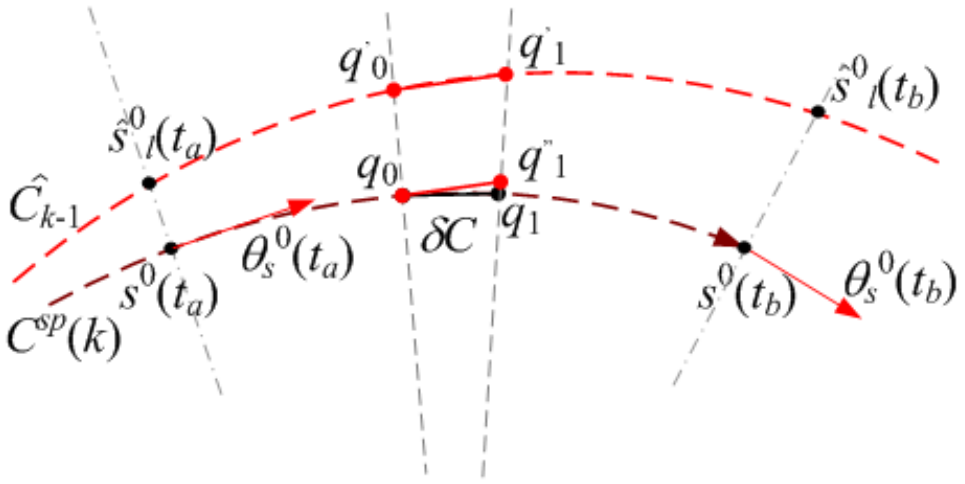


Figure 6.7: Illustration of *Lemma 6.6.2*.

Lemma 6.6.4: $\forall k \geq 1$,

$$L(C^{sp}(k+1)) < L(C^{sp}(k)) < L(C_{k-1}). \tag{6.47}$$

Remark 6.6.1: *Lemma 6.6.2* $\sim$ *6.6.4* indicates that the "gap" between MSA_0 and MSA_{M-1} (i.e. the distance between $\hat{s}^0(t)$ and $s^{M-1}(t)$ along the *spiral-in cycles*) is monotonically decreasing.

Lemma 6.6.5: During the first *spiral-in cycle* of the PSI algorithm,

$$d(s^0(t), \hat{s}^0(t) \leq \frac{w_s}{2}\left[\sqrt{1+\frac{\bar{d}_1^2}{L^2(C_0)}}-1\right] + \bar{d}_1\sqrt{1+\frac{\bar{d}_1^2}{L^2(C_0)}}, \tag{6.48}$$

where $\bar{d} = \overline{d}_{\mathcal{IC}(C_0,d_c)(C_0)}$ and $\hat{s}^0(t)$ is the intersection of line $\overline{s_r^0(t)s^0(t)}$ and $\hat{C}_0$

Proof As the PSI algorithm indicates, the right-hand end of the footprint of MSA_0, $s_r^0(t)$ is shifting from C_0 to its *isotropic contractions*. Depending on whether $s_r^0(t)$ is a *corner* and MSA_0's heading, the status of $s_r^0(t)$ can be categorized into the following three situations:

Case 1: $s_r^0(t)$ is not a *corner* of $\mathcal{IC}(C_0, \tilde{d}(s_r^0(t), C_0))$.

Denote q_r^0 as the *contraction seed* of $s_r^0(t)$ on C_0 and q^0 as the *expanding point* of q_r^0 on $\hat{C}_0$, as shown in Fig. 6.8. Since $s_r^0(t)$ is not a *corner*, $s_r^0(t)$, q_r^0 and q^0 are on the same straight line, which is perpendicular to both C_0 and $\hat{C}_0$ (due to (**P5**), (**P6**) and (**P8**) in section 6.3). According to the PSI algorithm, we know that $\angle s^0(t)s_r^0(t)q^0 \leq \alpha_1 = \arctan\frac{d_1}{L(C_0)}$. Meanwhile, due to the convexity of $\hat{C}_0$, one can easily show that

$$d(s_r^0(t), \hat{s}^0(t)) \leq d(s_r^0(t), q^1) = \frac{1}{\cos\alpha_1} d(s_r^0(t), q^0), \tag{6.49}$$

where q^1 is the intersection of line $\overline{s_r^0(t)s^0(t)}$ and the tangent of $\hat{C}_0$ at q^0, as illustrated in Fig. 6.8.

Eq. (6.49) further leads to

$$\begin{aligned} d(s^0(t), \hat{s}^0(t)) &= d(s_r^0(t), \hat{s}^0(t)) - \frac{w_s}{2} \\ &\leq \sqrt{1 + \frac{d_1^2}{L^2(C_0)}} d(s_r^0(t), q^0) - \frac{w_s}{2} \\ &\leq \sqrt{1 + \frac{d_1^2}{L^2(C_0)}} \left(\frac{w_s}{2} + d_1\right) - \frac{w_s}{2}. \end{aligned} \tag{6.50}$$

Clearly, the right side of (6.49) is monotonically increasing function of d_1. Thus, we can replace d_1 by $\overline{d}_{\mathcal{IC}(C_0,d_1)(C_0)}$ and the inequality still holds, which gives us (6.48).

Case 2: $s_r^0(t)$ is a *corner* of $\mathcal{IC}(C_0, \tilde{d}(s_r^0(t), C_0))$ and $\theta_s^0(t) \geq \phi_r(C'', s_r^0(t))$, where $C'' = \mathcal{IC}(C_0, \tilde{d}(s_r^0(t), C_0))$.

Denote q_r^0 as the *expansion seed* of $\hat{s}^0(t)$ on C_0. According to (**P7** and (**P9**) in section 6.3, any q_r^0 in this case will be a *contraction seed* of $s_r^0(t)$, as illustrated in Fig.

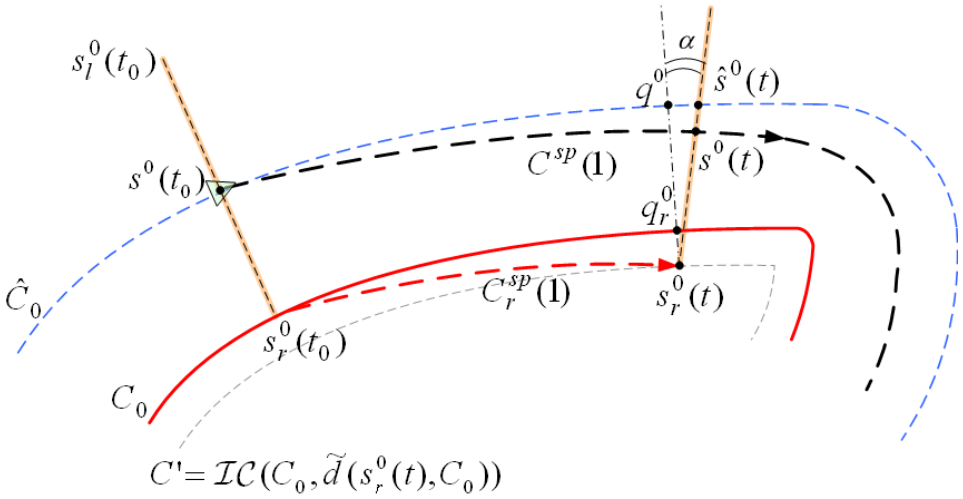


Figure 6.8: Case 1 of *Lemma 6.6.5*

6.9. Therefore, we have

$$\begin{aligned} d(s^0(t), \hat{s}^0(t)) &= d(s_r^0(t), \hat{s}^0(t)) - \frac{w_s}{2} \\ &\leq d(s_r^0(t), q^0) + d(q^0, \hat{s}_r^0(t)) - \frac{w_s}{2} \\ &= d(s_r^0(t), q^0) \\ &\leq \overline{d}_{\mathcal{IC}(C_0, d_1)(C_0)}. \end{aligned} \tag{6.51}$$

Comparing the right side of (6.51) with that of (6.48), we know that the inequality in (6.48) holds in this case.

Case 3: $s_r^0(t)$ is a *corner* of $\mathcal{IC}(C_0, \tilde{d}(s_r^0(t), C_0))$ and $\theta_s^0(t) < \phi_r(C'', s_r^0(t))$, where $C'' = \mathcal{IC}(C_0, \tilde{d}(s_r^0(t), C_0))$.

Denote q_r^0 as the last *contracting seed* of $s_r^0(t)$ on C_0 (in a clockwise way). According to (**P10**) in section 6.3, we know that line $\overline{s_r^0(t)q_r^0}$ is perpendicular to both C_0 and $\hat{C}_0$. Let q^0 be the intersection of line $\overline{s_r^0(t)q_r^0}$ and $\hat{C}_0$. Similar to Case 1, we

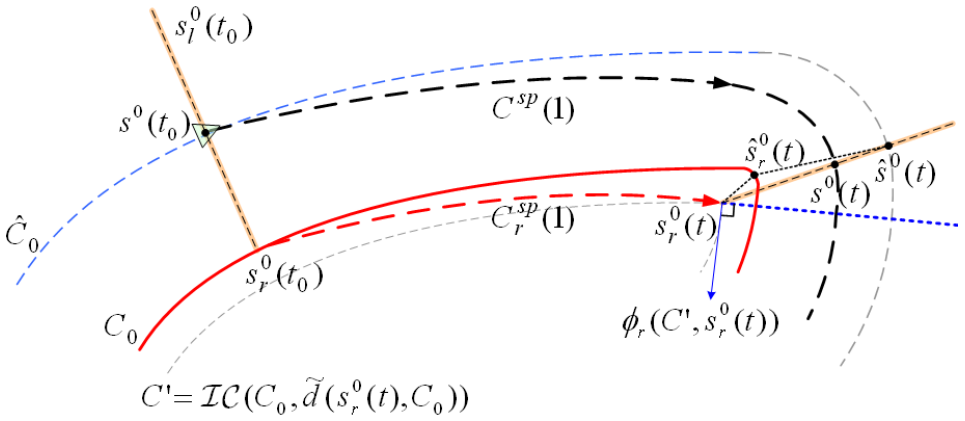


Figure 6.9: Case 2 of *Lemma 6.6.5*

have $\angle s^0(t)s_r^0(t)q^0 \leq \alpha_1 = \arctan\frac{d_1}{L(C_0)}$, and

$$\begin{aligned} d(s^0(t), \hat{s}^0(t)) &= d(s_r^0(t), \hat{s}^0(t)) - \frac{w_s}{2} \\ &\leq (1 - \frac{1}{\cos\alpha_1} d(s_r^0(t), q^0) - \frac{w_s}{2}, \end{aligned} \tag{6.52}$$

which will lead to (6.48) in the same way as it does in case 1.

Lemma 6.6.6: In the rest *spiral-in cycles* in the PSI algorithm (i.e. $k \geq 1$),

$$\begin{aligned} d(s^0(t), \hat{s}^0(t) &\leq \frac{w_s}{2}\left(\sqrt{1 + \frac{\bar{d}_{k+1}^2}{L^2(C_k)}} - 1\right) \\ &+ (\bar{d}_{k+1} + \bar{d}_k)\sqrt{1 + \frac{\bar{d}_{k+1}^2}{L^2(C_k)}}, \quad \forall t_k \leq t \leq t_{k+1}, \end{aligned} \tag{6.53}$$

where C_k is as defined in (6.37), $\bar{d}_k = \overline{d}_{\mathcal{IC}(C_{k-1}, d_k)}(C_{k-1})$ and $\hat{s}^0(t)$ is the intersection of line $\overline{s_r^0(t)s^0(t)}$ and $C^{sp}(k)$.

Proof Depending on whether $s_r^0(t)$ is a *corner* and its heading with respect to the previous cycle, the status of MSA_0 during the rest *spiral-in cycles* can also be categorized into three cases as we did in the proof of *Lemma 6.6.5*. Here we only give

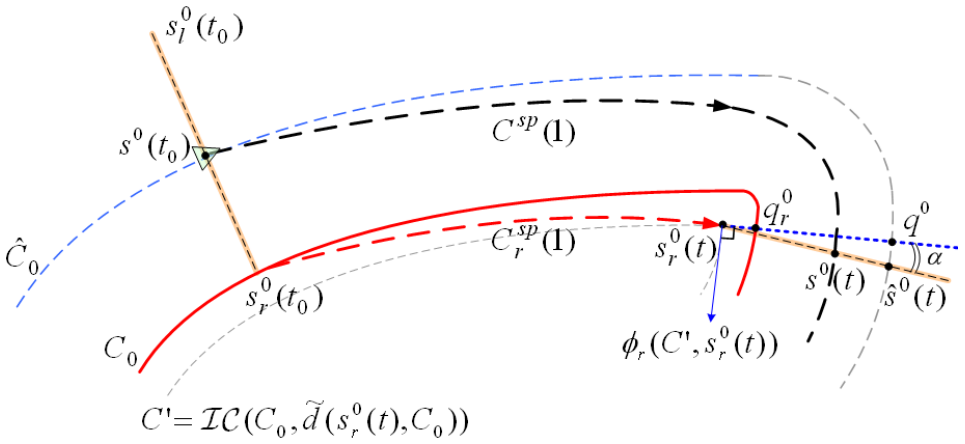


Figure 6.10: Case 3 of *Lemma 6.6.5*

the proof of (6.53) for Case 1 (i.e. $s_r^0(t)$ is not a *corner*). The proofs of the other two cases are omitted since they can be achieved in a similar way.

Let q_0 be the *contraction seed* of $s_r^0(t)$ on C_k, as illustrated in Fig. 6.11. Also, let q_1/q_2 be the intersections of line $\overline{s_r^0(t)q_0}$ and $C_r^{sp}(k)/C^{sp}(k)$.

As illustrated in Fig. 6.11, it is clear that

$$d(s_r^0(t), q_2) \leq \frac{w_s}{2} + d_k + d_{k+1}. \tag{6.54}$$

In the mean time, according to the PSI algorithm, we also know that

$$\alpha \leq \alpha_{k+1}, \tag{6.55}$$

where α is as shown in Fig. 6.11).

Due to the convexity of $C^{sp}(k)$, it can be easily shown that

$$d(\hat{s}^0(t), s_r^0(t)) \leq \frac{d(q_2, s_r^0(t))}{\cos\alpha}. \tag{6.56}$$

Combining (6.54) (6.55) and (6.56), we get

$$d(\hat{s}^0(t), s_r^0(t)) \leq \frac{w_s/2 + d_k + d_{k+1}}{\cos \alpha_{k+1}}, \tag{6.57}$$

which is equivalent to (6.53).

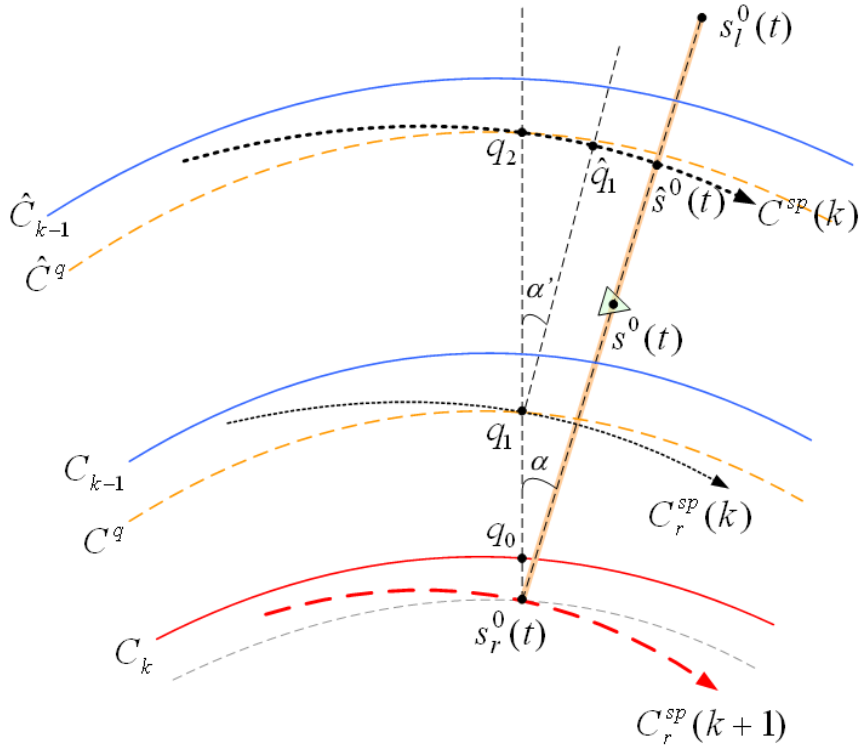


Figure 6.11: Illustration of *Lemma 6.6.6*.

Lemma 6.6.7: If the initial condition of the PSI algorithm satisfies that

$$\tilde{d}_{C_0}(s^0(t_0), s^{M-1}(t_0)) < \frac{w_s}{v_s} v_t, \tag{6.58}$$

there exists $d_1 > 0$, s.t. any target that tries to move across the gap between MSA_0 and MSA^* during the first *spiral-in cycle* will be detected by at least one MSA, where

MSA^* is the MSA that is ahead of MSA_0. Note that MSA^* is not necessarily to be MSA_{M-1}, since MSA_0 may catch MSA_{M-1} and be ahead of MSA_{M-1} during the first *spiral-in cycle* (in this case, MSA_{M-2} becomes MSA^*). **Proof** Let us first consider the following function:

$$g(\tau) = \frac{w_s}{2}\left[\sqrt{1+\frac{\tau^2}{L^2(C_0)}}-1\right] + \tau\sqrt{1+\frac{\tau^2}{L^2(C_0)}} + \frac{\tilde{d}_{C_0}(s^0(t_0), s^{M-1}(t_0))}{v_s}v_t - w_s \quad (6.59)$$

Obviously, $g(0) < 0 < g(w_s)$. Since $g(\tau)$ is a continuous function of τ, there exists a $d^* \in (0, w_s)$ s.t. $g(d^*) = 0$.

Consider C_0 and its *isotropic contraction* $\mathcal{IC}(C_0, d)$. Clearly, $\bar{d}(C_0, \mathcal{IC}(C_0, v_0 t))$ is a continuous function of d. Therefore, there exists $d_c^* > 0$, s.t. $\bar{d}(C_0, \mathcal{IC}(C_0, d_c^*)) = d^*$.

In the meantime, *Lemma 6.6.3* indicates that

$$\tilde{d}_{C_0}(s^0(t_0), \hat{s}^0(t)) \geq \tilde{d}_{C^{sp}(1)}(s^0(t_0), s^0(t)) \quad (6.60)$$

Since MSA_0 and MSA_{M-1} have the same cruise speed, we also have

$$\tilde{d}_{C^{sp}(1)}(s^0(t_0), s^0(t)) = \tilde{d}_{C_0}(s^{M-1}(t_0), s^{M-1}(t)) \quad (6.61)$$

Combining (6.60) and (6.61), we can obtain

$$\tilde{d}_{C_0}(\hat{s}^0(t), s^{M-1}(t)) \leq \tilde{d}_{C_0}(s^0(t_0), s^{M-1}(t_0)). \quad (6.62)$$

On the other hand, (6.48) indicates that $d(s^0(t), \hat{s}^0(t))$ is a monotonically-increasing function of d_1. Clearly, there exists a d_w^*, s.t. $d(s^0(t), \hat{s}^0(t)) \leq \frac{w_s}{2}$, $\forall d_1 \leq d_w^*$ and $\forall t_0 \leq t \leq t_1$.

Now, if we choose the constant $d_1 = \min\{d_c^*, d_w^*\}$ in the PSI algorithm, we have

$$\frac{d(s_l^0(t), \hat{s}_l^0(t))}{v_t} + \frac{\tilde{d}_{C_0}(\hat{s}^0(t), s^{M-1}(t))}{v_s}$$

$$
\begin{aligned}
&\leq \frac{w_s}{2v_t}\left[\sqrt{1+\frac{\bar{d}_1^2}{L^2(C_0)}}-1\right]+\bar{d}_1\sqrt{1+\frac{\bar{d}_1^2}{L^2(C_0)}}+\frac{\tilde{d}_{C_0}(s^0(t_0),s^{M-1}(t_0))}{v_s}\\
&= \frac{w_s}{v_t}, \qquad (6.63)
\end{aligned}
$$

where $\bar{d}_1 = d^* = \bar{d}(C_0, \mathcal{IC}(C_0, d_1))$.

Based on *Lemma 6.4.4*, we get the conclusion that any target that tries to move across the gap between MSA_0 and MSA^* during the first *spiral-in cycle* will be detected either by MSA_0 (before MSA_0 catch MSA_{M-1}), or by MSA_{M-1} (after MSA_0 goes ahead of MSA_{M-1}).

In a similar way, one can also prove that

Lemma 6.6.8: In the PSI algorithm, if there exists $d_{k-1} \geq 0$, s.t. any target that tries to move across the gap between MSA_0 and MSA^* during the $(k-1)^{th}$ *spiral-in cycle* will be detected by at least one MSA, then there exists $d_k \geq 0$, s.t. any target that tries to move across the gap between MSA_0 and MSA^* during the k^{th} *spiral-in cycle* will be detected by at least one MSA. Here MSA^* stands for the MSA that is ahead of MSA_0 and $k > 1$.

Proof After MSA_0 finishes the $(k-1)^{th}$ *spiral-in cycle*. The MSA that is ahead of him can be either MSA_{M-1} or others.

Case 1: MSA_{M-1} is ahead of MSA_0 at t_{k-1}:

Consider the following function:

$$
\begin{aligned}
g(\tau) &= \frac{w_s}{2}\left[\sqrt{1+\frac{\tau^2}{L^2(C_k)}}-1\right]\\
&+ (\tau+d_{k-1})\sqrt{1+\frac{\tau^2}{L^2(C_k)}}\\
&+ \frac{\tilde{d}_{C^{sp}(k-1)}(s^0(t_{k-1}),s^{M-1}(t_{k-1}))}{v_s}v_t - w_s. \qquad (6.64)
\end{aligned}
$$

Similar to *Lemma 6.6.7*, we know that there exists a $d_c^* > 0$, s.t. $g(d^*) = 0$, where $d^* = \bar{d}(C_{k-1}, \mathcal{IC}(C_{k-1}, d_c^*))$.

Therefore, if we choose the $d_k = \min\{d_c^*, d_w^*\}$ in the k^{th} *spiral-in cycle* (here d_w^* is obtained in a similar way as we did in *Lemma 6.6.7*) we will have

$$\begin{aligned} & \frac{d(s_l^0(t), \hat{s}_l^0(t))}{v_t} + \frac{\tilde{d}_{C^{sp}(k-1)}(\hat{s}^0(t), s^{M-1}(t))}{v_s} \\ \leq & \frac{w_s}{2v_t}\left[\sqrt{1 + \frac{\bar{d}_k^2}{L^2(C_{k-1})}} - 1\right] \\ & + (\bar{d}_k + \bar{d}_{k-1})\sqrt{1 + \frac{\bar{d}_k^2}{L^2(C_{k-1})}} \\ & + \frac{\tilde{d}_{C^{sp}(k-1)}(s^0(t_{k-1}), s^{M-1}(t_{k-1}))}{v_s} \\ = & \frac{w_s}{v_t}, \end{aligned} \tag{6.65}$$

where $\bar{d}_k = \bar{d}(C_{k-1}, \mathcal{IC}(C_{k-1}, d_{k-1}))$.

Based on *Lemma 6.4.4*, we get the conclusion that any target that tries to move across the gap between MSA_0 and MSA^* during the k^{th} *spiral-in cycle* will be detected either by MSA_0 (before MSA_0 catch MSA_{M-1}), or by MSA_{M-1} (after MSA_0 goes ahead of MSA_{M-1}).

Case 2: MSA_{M-l} is ahead of MSA_0 at t_{k-1}, $l > 1$:

Consider the gap between MSA_0 and MSA_{M-l} at t_{k-1}. If $\tilde{d}_{C^{sp}(k-1)(\hat{s}^0(t_{k-1}), s^{M-l}(t_{k-1})} \leq \frac{w_s}{2}$, we can just use a smaller team of $M - l + 1$ MSAs (i.e. $\{MSA_i\}_{i=0..M-l}$) to fulfill the rest search task. Similar to case 1 above, we have the conclusion that there exist a $d_k > 0$, s.t. any target that tries to move across the gap between MSA_0 and MSA^* during the k^{th} *spiral-in cycle* will be detected. In the case that $\tilde{d}_{C^{sp}(k-1)(\hat{s}^0(t_{k-1}), s^{M-l}(t_{k-1})} > \frac{w_s}{2}$, one can simply choose $d_k = 0$. Note that the gap between MSA_0 and MSA_{M-l} here will be covered by MSA_{M-l+1} instead of MSA_0.

It is also worth noting that after such a "non-contracting" *spiral-in cycle* (i.e. $d_k = 0$), we will have $d(\hat{s}^0(t_k), s^0(t_k)) = 0$, which leads to a similar situation as in the first *spiral-in cycle*.

6.6.4 A Sufficient Condition for Non-Escape Search

Lemma 6.6.7 and *Lemma 6.6.8* directly lead to the following theorem:

Theorem 6.6.9: Given a team of M homogenous MSAs and an initial probability density function of the positional information of a target being distributed inside a closed, convex curve $C_0 \subset \mathcal{R}^2$, the MSAs team can perform a non-escape search in finite time if

$$M > M_{max} = \frac{(L(C_0) + \pi w_s)v_t}{w_s v_s} \tag{6.66}$$

Proof In the PSI algorithm, every MSA except the leader (i.e. MSA_0) is chasing the one in front of him along the same path and with the same speed. Thus $\tilde{d}_{C^{sp}(k)}(s^i(t), s^{i+1}(t)) \equiv \frac{w_s}{v_t} v_s, i = 0..M-2$ all the time, which means there is no chance that the target (or its *survival zone* $S(t|t)$) can leak out from the gap between MSA $i-1$ and MSA $i, i = 1..M-1$. The only place one should look after is the gap between MSA_{M-1} and the leader MSA_0. According to *Lemma 6.6.7* and *Lemma 6.6.8*, there exists a series of *spiral-in cycles* that can always prevent the target from escaping from the gap between MSA_0 and MSA_{M-1}. Thus, $S(t|t)$ will be maintained inside the belt-shape zone enclosed by the trajectory of the right end of the footprint of MSA_0, $s_r^0(t)$. As implied by (6.59) and (6.64), d_k will not converge to zero unless $L(C_k)$ goes to zero, which guarantees that $S(t|t)$ will keep shrinking to its minimum in finite time. As (**P3**) in section 6.3 indicates, C_k will contract to either a point or a

line segment, which means the target will be found by the MSAs in finite time even for the worst case.

The result above as well as the PSI algorithm can also be extended to the case of heterogenous MSAs with different width (but with the same cruise speed).

Theorem 6.6.10: Given a team of M MSAs with the sample cruise speed, and an initial probability density function of the positional information of a target being distributed inside a closed, convex curve $C_0 \subset \mathcal{R}^2$, the MSAs team can perform a non-escape search in finite time if

$$\sum_{i=0}^{M-1} w_s^i > \frac{2(L(C_0) + \pi w^*)v_t}{v_s}, \tag{6.67}$$

where $w^* = \max\{w_s^i; i = 0..M-1\}$.

Proof In this case, we can choose the MSA that has the maximum width w^* as the team leader. In the initialization stage of the PSI algorithm, we spread out the MSA team along $\hat{C}_0 = \mathcal{IE}(C_0, w^*/2)$. The initial distance between each two MSAs is chosen as $\tilde{d}(s^{i+1}(t_0), s^i(t_0)) \equiv \frac{w_s^i + w_s^{i+1}}{2v_t} v_s, i = 0..M-2$.

Similar to the case of homogenous MSAs, one can prove that there exist a set of $\{d_k\}$, s.t. $S(t|t)$ will be maintained inside the belt-shape zone enclosed by the trajectory of the left end of the footprint of MSA_0, $s_l^0(t)$, which converges to a point in finite time.

Remark 6.6.2: It is worth noting that the PSI algorithm is not limited to single-target search only. By executing the PSI algorithm, any target that is inside $S(t_0|t_0)$ initially will be detected by at least one MSA. The PSI algorithm can also be helpful in general information gathering problems. For instance, in applications such that the MSAs only collect measurements (e.g. through an on-board camera) and send them back (possibly to human operators) for off-line analysis, the PSI algorithm can

guarantee the appearance of all the moving targets that are initially inside the search domain.

6.7 Application Examples and Discussions

6.7.1 Task assessment and Search Strategy Selection

The most straightforward application of the lower bound (i.e. M_{min}) and the upper bound (i.e. M_{max}) is task assessment. Consider an L km by L km square$S(t_0|t_0)$ and a team of M homogenous MSAs. For each MSA, we assume that its footprint is a disk-shape footprint of radius $100m$ (i.e. $w_s = 200m$), $v_s = 100m/s$ and $w_t = 20m/s$. The relationship between M_{max}/M_{min} and L is shown in Fig. 6.12. Thus, given a specific search task, one can evaluate whether the MSA resource is sufficient or not. If M is larger than M_{max} (i.e. M falls in to zone S_3 in Fig. 6.12), the PSI algorithm is a good choice since it can guarantee a non-escape search. On the other hand, if the $M \leq M_{min}$ (i.e. M falls in to zone S_1 in Fig. 6.12), an more aggressive and short-term benefit oriented approach, such as the optimal search methods described in [23, 24, 77, 78, 81–88], becomes a reasonable choice.

It is worth noting that the ratio between M_{max} and M_{min} decreases as the ratio between v_s and v_t increases. In this experiment, $\frac{M_{max}}{M_{min}}$ converges to 1.5 as $\frac{v_s}{v_t}$ goes to infinity (Fig. 6.13).

6.7.2 The PSI algorithm for limited number of MSAs

The PSI algorithm can still be useful even the MSA resource is not sufficient to perform a non-escape search directly. For instance, given an M that falls into zone S_1 or S_2 in Fig. 6.12, one can always find a smaller $L = L'$, which corresponds to a smaller region that allows a non-escape search. Denote $S'(t_0|t_0)$ as such a subset

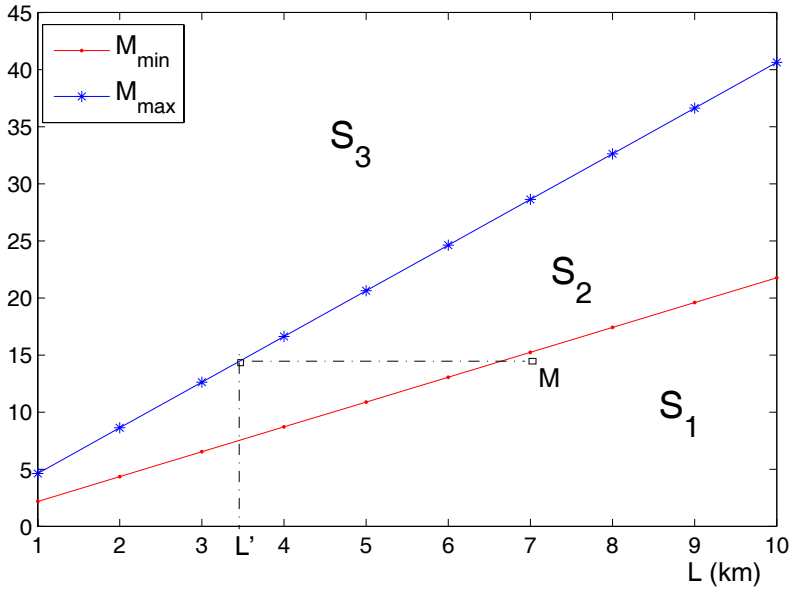


Figure 6.12: The relationship between size of $S(t_0|t_0)$ and M_{max}/M_{min}.

of $S(t_0|t_0)$ and let $P_{sub} = \int_{x \in S'(t_0|t_0)} p_{tgt}(x, t_0)dx$. By executing the PSI algorithm in $S'(t_0|t_0)$, the MSA team will have at least a probability of P_{sub} to find the target.

Consider a $10km \times 10km$ square $S(t_0|t_0)$, in which $p_{tgt}(x, t_0)$ is uniformly distributed. Fig. 6.14 shows the relationship between P_{sub} and M with $w_s = 200m$, $v_s = 100m/s$ and $v_t = 20m/s$. In practice, the initial distribution of the target's state may not be uniform. In this case, P_{sub} corresponds to the maximum probability that $S'(t_0|t_0)$ can cover.

Thus, in the case the number of MSAs is not enough to realize a global non-escape search, one can still use the PSI algorithm by concentrate on the main part of the initial *survival zone*. In the meantime, we can also use P_{sub} as a benchmark metric to evaluate the performance of different search algorithms. Clearly, an aggressive search

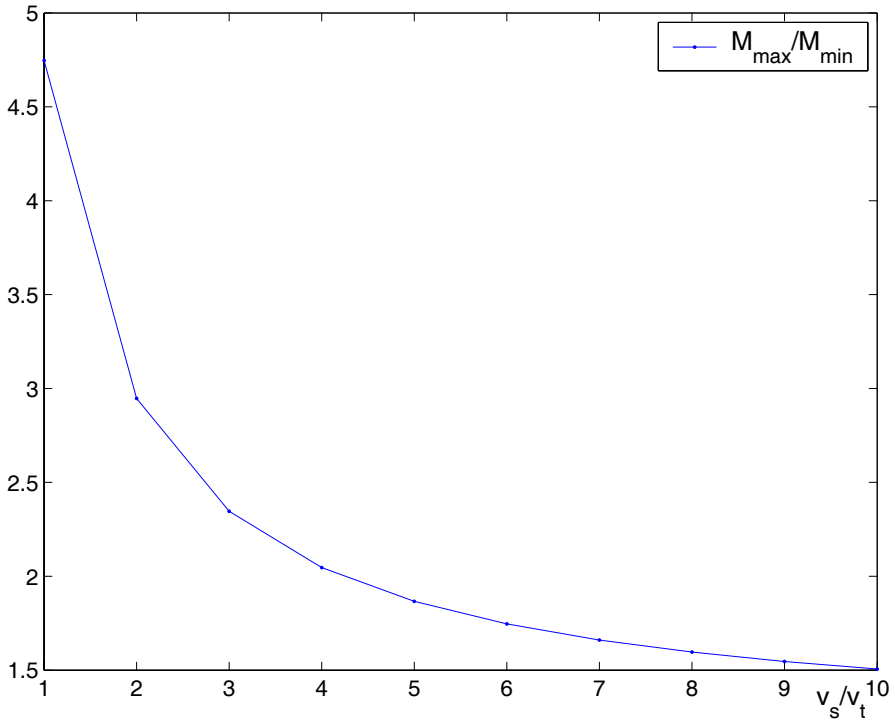


Figure 6.13: The relationship between $\frac{v_s}{v_t}$ and $\frac{M_{max}}{M_{min}}$.

algorithm that can not achieve an average detection probability of P_{sub} is probably not a better choice than the PSI algorithm.

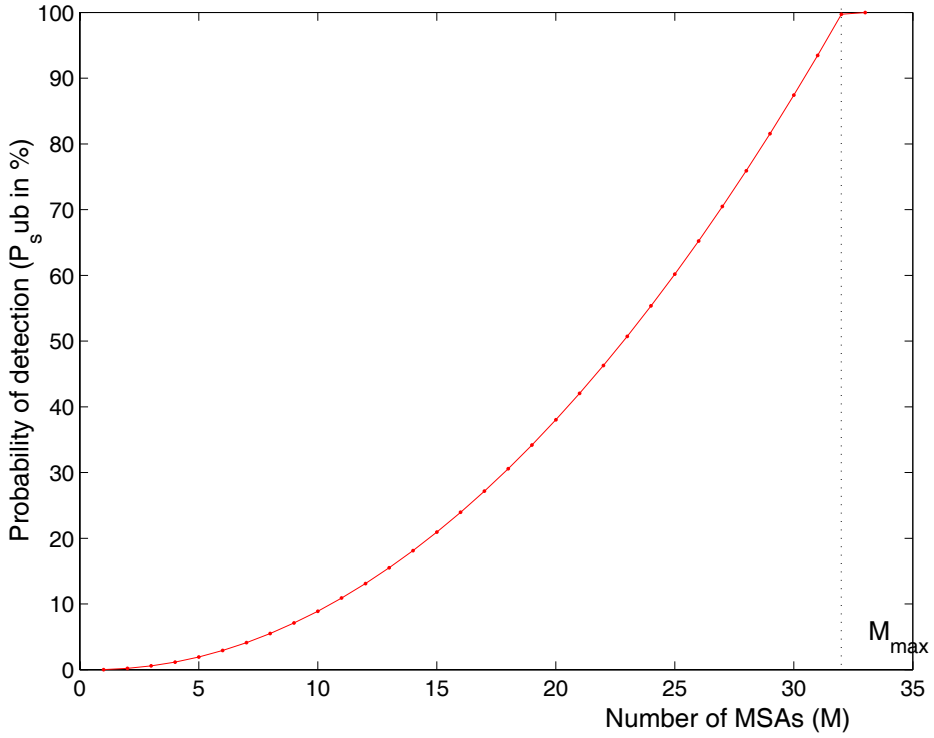


Figure 6.14: Detection probability by applying the PSI algorithm with limited MSAs.

6.7.3 Multi-team search for disconnected $S(t_0|t_0)$

The standard PSI algorithm starts from a convex hull of the initial *survival zone*. In the case that the initial *survival zone* consists of multiple disconnected regions, we may have more than one choice to realize a non-escape search.

For example, assume that $S(t_0|t_0)$ consists of three convex regions, S_a, S_b and S_c, as shown in Fig. 6.15. The simplest way to execute the PSI algorithm is to let an

MSA team start from the boundary of the convex hull of $S(t_0|t_0)$ (denoted as S_{abc} in Fig. 6.16). On the other hand, we can also divide $S(t_0|t_0)$ into two subregions. One is S_c, and the other is $S_a \cup S_b$. Thus, we can separate the MSAs into 2 teams and let each team execute the corresponding PSI algorithm with respect to its own search region (i.e. S_c and S_{ab} as shown in Fig. 6.17).

Obviously, different choices of tasking lead to different numbers of MSAs to fulfill a non-escape job. In the example shown in Fig. 6.15, when S_a and S_b are much closer to each other than they are to S_c, the 2-team search plan (Fig. 6.17) will require less MSAs to guarantee a non-escape search than the single-team version (Fig. 6.16).

In practice, one can compare different tasking configurations and choose the one that requires the minimum number of MSAs to fulfill a non-escape search. Clearly, an $S(t_0|t_0)$ that consists of N disconnected sub-regions will lead to 2^N possible search plans, either single-team or multi-team. Fortunately, such a task assessment procedure takes place only once before the search starts, whose computational load is then affordable in most applications.

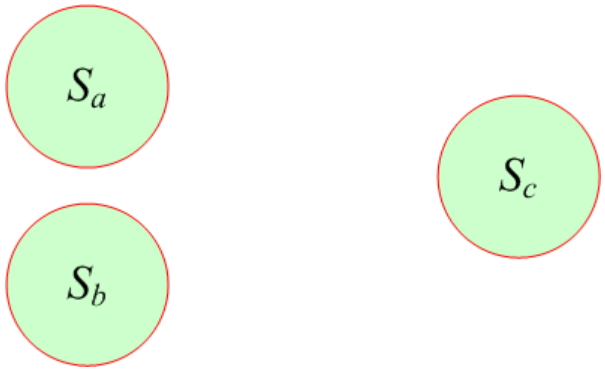


Figure 6.15: An example of $S(t_0|t_0)$ consisting of multiple disconnected regions.

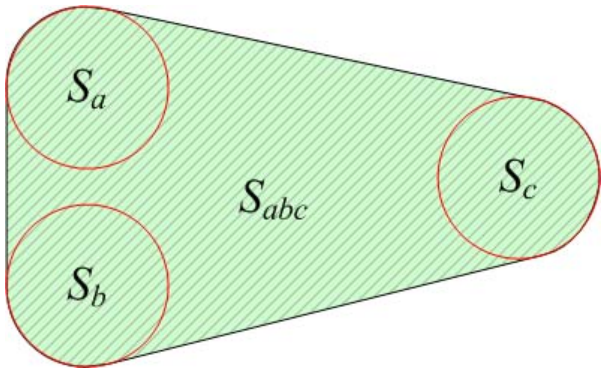


Figure 6.16: A single-team search plan.

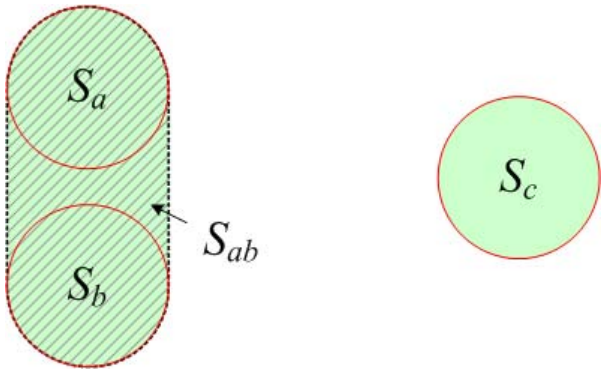


Figure 6.17: A two-team search plan.

6.8 Conclusions

In this chapter, the problem of searching for a moving target by multiple mobile sensor agents (MSA) is studied. The search problem is formulated using an

information-theoretic approach, whose objective is to stabilize the entropy of the target system. An upper bound and a lower bound in terms of the number of MSAs for the existence of a non-escape solution to the search problem are provided. In addition, a multi-MSA search formation to find the target in finite time, the *Progressively-Spiral-In* (PSI) algorithm, is proposed. The main result of this paper is further extended to heterogeneous MSAs.

The PSI algorithm can be considered as a benchmark approach for performance comparison among different target search methods. Future work in this area includes the extension of the PSI algorithm to target search in inhomogeneous terrain surfaces. The idea of *survival zone* control can also be applied to various sensing problems such as border patrolling and region-based surveillance.

CHAPTER 7

MOTION PLANNING FOR MULTI-TARGET SURVEILLANCE WITH MOBILE SENSOR AGENTS

7.1 Introduction

This chapter studies the cooperative motion-planning problem for monitoring N targets by M mobile sensor agents ($M < N$). The work is motivated by the application for Unmanned Air Vehicles (UAV) in both military [24, 78, 79, 86] and civilian surveillance systems [22]. The main part of this chapter has been published in [**?**]. Similar to the previous chapter, the MSA here is simplified as a point mass moving on a 2-D plane at a constant speed, except that a non-holonomic constraint, a bounded turning radius, is added to the model as follows:

$$\begin{cases} \dot{x}_j(t) = V_M cos\vartheta_j(t); \\ \dot{y}_j(t) = V_M sin\vartheta_j(t); \\ \dot{\vartheta}_j(t) = u_j(t), \qquad |u_j(t)| \leq V_M/R_M; \end{cases} \tag{7.1}$$

where $s_j(t) = (x_j(t), y_j(t)) \in \mathcal{R}^2$ and $\vartheta_j(t) \in [0, 2\pi]$ denote the horizontal position and orientation of agent j ($j = 1..M$) at time instant t, respectively. V_M is the speed of the MSA; R_M is the minimum turning radius; and $u_j(t)$ is the control input.

This model is also called the *Dubins car* in the literature [27, 30, 31, 34, 36, 39], and has been widely used as the kinematic model of UAV by researchers [21, 38, 55, 86].

Each MSA is assumed to have an onboard sensor with a restricted local field of view around itself. In this work, the FOV of MSA j at time instant t is defined as a small circle centered at $s_j(t)$, as shown in Fig. 6.1.a) in the previous chapter. However, the general result developed here can be extended to other FOV shapes. It is also assumed that the MSA's move much faster than the targets, which agrees with reality in UAV applications.

The task of the MSAs is to monitor a number of ground targets $Q = \{q_i\}, i = 1..N$, where $q_i = (q_{ix}, q_{iy})$ denotes the expected ground position of target i. One can also consider q_i as the center of the cloud of the track PDF of the t^{th} target. Without further notice, we will use i and j as the indexes for the targets and the MSAs respectively in the rest of this chapter. An example of the *multi-MSA-multi-target* (MMMT) scenario is shown in Fig. 7.1. In order to keep the targets in surveillance with limited MSA resources, the members of the MSA team have to move back and forth to update the targets' status. Therefore, the motion planning in such an MMMT environment has to consider not only how each MSA goes from one point to another, but also which target (or target set) each MSA should look after. This is essentially a combination of the problems of *sensor resource management* and *robot motion planning*. The main purpose of this research work is to seek a systematic framework for designing a real-time implementable motion-planning approach for the MMMT scenario.

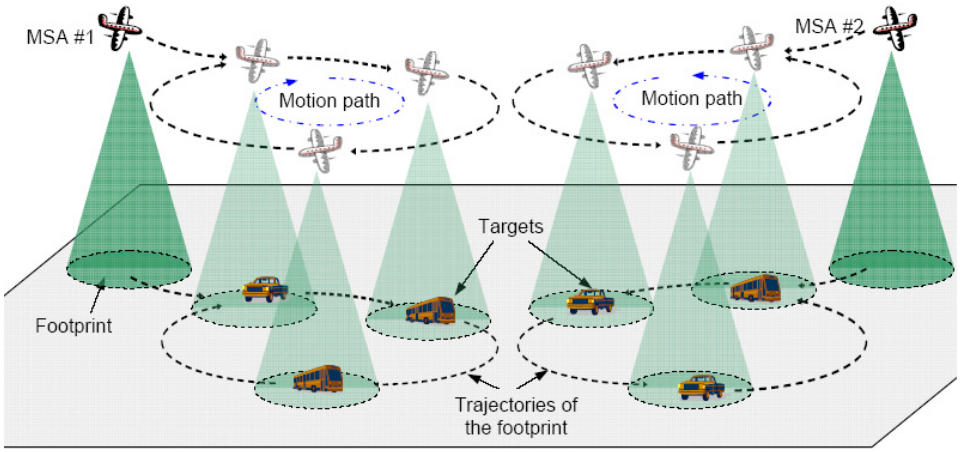


Figure 7.1: An example of the MMMT scenario.

7.1.1 Related Work

Although both the problems of *sensor resource management* and *robot motion planning* have been intensively studied for decades, very few available methods can be directly applied to the MMMT scenario.

Most previous *robot motion planning* approaches assume that a simple, specific origin-to-destination configuration is given to each agent, so that their main focus is usually on the optimal path generation between two predefined positions [27–36, 39, 40, 42–45]. However, with limited resources of MSA's, the task for each MSA often covers more than one target, which cannot be interpreted as a simple end-to-end configuration problem. Meanwhile, distinct from many traditional motion-planning applications (e.g. target search [25, 26, 78, 79], and target engagement [41, 86]), the surveillance job here does not end after each target is visited. The MSA's have to

come back to the targets repetitively to update their status. The optimal path for a given cycle is not necessarily optimal in the long run.

On the other hand, the majority of previous work in the area of *sensor resource management* treats the sensor control problem as (or similar to) a sensor scheduling problem [6–15], The management of sensors is used to be achieved by choosing different sensors (or sensor modes) for different tasks (targets) at different time. Motion planning is not involved in these approaches, which may not be a problem as these approaches are applied to sensor platforms with large, global coverage. As for mobile sensor platforms, however, controlling the sensors is not merely to assign them tasks or schedules, but to find them motion plans.

Among the very few approaches in dealing with cooperative multi-sensor motion planning, Parker et al. [49] formulated the motion planning problem as an optimization problem, whose objective is to maximize the collective time during which each target is monitored by at least one sensor agent. An approximation method based on the ALLIANCE architecture has then been proposed in [49]. The motion control of each agent is achieved in an implicit way by a force vector. The force vector is essentially a tradeoff among different sub-goals to keep the agents within certain distance to the targets as well as away from each other. Real-world experiments have demonstrated the feasibility of this approach with sufficient sensor resources. Jung et al., on the other hand, suggest a region-based approach for cooperative multi-target tracking in a structured environment in [50], in which the whole area of interest is assumed to be divided into topologically simple regions. The objective of individual motion control is formulated to locate each agent a certain distance from the center of gravity of targets that it is tracking. However, the cooperation among the robots within the

same region for multi-target tracking is not mentioned. Cortes et al. [51] studied the multi-sensor localization problem in a polygonal environment, and developed a gradient descent algorithm to realize optimal coverage and sensing policies. Each sensor agent is expected to converge to its optimal location and stay there. Similar to the other two methods mentioned previously, no motion constraints are considered, which makes it very difficult to apply these approaches directly to non-holonomic sensor agents with minimum-speed constraint, such as the *Dubins car*. Walker et. al addressed the multi-agent-multi-target path-planning problem for the *Dubins car* in [55]. In their approach, the coupled target assignment and path-planning problems are solved at the same time by searching over a tree of step-wise feasible flying paths. The real-time A^* algorithm [46] is used to find a sub-optimal path for each agent. By considering the effect of sensor footprint, this approach is capable of finding a sub-optimal path for each agent which is not necessary to pass the targets as long as they will be covered by the sensor footprint. Unfortunately, it is still difficult to fit this method into the multi-target surveillance problem addressed here due to the following reasons. First, although this method allows the targets to be visited multiple times, the number of visits on each target has to be pre-determined before path planning. Meanwhile, visiting a target multiple times does not necessarily lead to a good tracking performance. It is how these multiple observations are made in the time domain that determines the tracking performance. Furthermore, all the targets are assumed to be strictly static in Walker's approach [55]. Thus, the objective of path-planning is to generate the shortest feasible paths for the MSA's to traverse the targets once or multiple times. There is no re-planning scheme in dealing with a dynamic environment with moving targets. There are some other research work

that is related to the problem or a sub-problem of the problem addressed here, such as [26, 37, 38].

7.1.2 Main Contributions

Studies in Chapter 5 has shown that the uncertainty of a target system, which is represented by its entropy, is proportional to how frequently the target is observed. Considering the transition time for an MSA's switching from one target to another, we model the MSA management problem here as an optimization problem, whose objective is to minimize the average time duration (ATD) between two consecutive observations of each target:

$$J = \frac{1}{N} \sum_{i=1}^{N} ATD_i, \tag{7.2}$$

where ATD_i is the ATD between two consecutive observations of the i^{th} target over a sufficiently large time period.

It is worth noting that the formulation above is distinct from previous approaches [11,46,49–51,55], and can be applied to general surveillance and information gathering problems with limited sensor resources, in which one target can be a building, an intersection, a car under surveillance or a military unit.

Obviously, each components (i.e. ATD_i) in (7.2) is a function of future states of both the MSAs and the targets. As pointed out in [49], the general MMMT motion planning problem is NP hard both in the number of targets and in the number of sensor agents. Thus, looking for the global optimal solution is computationally prohibitive. In addition, since the target are mobile, a long term optimal motion plan is neither effective nor necessary. The desired MMMT motion-planning approach is probably a combination of an appropriate short-term planning method and a timely

re-planning scheme. Based on these considerations above, a sub-optimal motion-planning approach for the MMMT problem is proposed in this chapter.

The main contributions of this work are the following:

1. A computationally efficient gradient-based method is developed to determine a sub-optimal loop path for a single MSA to traverse multiple target points (i.e. the single-MSA-multi-target (SMMT) case). The core of this SMMT motion planning approach is the idea of searching for the optimal path with respect to the orientations of the MSA when it passes that targets. This method can also be applied to the general multi-target engagement problem.

2. A decentralized on-line motion-planning algorithm for multi-target tracking by multiple MSAs is proposed. In this algorithm, the targets are divided into M (i.e. the number of MSAs) disjoint groups based on a simple geometric clustering method. Each target group is assigned to a single MSA, and the sub-optimal traversal path generated by the aforementioned SMMT motion-planning method is considered as a short-term plan for each MSA. As the targets are moving around, on-line re-planning is conducted asynchronously on each MSA in a decentralized way. Target hand-off will be triggered among MSAs if new target-grouping result is obtained.

The rest of the chapter is organized as follows. The existing methods for the simplest single-MSA-single-target (SMST) case is briefly reviewed in section 7.2. Then, a sub-optimal path generation approach for a MSA of type *Dubins* to traverse a set of target points in minimum time (i.e. the SMMT case) is introduced in section 7.3. After that, a target-based on-line motion-planning algorithm for the MMMT case is

proposed in section 7.3, followed by the coverage stability of the approach discussed in section 7.5. Simulations and results are shown in section 7.6. Finally, conclusions are given in section 7.7.

7.2 Time-Optimal Motion Planning for Single Target Engagement (the SMST Case)

When there is only one MSA and one target, the motion-planning problem (7.2) is equivalent to the traditional single target engagement problem [27–36, 39]:

$$\begin{aligned} &\text{Minimize } J = \textstyle\int_{t_0}^{t_f} dt, \\ &\text{subject to equation (7.1)}, \\ &\text{with } s(t_0) = q_0,\ \vartheta(t_0) = \vartheta_0,\ s(t_f) = q_1,\ \vartheta(t_f) = \vartheta_1. \end{aligned} \tag{7.3}$$

Note that in different applications, the initial and final conditions (i.e. q_0, ϑ_0, q_1 and ϑ_1) can be either fixed or free. The earlier work of Dubins [27] has proved the existence of a time-optimal path for a system of type (7.1). More recently, Reeds and Shepps [28] have extended the work to the case that the robot can move both forward and backward. Meanwhile, Sussmann [33] and Boissonnat [34] have solved the problem using Pontryagin's Maximum Principle, which coincides with the following theorem given by Dubins in [27]:

Lemma 7.2.1: For the time optimal control problem described in (7.3) with any initial and final configurations, there exists a minimum-time trajectory χ^* for system (7.1), which is a combination of arcs from circles (which we shall denote as C) and straight line segments (which we shall denote as L). More specifically, the time-optimal path χ^* is a sub-path of a path of type Circle-Line-Circle (CLC) or of type Circle-Circle-Circle (CCC).

Practically, the minimum-time trajectory as well as the corresponding control u^* can be determined by simple geometric methods, which basically choose the shortest path from a finite set of extremal trajectories. An example of seeking the optimal trajectory is shown in Fig. 7.2. In this example, there are only 4 candidate paths that satisfy *Lemma 7.2.1*, which are C_LLC_R (i.e. a left-turn arc followed by a straight line and a right-turn arc), C_LLC_L, C_RLC_L and C_RLC_R. The minimum-time trajectory in this case is C_RLC_R. More complete discussions on how to geometrically synthesize the optimal trajectory can be found in [30–33].

Remark 7.2.1: The geometric method provides us a convenient way to specify the minimum-time trajectory for an end-to-end problem. This computationally low-cost feature of the geometric solution to the SMST problem is very helpful in developing our algorithm for the multi-target case.

7.3 A Sub-optimal Motion Planning Approach for the SMMT Case ($M = 1, N > 1$)

In this section, we focus on the case of a single MSA ($M = 1$) monitoring multiple targets ($N > 1$). Given a specific list of targets, a MSA has to move around and update the status of the targets one after another. Here we further assume that the targets are spread in the field of interest with considerable distance among each other. In the case that several targets are very close to each other, we can merge them can treat the group as a pseudo target. Thus, the corresponding motion plan for the MSA is a *traverse* path. By *traverse* here we mean one MSA visiting each target once along such a path. Since the surveillance job is not finished after one cycle, what we are looking for is actually a loop path. To minimize (7.2), the MSA has to execute the traverse loop as fast as possible. Similar to (7.3), we can rewrite (7.2) for this case as

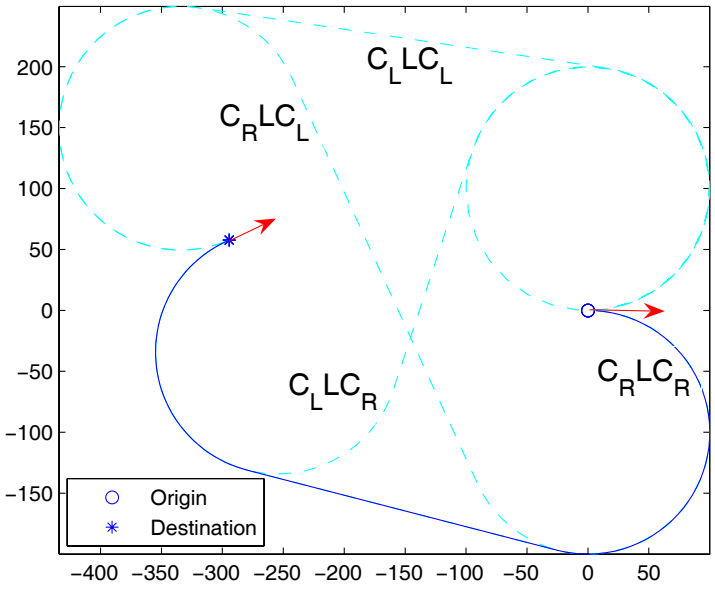


Figure 7.2: An example of finding the minimum-time trajectory.

the following time-optimal control problem:

$$\arg\min_{Q',u} J(Q'|Q' = \{q'_i\} \in \Omega) = \frac{1}{N}\sum_{i=1}^{N} T_i, \tag{7.4}$$

$$\text{subject to: equation (7.1)},$$

$$\text{with } s(t_0) = q'_0, \text{ and } s(t_0 + T_i) = q'_i,$$

where $q'_N = q'_0$, Ω is the collection of all permutations of the target set $Q = \{q_i\}$, T_i is the flying time between q'_{i-1} and q'_i due to the controller u.

Although motion planning for the single target case is quite a mature area, the extension of the available methods to the multi-target case is a non-trivial problem. As (7.4) implies, to find the optimal motion plan, one has to search for the best traverse order as well as the time-optimal trajectory for it. The number of possible

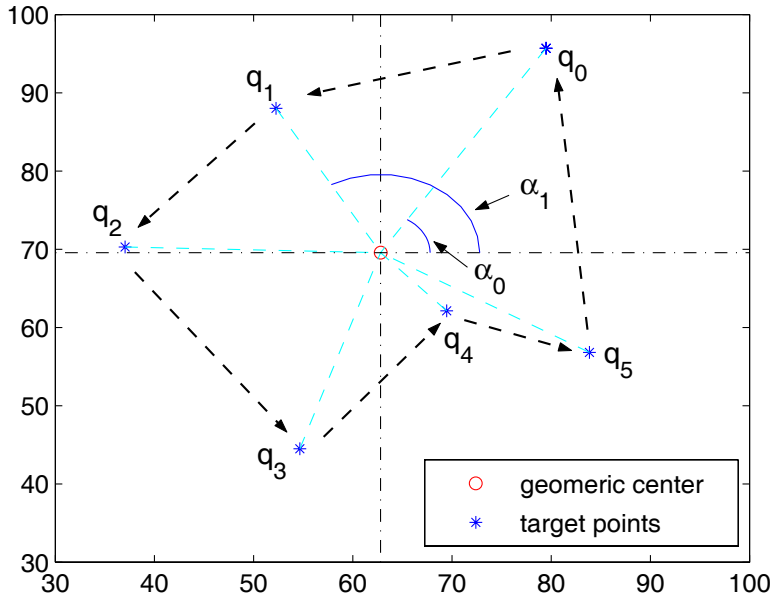


Figure 7.3: An example of determining the traverse order.

sub-optimal trajectories in the search space increases exponentially as the number of targets increases. For example, a set of N targets will lead to $(N-1)!$ distinct traverse order (rather than $N!$ due to the symmetry of a loop), which makes it very difficult to achieve the optimal path since the search space is literally $(N-1)!$ times bigger. Fortunately, since we are looking for a loop path in the MSA-target scenario, the desired path is usually in a circle pattern around the geometric center of the targets. Based on this heuristic observation, here we develop the following geometric method with much less computational complexity ($O(N \log N)$) to determine a sub-optimal the traverse order first.

7.3.1 Determination of the Traverse Order

The geometric method to determine the traverse order consists of three steps:

1. *Calculate the geo-center ($\bar{q}$) of the target set $Q = \{q_i\}$;*

2. *Calculate the orientation (α_i) of each target point (q_i) with respect to the center $\bar{q}$, as shown in Fig 7.3;*

3. *Sort the target points by $\{\alpha_i\}$ and the result is chosen as the traverse order.*

After determining the traverse order, the motion-planning problem for the SMMT case further reduces to a pre-ordered multi-target engagement problem.

7.3.2 Motion Planning for the SMMT Case with a Given Traverse Order by Approximated Gradient Decent

Definition 7.3.1: Given a target set $Q = \{q_i\}_{i=0..N-1}$, an *impact angle configuration* Θ is the set of impact angles $\Theta = \{\theta_i\}_{i=0..N-1}$, where θ_i denotes the orientation of the MSA as it passes target i.

Definition 7.3.2: A *motion plan* $\chi(Q, \Theta) = \{\chi_i(Q, \Theta)\}$ is an admissible motion trajectory for the MSA to traverse a pre-ordered target sequence $Q = \{q_i\}_{i=0..N-1}$, where $\chi_i(Q, \Theta)$ denotes the sub-path of $\chi(Q)$ from q_i to q_{i+1}. Note that in the case of a loop path, $q_N = q_0$.

The time for the MSA to execute a motion plan χ is then denoted as $J(\chi)$.

Definition 7.3.3: A motion plan $\chi(Q, \Theta)$ is a *Dubins path* if each sub-path $\chi_i(Q, \Theta)$ of $\chi(Q, \Theta)$ is the time-optimal path between q_i and q_{i+1} that satisfies (7.3).

According to *Lemma 7.2.1*, we know that for any given target set Q and its *impact angle configuration* Θ, there exists a *motion plan* that is a *Dubins path*. We denote this path as $\bar{\chi}(Q, \Theta)$.

Lemma 7.2.1 also leads to the following important properties of a *Dubins path*:

Property 7.3.1: Given two *impact angle configurations* (Θ, Θ') and the corresponding *Dubins paths*: $\bar{\chi}(Q, \Theta)$ and $\bar{\chi}(Q, \Theta')$, if $\theta_i = \theta_i'$ for all $i = 0..N-1$ except for $i = i^*$, then we have $\bar{\chi}_i(Q, \Theta) = \bar{\chi}_i'(Q, \Theta)$ for all $i = 0..N-1$ except for $i = i^* - 1, i^*$.

Property 7.3.2: Define $\Gamma(Q)$ as the collection of *Dubins paths* for all possible *impact angle configurations*: $\Gamma(Q) = \bigcup_{\Theta}\{\bar{\chi}(Q, \Theta)\}$, where $\bar{\chi}(Q, \Theta)$ denotes the *Dubins path* that traverses Q with *impact angle configuration* Θ. If a *motion plan* χ^* is the global minimum-time plan, it follows that $\chi^* \in \Gamma(Q)$.

Remark 7.3.1: *Property* 7.3.2 tells us that given any impact angle configuration Θ, the only candidate for the global time-optimal path is the *Dubins path* $\bar{\chi}(Q, \Theta)$.

Therefore, in order to obtain the time-optimal path, we can minimize $J(\bar{\chi}(Q, \Theta))$ by searching in the space $\Gamma(Q)$. An intuitive way to realize that is to use the gradient method. However, the nature of the *Dubins path* is space-dependent, which can not be formulated in a simple explicit way and so is $J(\bar{\chi}(Q, \Theta))$. Meanwhile, the cost function $J(\bar{\chi}(Q, \Theta))$ may not even be differentiable as some sub-paths of $\bar{\chi}$ change their types. To deal with this problem, we use the following gradient approximation method.

Consider two different impact angle configuration $\Theta = \{\theta_0, \theta_1, ..\theta_i, ..\theta_{N-1}\}$, and $\Theta' = \{\theta_0, \theta_1, ..\theta_i', ..\theta_{N-1}\}$. Let $\chi = \bar{\chi}(Q, \Theta)$ and $\chi' = \bar{\chi}(Q, \Theta'))$ be the corresponding *Dubins paths*. According to *Corollary* 1, we have:

$$J(\chi) - J(\chi') = J(\chi_{i-1}) + J(\chi_i) - J(\chi_{i-1}') - J(\chi_i'). \tag{7.5}$$

Based on (7.5), we can approximate the gradient function $J_{\theta_i}(\chi)$ as:

$$\frac{\partial J(\chi)}{\partial \theta_i} \approx \frac{J(\chi^+) - J(\chi^-)}{2\Delta\theta}$$

$$= \frac{J(\chi_{i-1}^{+}) + J(\chi_{i}^{+}) - J(\chi_{i-1}^{-}) - J(\chi_{i}^{-})}{2\Delta\theta}, \tag{7.6}$$

where $\chi^{+} = \bar{\chi}(Q, \{\theta_0, ..\theta_i + \Delta\theta, \theta_{i+1}, ..\theta_{N-1}\})$ and $\chi^{-} = \bar{\chi}(Q, \{\theta_0, ..\theta_i - \Delta\theta, \theta_{i+1}, ..\theta_{N-1}\})$.

Remark 7.3.2: One important feature of (7.6) is that the change of the impact angle of one target point q_i only affects two sub-paths of $\bar{\chi}$: $\bar{\chi}_{i-1}$ (from q_{i-1} to q_i) and $\bar{\chi}_i$ (from q_i to q_{i+1}). Thus, the computational complexity of calculating the approximated gradients $\nabla\tilde{J}_{\Theta}$ is just $O(N)$, which is equivalent to that of the traditional gradient method, rather than $O(N^2)$.

With the help of (7.6), we can search for a sub-optimal path in a recursive way as the traditional gradient method does. The corresponding update equation for Θ is:

$$\Theta(k+1) = \Theta(k) - \eta\nabla\tilde{J}_{\Theta}, \quad k \geq 0 \tag{7.7}$$

where η controls the convergence speed of the algorithm. The initial condition $\Theta(0)$ can be randomly chosen.

The approximated gradient method provides us a fast way to find a optimum. However, unlike the single target case, there are many local minima for the SMMT traverse problem. Fig. 7.4.a shows an example of sub-optimal path obtained by the approximated gradient method introduced above, while the global minimum-time path in this example is shown in Fig. 7.4.b.

To get a better local optimum, Yang et al. [38] suggest a set of empirical initial angular configurations that can preserve a good search performance. In this approach, we revise optimization procedure above by adding a *check-and-flip* procedure as follows.

7.3.3 Optimization Improvement

By carefully investigating the sub-paths that pass each target (as shown in Fig. 7.5), it is noticed that most of the local minimal paths have at least one such sub-path that is an arc larger than π (Fig. 7.5.a). For example, the sub-optimal path shown in 7.4.a has two large arcs as it passes the upper-left target and the bottom-right one. For each of these sub-pathes (e.g. Fig. 7.5.a), a better configuration often comes from a totally opposite impact angle (e.g. Fig. 7.5.b). Based on this observation, here we revise the approximated gradient method by adding an extra step to check the characteristics of a "sub-optimal" path when it converges to a local minimum.

According to *Lemma 7.2.1*, the sub-paths that comes into and goes out from one target point q_i are two arcs ($\widehat{AB}$ and $\widehat{AC}$ in Fig. 7.5). Denote the radians of the two arcs as β_i^{in} and β_i^{out}, as shown in Fig. 7.5. We revise the search algorithm as:

1. *Search the optimal path by the approximated gradient method until it converges to a local minimum;*

2. *Calculate the radians of the arcs passing each target point:* $\rho(i) = \mid c_i^{in}\beta_i^{in} + c_i^{out}\beta_i^{out} \mid$;

3. $i^* = argmax_i\{\rho(i)\}$,*where* $c_i^{in}, c_i^{out} = 1$ *if the arc is clockwise and* -1 *otherwise;*

4. *If* $\rho(i^*) \leq \pi$ *or* $k > T_k$, *quit, where* k *is the counter of the number of flips;*

5. *Flip the impact angle of target* i^* *over:* $\theta_{i^*} \leftarrow \theta_{i^*} + \pi$, $k = k + 1$, *and go back to step* 1).

The additional steps 2)-5) in the algorithm above is called the procedure of *check-and-flip*, which reverses the impact angle on a target when the sub-path that passes

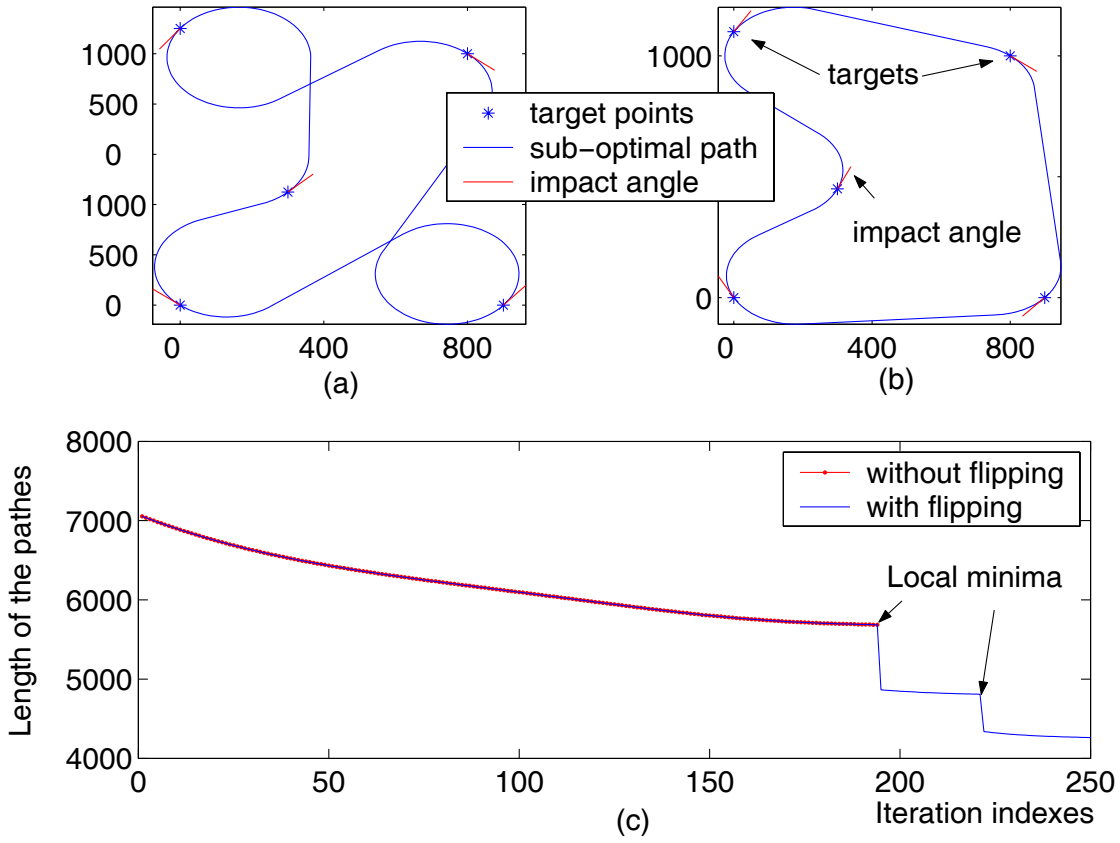


Figure 7.4: An example for the SMMT case ($N = 5$): a) The sub-optimal path without flipping; b) The sub-optimal path with flipping; c) Length comparison.

it an arc larger than π. As the example illustrated in Fig. 7.4 shows, the revised algorithm evolves exactly the same way as the approximated gradient method does until a local minimum is found. Without the check-and-flip procedure, the search will stop and a sub-optimal path is obtained (Fig. 7.4.a). Because of the flipping, this revised algorithm is capable of escaping from this local optimum and another one following that (Fig. 7.4.c), which helps the search converge to a shorter path as shown in Fig. 7.4.b.

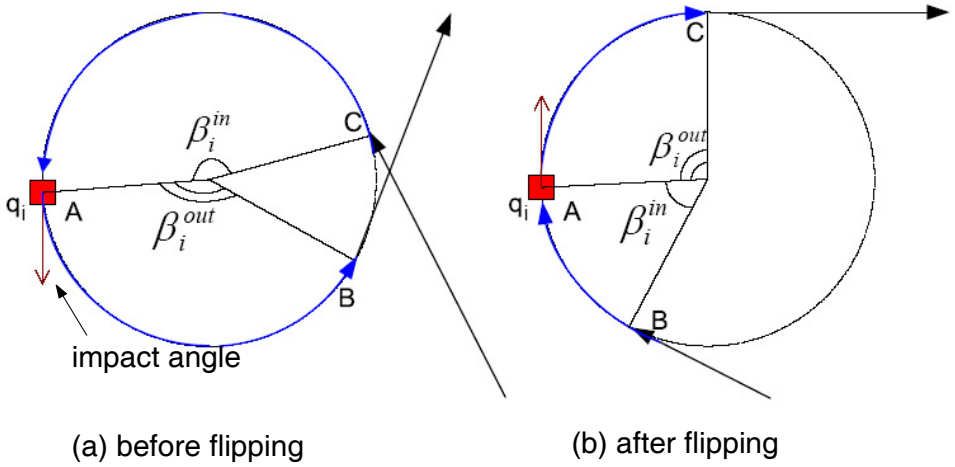


Figure 7.5: Illustration of the check-and-flip procedure: a) before flipping; b) after flipping.

It is worth noting that in some extreme cases, a flip does not necessarily lead to a better local optimum. Therefore, each local minimum is saved before flipping. We also limit the number of check-and-flip procedures within a certain threshold T_k, as indicated in step 4). The shortest path among the multiple local minima is then selected as the search result.

Remark 7.3.3: The *check-and-flip* procedure here can help to improve the optimization performance, but still does not guarantee the convergence to the global optimum. Especially, when two target points are close to each other, the effectiveness of the *check-and-flip* procedure is limited. In fact, it is not very efficient to visit two close targets individually. Taking advantage of the FOV of an MSA, a simple way to deal with two close target is to merge them together and replace them by a pseudo target. It is also worth noting that to merge the targets is not a trivial task if the target density is high.

Remark 7.3.4: In the case that the FOV of an MSA is very small, another possible way to deal with close targets is to fix the sub-path between two close targets by a straight line segment directly. By doing so, we actually fix the impact angles of these two targets. Fortunately, the approximated gradient method still applies to this situation, since the numerical gradient of each impact angle is computed independently in this method. Note that due to the minimum turning radius constraint, the straight line segment option can only be applied to every other pair of target points. In the case that all the targets are close to each other, the resulting traversal path is equivalent to that of the *Alternating Algorithm* introduced in [47].

Remark 7.3.5: Since the detail of the optimization process is not the main focus of this study, a generic gradient decent formulation is chosen in (7.7). A lot of practical methods can be applied to select a good step size, possibly in a dynamic fashion. There are also many other more sophisticated optimization tools other than gradient-based methods [98] that can be used to improve the optimization performance.

Remark 7.3.6: The path generation algorithm introduced here can be extended to the general multi-target engagement scenario with a given order, which is not necessarily to be a loop (i.e. $q_N \neq q_0$). The geometric method to determine the traverse order introduced earlier, however, is not suitable for the general multi-target engagement problems without a pre-determined order.

7.4 A Decentralized Cooperative Motion Planning Approach for the MMMT Case

The general MMMT motion planning problem is NP hard both in the number of sensor agents (M) and the number of targets (N) [49]. It is computationally prohibitive to find the optimal solution for motion planning. Furthermore, due to the

nondeterministic motion of the targets, it is even more difficult to make an optimal long-term motion plan for the MSAs. In this section, we propose a sub-optimal on-line motion planning approach with a lower computational load as follows.

In this scenario, the MSAs are the only sensor sources available. Without any prior knowledge, the motion of the targets is un-predictable. Here we use the most previous positional measurement of each target as its estimated position until it is renewed by the MSAs. Since we assume the MSA moves much faster than the targets, the possible movements of the targets are then regarded as system perturbations, which are adapted by adjusting the motion plans through on-line re-planning.

At each time, the motion planning consists of two steps: *Task Decomposition* and *Individual Path Generation*.

In *Task Decomposition*, the whole task of tracking N targets is divided into M disjoint task assignments. Then, given a task assignment, each MSA makes its own motion plan, which can be achieved by the approach introduced in the previous section.

7.4.1 Task Decomposition

Given a set of N elements, the total number of ways to partition these elements into M non-empty subsets is called the *Stirling Set Number* $S(N, M)$. We shall use this notion with the elements representing targets. The *Stirling Set Number* can be obtained by the following equation [57]:

$$S(N, M) = \frac{1}{M!} \sum_{j=0}^{M-1} (-1)^j C(M, j)(M - j)^N, \tag{7.8}$$

where $C(M, j)$ is a binomial coefficient.

Eq. (7.8) indicates that $S(N, M)$ increases exponentially as N or M increases, which makes the task decomposition an NP hard problem by itself. Since the desired motion plan for each MSA is a traverse loop and the ultimate goal is to have the MSAs execute their loop pathes in shortest time, it is more likely that the optimal partitioning divides the targets into clusters. Based on this consideration, we use the K-means clustering method here to realize the task decomposition in a recursive way. The details of the K-means clustering method can be found in [61].

After the task decomposition, we still have to distribute the M tasks (i.e. target groups) among the M MSAs. An optimal assignment is defined as the one that requires the shortest total time for the MSAs to catch up their tasks. To achieve such an optimal M-to-M assignment, we use Murty's *k-best* algorithm [58].

Remark 7.4.1: Although the M-to-M assignment still yields a considerable computational load, it only happens once in the initialization stage of the MMMT motion-planning problem. With the assumption that the MSAs move much faster than the targets, the majority of the new target assignments generated by on-line re-planning (see to the next subsection) will be consistent to the previous ones. Most MSAs will stay with their current target groups after re-planning, except that some targets may be switched to other groups. In implementation, the cluster center $\bar{q}_j$ from the previous planning cycle can be used as the default initial cluster center for MSA j's new motion plan. No M-to-M assignment is needed in the re-planning. Therefore, as far as the long-term performance is concerned, the cost of assignment initialization can be neglected. There are also several approaches suggesting an efficient way to implement the *k-best* assignment algorithm [59] [60].

7.4.2 Online Motion Planning

The motion plan for each MSAs is essentially a function of the expected positions of the targets (Q). In reality, a target may move away from its previous spot. It is necessary to have a corresponding re-planning scheme as the target information is updated. In this approach the motion re-planning is achieved in a decentralized way.

Define $Q_j(k) = \{q_i^j(k)\}$, $i = 1..n_j$ as the task assignment for MSA j, at time t_k. Since the status of one target $q_i^j(k)$ can only be updated when it is observed by MSA j, it is reasonable for the MSA to renew its traverse plan once the status of each target in its task assignment is updated (from $Q_j(k)$ to $Q_j(k+1)$). As one MSA re-considers its motion plan, it may decide to change its task assignment, which means the MSA may request some targets from or pass some targets to other MSAs. The exchange of target assignment here is called a procedure of *target hand-off*. In summary, a re-planning process is activated on one MSA by either one of the following events:

Event 1: A MSA finishes its current traverse loop

In this case, the MSA will reconsider it's task assignment by re-grouping the targets based on its own target information and the most recent information it obtained from other MSAs. Here we assume that the target information is shared by the whole MSA team. Note that to guarantee such a communication capability, the communication channel among the MSAs has to have a bandwidth no less than $O(MN)$ [49].

If there is no change in the task assignment (i.e. $Q_j(k+1) = Q_j(k)$), the MSA only has to adjust its own traverse loop path. Otherwise, a *target hand-off* is triggered.

Event 2: A MSA is requested for a *target hand-off*

In this case, the MSA will renew its target list ($Q_j(k+1)$), and then replan the optimal path for the new task. Note that there will be no negotiation between the MSA in this

approach, a MSA involved in a *target hand-off* will unconditionally accept the change request of its task assignment. There are several advantages of this non-negotiation scheme. First of all, there will be no dead-lock in the decision making among MSAs. Secondly, the task partitioning is always complete, since no target will be abandoned accidently during the *target hand-off*.

Remark 7.4.2: It is worth noting that for Event 1, one MSA will go through the whole motion-planning procedure, which including task re-decomposition (based on its own global information) and its own path re-generation. As for the result of Event 2, however, one MSA only has to update its own path. In summary, the on-line re-planning is achieved asynchronously by the MSAs. One MSA re-considers its motion plan only when either of the two events above happens.

Once a re-planning process is activated, the MSA will renew its traverse path according to the new task assignment, and catch it up from its current configuration, which is briefly summarized as follows:

1. *Renew the traverse order from* $Q_j(k)$ *to* $Q_j(k+1)$*:*

   In case of Event 1, keep $q_0^j(k)$ *as the start point:* $q_0^j(k+1) \leftarrow q_0^j(k)$*;*

   In case of Event 2, choose the next un-visited target point in the previous plan as $q_0^j(k+1)$*,*

2. *Renew the traverse path* $\bar{\chi}(Q_j(k+1), \Theta_j(k+1))$*;*

3. *Obtain the minimum-time path from the current configuration of the MSA* $\{s(t_k), \vartheta(t_k)\}$ *to the start point of the new traverse loop* $\{q_0^j(k+1), \theta_0^j(k+1)\}$*;*

4. *Execute the new traverse path* $\bar{\chi}(Q_j(k+1), \Theta_j(k+1))$ *from* $q_0^j(k+1)$ *to* $q_{n_j-1}^j(k+1)$ *until a new event is triggered.*

7.5 Coverage Stability Analysis

Stability and robustness is very important to a motion plan, and show how reliable the motion plan is in response to system perturbations. Some perturbations are caused by numerical errors when we generate the desired trajectory coordinates from the motion plan, or by process noises as we realize the trajectory in physical maneuvering. A good analysis on the stability and robustness of the time-optimal trajectory generation for a *Dubins car* can be found in [36]. To deal with the high sensitivity to modeling and measurement noises that a time-optimal path inherits, Turnau et al. proposed a near time-optimal planning method in [54]. Our aim in this section, on the other hand, is to investigate the stability in the sense of coverage, which is motivated by the question of whether the motion plan is able to maintain the target-tracking job as the targets are moving.

Without any prior knowledge, the motion of each target is un-predictable when it is outside the footprints of the MSAs. When an MSA traverses the estimated target positions (the previous positional measurements in this case), the hope is that each target is not far away, and is still covered by the footprint of the MSA. Obviously, this coverage is not un-conditionally guaranteed by any motion plan. The more frequently the MSA comes back to renew the target track, the more likely the target is close to the estimated position. If the target is not inside the sensor footprint, the MSA may have to search around for the target, rather than just take a single look as planned. As result, it will take more time to finish a traverse cycle for the MSA and lead to more possibility of missing the target in the future, which means the motion plan is unstable in this sense.

Definition 7.5.1: Assume that the on-board sensor system of an MSA is reliable (i.e. no miss detection or false alarm). The coverage of an MSA's motion plan is *stable* if the targets are guaranteed to be observed by executing the motion plan.

Consider a target observed at $q = (q_x, q_y)$. It obviously that the coverage is guaranteed if the "cloud" of the track PDF (i.e. the *survival zone* $S(t|t)$ defined in the previous chapter) can be covered by the sensor footprint. In particular, when there is no measurement noise, we have the following conclusion.

Lemma 7.5.1: The coverage of a motion plan is stable if

$$TD_i \leq \frac{D}{2v_t}, \quad \text{for all the } i = 1..N, \tag{7.9}$$

where TD_i is the time duration between two consecutive observations of target i, which is basically the length (in terms of time) of the traversal path that covers q_i. v_t is the maximum speed of a target, and D is the minimal diameter of the footprint of the MSA.

The proof of *Lemma 7.5.1* is fairly straightforward. As long as (7.9) is hold, $S(t|t)$ will be inside a circle of radius $v_t TD_i$, centered at q_i. Thus, no matter how the targets move, they will always be covered by the MSAs. It is worth noting the actual TD_i will be smaller when the measurement noise is taken into account.

Remark 7.5.1: Obviously, the coverage stability is not preserved in an open field as long as $N > M$, in which the targets can move in opposite dictions to break the coverage. Nevertheless, *Lemma 7.5.1* gives us a clue to evaluate the feasibility of a motion plan. For instance, if the time to execute a traversal path does not satisfies (7.9), an MSA has the option to reduce its surveillance load (e.g. to drop a few targets), or to ask for more MSAs to help the situation.

In the case that the targets are moving inside a bounded square region, we have the following conclusions for a *stable* motion plan:

Lemma 7.5.2: Given N targets moving inside a $L \times L$ square region, there exists a traversal motion plan for a single MSA whose coverage is stable if

$$(\sqrt{2N} + 1.75)L + \frac{1}{2}\kappa N \pi R_M \leq \frac{DV_M}{2v_t}. \tag{7.10}$$

Proof: As pointed out in [47], the SMMT motion planning problem is essentially a generalized version of the famous traveling salesmen problem (TSP). In fact the TSP can be considered as a special case of the SMMT motion planning problem addressed here, in which $R_M = 0$.

In the early work of Few [48], it has been shown that

$$ETSP(N, 2) \leq \sqrt{2N} + 1.75, \tag{7.11}$$

where $ETSP(N, 2)$ denotes the shortest TSP path for N point in a 2-D unit square.

Recently, Savla et al. [47] have proved that

$$ETSP(N, 2) \leq DTSP(N, 2) \leq ETSP(N, 2) + \frac{1}{2}\kappa N \pi R_M, \tag{7.12}$$

where $\kappa \leq 2.658$ and $DTSP(N, 2)$ denotes the shortest Dubins path for the N-target SMMT problem addressed in this chapter.

Combining (7.11), (7.12) and *Lemma 7.5.1*, we can get (7.10).

Consider the option that multiple MSAs move in a team formation along the same loop path that traverses N targets. The result above can be easily extended to the MMMT case as follows.

Theorem 7.5.3: Given N targets moving inside a $L \times L$ square region, there exists a traversal motion plan for M MSAs whose coverage is stable if

$$M \geq \frac{v_t[2(\sqrt{2N} + 1.75)L + 2.658N\pi R_M]}{DV_M}. \tag{7.13}$$

Remark 7.5.2: (7.13) gives us a sufficient condition to perform a MMMT surveillance task, which also implies the possibility of multiple MSAs' looking after the same group of targets. It is worth mentioning that the MMMT motion planning approach introduced in this chapter can be extended to incorporate this option. However, such extension will lead to more computational cost.

7.6 Experiments and Results

7.6.1 Path Generation for the SMMT Case

Two simulations for the SMMT case and their results are shown in Fig. 7.6, and Fig. 7.7. In simulation 1 (Fig. 7.6), four targets are generated. Three of them (target $1, 2, 3$) are modeled as random walks, while the other one (target 4) is a maneuvering target. As (Fig. 7.6) shows, the MSA is able to smoothly adjust its motion plan as the targets are moving around. Note that at some point, the MSA changes the traverse order from $1-3-4-2$ to $1-4-3-2$ to achieve a shorter traverse loop. Fig. 7.7 shows another example, in which there is a convoy of 3 moving targets. As the convoy moves, a smooth trajectory is generated for the MSA to follow up the convoy and keep updating the statuses of the targets.

The performance of the sub-optimal method has been examined by Monte Carlo simulations. Fig. 7.8 shows the results of one experiment, in which we chose $N = 6$ and conducted 500 MC simulations. The histogram of approximation errors is shown in Fig. 7.8. The actually minimum-time path for each simulation is achieved by exhaustively searching over all the permutations of the target points. In this experiment, most of the sub-optimal paths are no more than 10% longer than the actual one. The average error is only 2.44%, which is quite satisfactory.

7.6.2 Cooperative Online Motion Planning for the MMMT Case

In this experiment (Fig. 7.9), 4 MSAs, 16 randomly walking targets and 2 maneuvering targets are generated. The parameters are chosen as $R_M = 72m, V_M = 96m/s, v_t = \frac{1}{20}V_M$. The MSAs all start from the center of the field for simplicity, whose footprint is defined as a circle with a radius of $200m$ (i.e. $D = 400m$). As Fig. 7.9 shows, two *target hand-off* events are triggered as the two maneuvering targets move across the field. Note that as a decentralized algorithm, each MSA will asynchronously update its own motion plan if there is no *target hand-off*. When a target hand-off happens, it only affects the motion plans the MSAs that are involved in this hand-off.

The ATD between two consecutive observations of each target is shown in Fig. 7.11. The distance between each target and its expected position when the MSAs are making observations of the target is shown in Fig. 7.10. The result shows that the motion plan is successful in this example (i.e. $TD_i < D/2v_t = 41.7sec$ and $Dist_i < D/2 = 200m$, $\forall i = 1..N$). However, one can imagine that if target 17 and 18 keep moving froward, they will eventually break the coverage.

Given the configurations (i.e. D, R_M, V_M. v_t) of the targets and the MSAs in this experiment, the sufficient number of MSAs (given by *Theorem 7.5.3*) with respect to the number of targets (N) and the size of a squared region (L) is shown in Fig. 7.12. In particular, if the motion of all the targets in this experiment is limited inside the $2.2km \times 2.2km$ square region shown in Fig. 7.9, (7.13) indicates that a team of 5 MSAs can guarantee the surveillance coverage.

The performance of the task partitioning method has also been evaluated by Monte Carlo simulations. Fig. 7.13 shows the results of one experiment, in which $(N, M) = (12, 3)$ and 500 MC simulations are conducted. The results of the sub-optimal method based on K-means clustering is compared with those of the actual optimal partitions. The optimal partition is also achieved by exhaustive search. In this experiment, most of the sub-optimal paths are no more than 15% longer than the actual one. The average error is 11.21%.

7.7 Conclusions

This chapter addresses the motion-planning problem for multiple target surveillance with limited resources of MSAs. The kinematics of the MSA is modeled as a non-holonomic UAV of type *Dubins*. Based on the fact that the track information of each target degrades over time until it is renewed by the MSAs, the motion-planning problem here is formulated as an optimization problem, whose objective is to minimize the average time period between two consecutive observations of each target. Since the general optimal motion-planning problem for the MMMT case is NP hard, a computationally efficient sub-optimal approach is proposed in this paper.

The motion planning consists of two stages: *Task Decomposition* and *Individual Path Generation*. In *Task Decomposition*, the whole task of tracking N targets is divided into M disjoint task assignments, which is achieved by a heuristic method based on K-mean clustering. Then, given a task assignment, each MSA makes its own motion-plan, which is a SMMT motion-planning problem. The desired motion plan the SMMT case is formulated as a time-optimal loop path to traverse the targets. To find such a loop path, a particular family of trajectories is selected to compose a

reduced search space, in which we have proved that the optimal trajectory is always contained. Based on that, a gradient-based sub-optimal path generation algorithm is proposed for a mobile agent of type Dubins to traverse a sequence of target points. Meanwhile, a check-and-flip procedure is introduced to reduce the possibility of local minima. Furthermore, a decentralized online re-planning approach is also developed to deal with the situation that the targets are moving.

Experiments and simulations have demonstrated the effectiveness and efficiency of the proposed methods. The adoption of the proposed motion planning method to a real world test bed consisting of UAV's and ground robots that is under development is part of our ongoing research work.

The major results of this paper are not limited to this application only. The target-based motion-planning scheme can be extended to the cooperative control of multiple MSAs in the general information gathering scenario, in which one target can be a building, an intersection, a car under surveillance or a military unit. Meanwhile, the motion-planning algorithm for the SMMT case can be applied to the general multi-target engagement problem.

On the other hand, the extension of the proposed approach to more complicated situations with heterogeneous sensor agents and heterogeneous targets are open for further research. The discussions on coverage stability in this paper can also be extended to related topics such as task assessment and high-level sensor resource management.

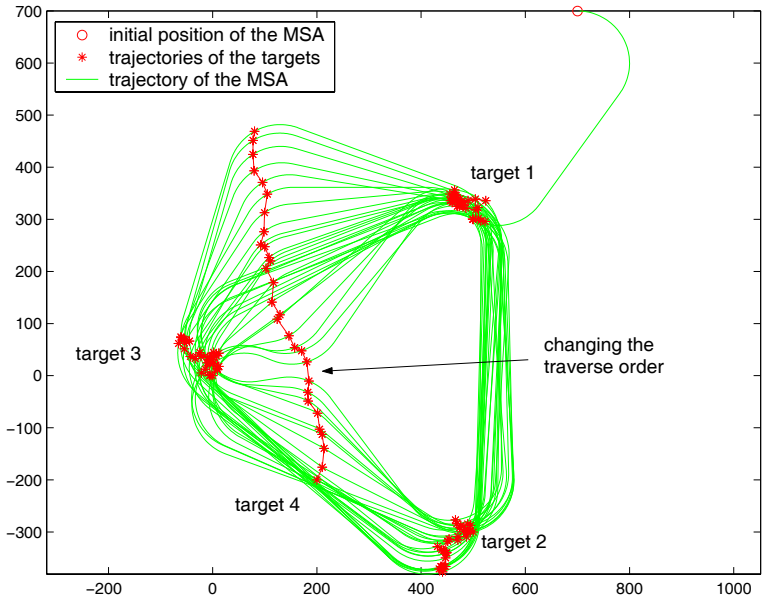


Figure 7.6: Simulation 1: One MSA, 3 randomly-walking targets and one maneuvering target.

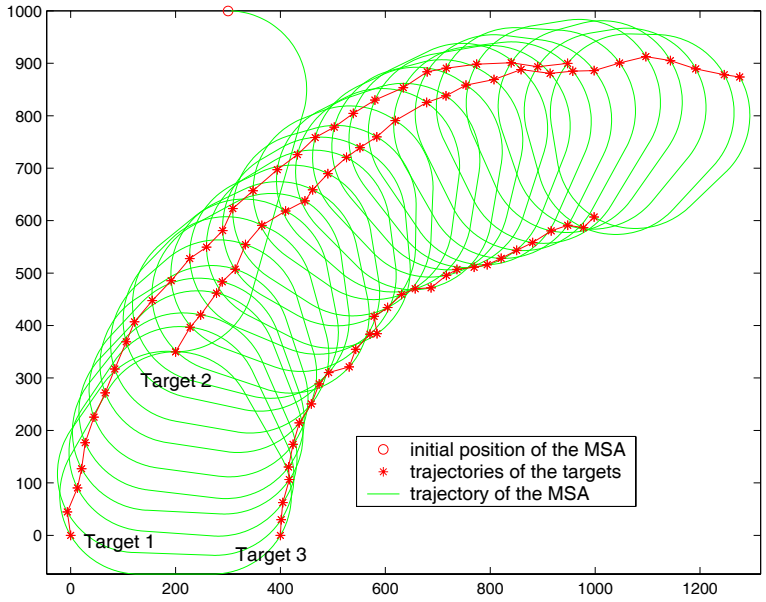


Figure 7.7: Simulation 2: One MSA and a moving convoy of 3 targets.

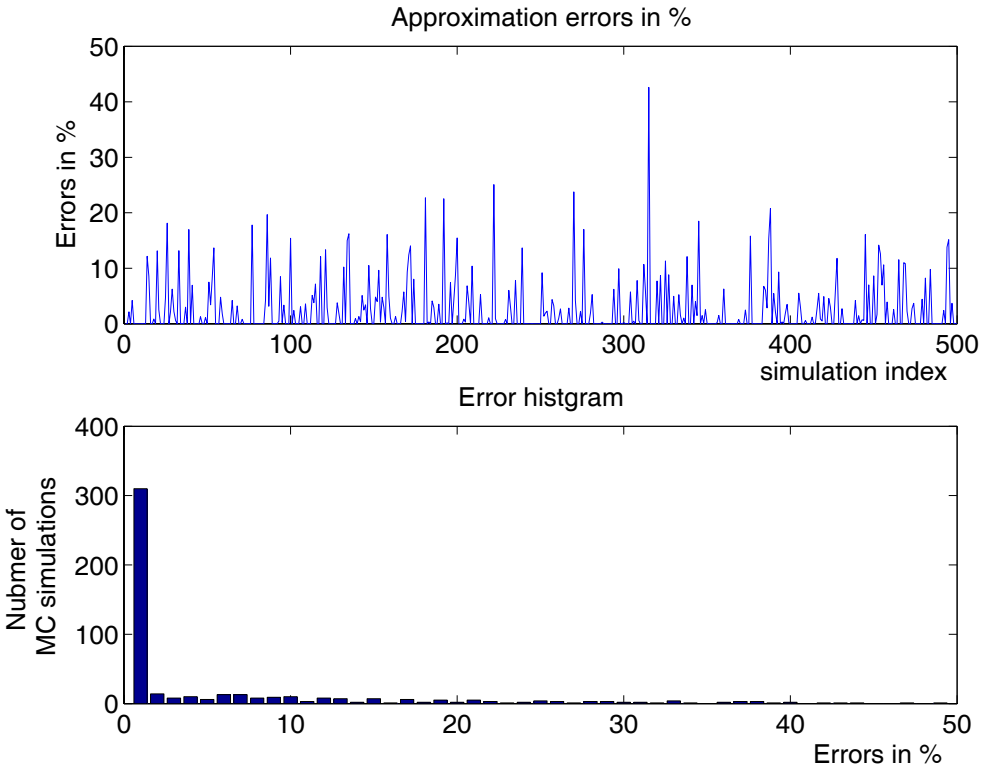


Figure 7.8: Performance evaluation for the geometrical method to determine the traverse order ($N = 6$, 500 MC simulations).

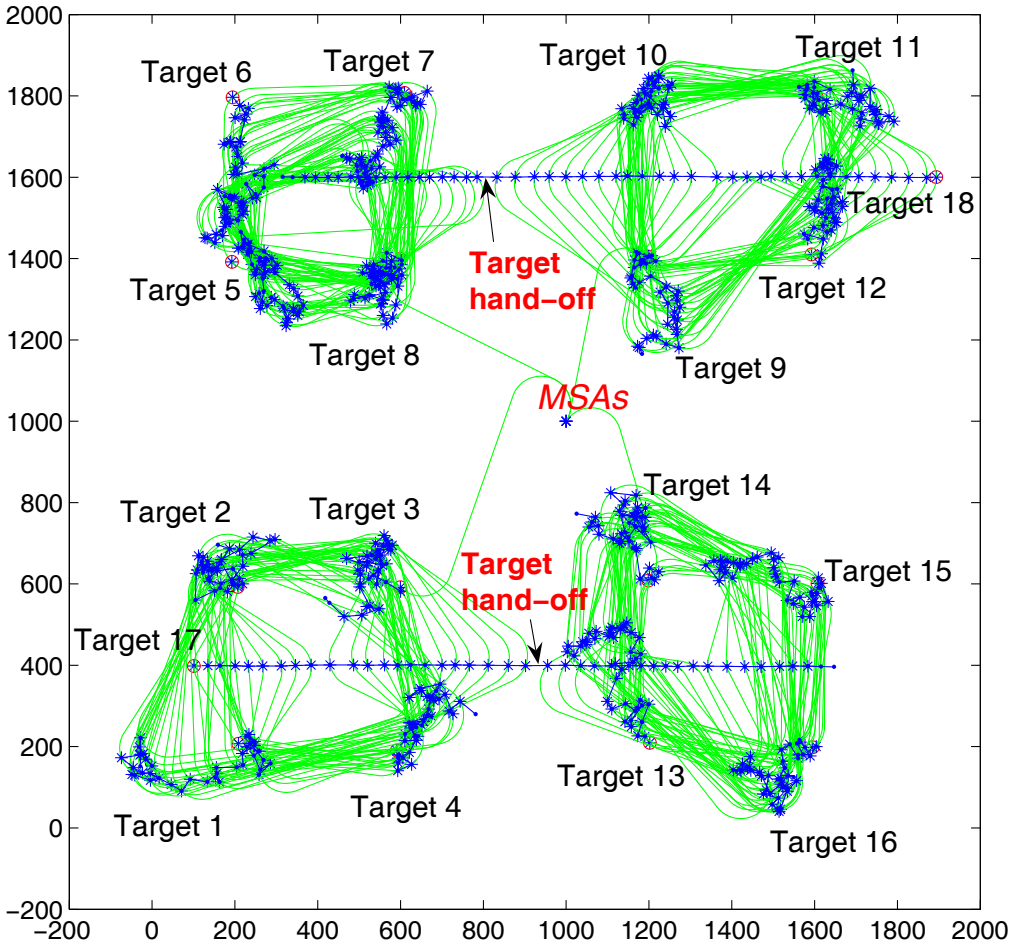


Figure 7.9: Simulation 3: 4 MSAs and 18 targets (16 randomly-walking and 2 maneuvering targets).

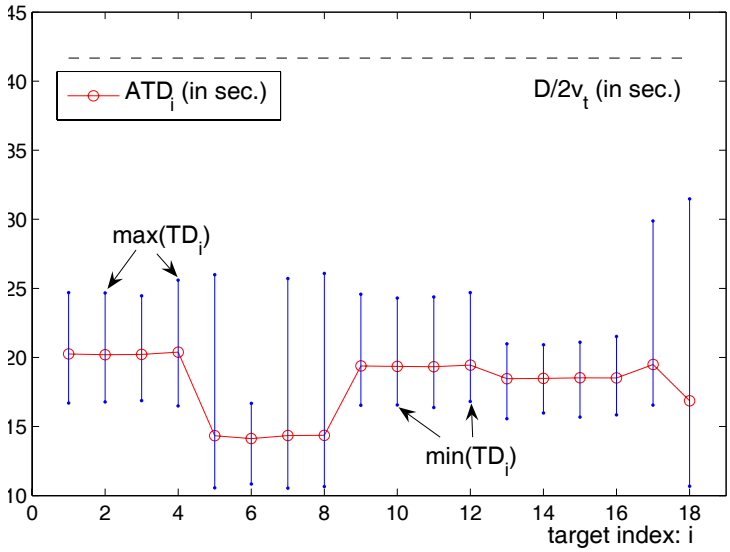


Figure 7.10: The ATD of each target in simulation 3.

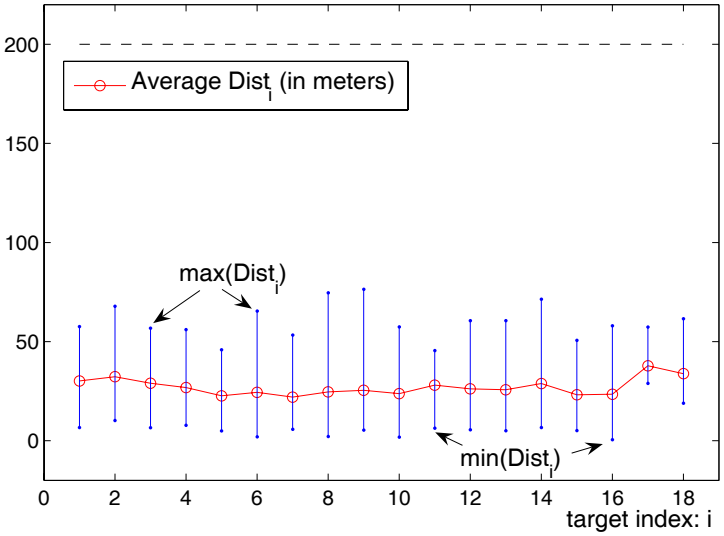


Figure 7.11: $Dist_i$: The distance between the i^{th} target and its expected position when it is observed in simulation 3.

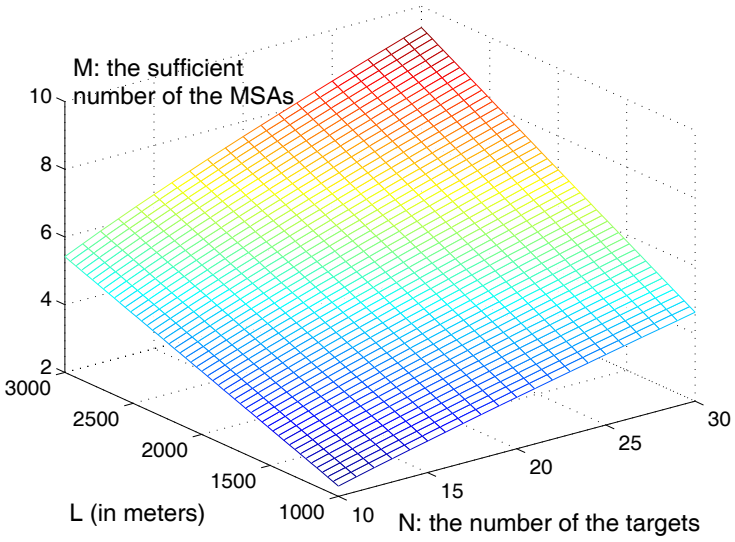


Figure 7.12: The sufficient number of MSAs given by *Theorem 7.5.3* with respect to N and L ($V_M = 96m/s$, $v_t = \frac{1}{20}V_M$, $R_M = 72m$, $D = 400m$).

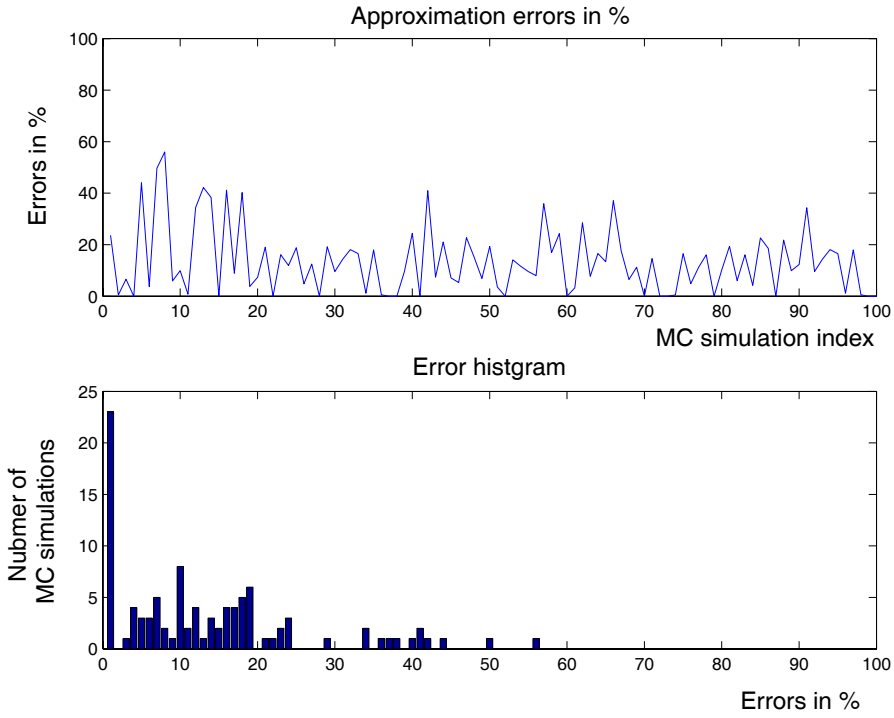


Figure 7.13: Performance evaluation for task assignment by K-means clustering ($N = 12$, $M = 3$, 100 MC simulations).

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this work, target track maintenance and MSA motion control, which are two key issues in the management of MSAs are jointly studied.

First, the BF-HMap approach is proposed as a generic method for target track maintenance based on the Bayesian filtering method and the *hospitability map*. To reduce the computational and memory load, the PF-HMap algorithm, which replaces the Bayesian filter in BF-HMap with a particle filter, is also introduced. Due to the flexible scheme of Bayesian inference inherent in both algorithms, BF-HMap and PF-HMap are capable of exploiting non-analytic prior environmental knowledge as well as handling intermittent and regional measurements caused by coverage and motion constraints of MSAs. In addition, a generalized particle filter for both in-sequence and out-of-sequence measurements, the so-called *Universal Particle Filter* (UPF), is developed for possible extensions of the PF-HMap algorithm to distributed sensor fusion in MSA networks.

In the meantime, the MSA motion control problem is studied in an information-theoretic way by choosing the conditional entropy (i.e. given the measurements) of the target state as a generic performance metric. Based on the analysis on the evolution of the entropy, strategic principles in modeling the MSA motion control

problem for two cases of studies, target search and target surveillance, are identified. More specifically, the target search problem is formulated as a stabilization problem of the entropy, and the target surveillance problem is modeled as to minimize the revisit time on the target. Both problems are further studied based on the entropy-based formulations above. A necessary condition and a sufficient condition by means of the number of MSAs for a non-escape search are derived. Given a sufficient number of MSAs, a cooperative search formation, called the *Progressively-Spiral-In* algorithm (PSI), is also presented for the MSA team to find the target in finite time. In the meantime, a cooperative motion-planning approach for multi-target surveillance by multiple non-holonomic MSAs is proposed.

As experiment results have demonstrated the feasibility of the aforementioned methodologies in various MSA-target applications, a comprehensive theoretical foundation for MSA management is still far from established. The current work presented in this dissertation just knocks on the door toward many other theoretical and practical issues in MSA management.

For instance, besides the OOSM problem, distributed particle filters also face a consistency challenge in practice. Due to the simulation-based nature of particle filters, the target track maintained by two identical particle filters are different even they share the same set of measurements without any communication delays. Therefore, it is crucial to keep the variations in estimation at an appropriate level so that multiple MSAs in a join task can make motion decisions that are compatible to each other.

In addition, although the BF/PF-HMap approach allows us to handle target search and target tracking problem in a universal way at the information processing end,

a similar treatment at the motion control end for jointly search and tracking is not available, yet. The preliminary stability study in Chapter 7 gives a sufficient condition keep the MSAs working in a "tracking" mode. However, how to recover from the loss of a target track (probably by a search algorithm) is not addressed.

Another direction in future work is the extension of the theoretic boundaries derived for non-escape search to non-homogeneous terrain surfaces, in which the evolution of the entropy as well as that of the *survival zone* becomes more complicated.

In a word, the study in the management of MSAs is a very rich research topic. A few pieces of this big puzzle are presented in this work, but more to be found, which probably requires collaborations and contributions of researchers from different domains.

BIBLIOGRAPHY

[1] R. Popoli, "The sensor management imperative", *Multitarget-Multisensor Tracking: Appications and Advances*, vol. II, pp. 325-392, Artech House, 1992.

[2] D. M. Buede and E. L. Waltz, "Issues in sensor management", *in Proc. IEEE* 5^{th} *International Stmposium on Intelligent Control*, pp. 839-842, Sept. 1990.

[3] J. M. Manyika and H. F. Durrant-Whyte, *Data Fusion and Sensor Management: A Decentralized Information-Theorectic Approach*, Ellis Horwood: New York, 1994

[4] G. W. Ng and K. H. Ng, "Sensor management - what, why and how", *Infomration Fusion*, vol 1, pp. 67-75, 2000.

[5] J. M. Nash, "Optimal allocation of tracking resources", *in Proc. IEEE Conf. Decision and Control*, vol. 1, pp. 1177-1180, New Orleans, LA, Dec. 1977.

[6] K. J. Hintz and E. S. McVey, "Multi-process constrained estimation", *IEEE Trans. Systems, Man, and Cybernetics*, vol. 21, no. 1, pp 434-442, Jan. 1991.

[7] G. A. McIntyre and K. J. Hintz, "An information theoretic approach to sensor scheduling", *in Proc. SPIE on Signal Procssing, Sensor Fusion and Target Recognition*, vol. 2755, pp. 304-312, Orlando, FL, Apr. 1996.

[8] D. A. Castanon, "Approximated dynamic programming for sensor management", *in Proc. IEEE conf. Decision and Control*, pp. 1202-1207, San Diego, CA USA, Dec. 1997.

[9] D. J. Cool, P. Gmytrasiewicz and L. B. Holder, "Decision-theoretic cooperative sensor planning", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 892-902, 1996.

[10] M. Kalandros, L. Y. Pao and Y. Ho, "Randomization and super-heuristics in choosing sensor sets for target tracking applications", *in Proc. IEEE conf. Decision and Control*, vol. 2, pp. 1803-1808, Pheonix, AZ USA, Dec. 1999.

[11] P. Vanheeghe, E. Duflos, P. E. Dumont and V. Nimier, "Sensor management with respect to danger level of targets", *in Proc. IEEE conf. Decision and Control*, vol 5., pp. 4439-4444, Orlando, FL USA, 2001.

[12] R. Evans, V. Krishnamurthy, G. Nair, and L. Sciacca, "Networked sensor management and data rate control for tracking maneuvering targets", *IEEE Trans. Signal Processing*, vol 53, no. 6, pp. 1979-1991, June 2005.

[13] D. Vaidya, J. Peng, L. Yang and J. W. Rozenblit, "A framework for sensor management in wireless and heterogeneous sensor network", *in Proc.* 12^{th} *IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pp. 155-162, April 2005.

[14] C. Kreucher, A. O. Hero, K. Kastella and D. Chang, "Efficient methods of non-myopic sensor management for multitarget tracking", *in Proc.* 43^{rd} *IEEE Conf. Decision and Control*, vol. 1, pp. 722-727, Dec. 2004.

[15] D. Lu, Y. Yao and F. He, "Sensor management based on cross-entropy in interacting multiple model Kalman filter", *in Proc. American Control Conference* vol. 6, pp. 5381 - 5386, June 2004.

[16] R. Johansson and N. Xiong, "Perception management - an emerging concept for information fusion", *Information Fusion*, vol.4, no. 3, pp.241-245, July 2003.

[17] B. Grocholsky, A. Makarenko and H. Durrant-Whyte, "Information-theoretic coordinated control for multiple sensor platform", *in Proc. IEEE International Conf. Robotics and Automation*, pp 1521-1526, Taipei, Taiwan, Step. 2003.

[18] S. Russell and P. Norvig, em Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, NJ, 1995.

[19] N. J. Nilsson, em Artificial Intelligence: A New Synthesis, Morgan Kaufmann, San Francisco, CA, 1998.

[20] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles", *IEEE Trans. Robotics and Automation*, vol. 18, no. 6, pp. 911-922, 2002.

[21] S. Rathinam and R. Sengupta, "A Safe Flight Algorithm for Unmanned Aerial Vehicles", to be presented in IEEE Aerospace Conference, Montana, 2004.

[22] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen and P. Burt, "Aerial video surveillance and exploitation", *in Proc. IEEE*, vol 89., no.10, pp 1518-1539, 2001.

[23] T. W. McLain, P. R. Chandler and M. Pachter, "A decomposition strategy for optimal coordination of unmanned air vehicles", *in Proc. American Control Conference*, vol. 1, no. 6, pp. 28-30, 2000.

[24] P. R. Chandler, M. Pachter and S. Rasmussen, "UAV cooperative control", *in Proc. American Control Conference*, vol. 1, pp. 50-55, 2001.

[25] K. E. Nygard, P. R. Chandler and M. Pachter, "Dynamic network flow optimization models for air vehicle resource allocation", *in Proc. American Control Conference*, vol. 3, pp. 1853-1858, 2001.

[26] C. Schumacher, P. R. Chandler, and S. R. Rasmussen, "Task allocation for wide area search munitions", *in Proc. American Control Conference*, vol. 3, pp. 1917-1922, 2002.

[27] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", *American Journal of Mathematics*, vol. 79, no. 3, pp. 497-516, 1957.

[28] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards", *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367393, 1990.

[29] G. M. Siouris and A. P. Leros, "Minimum-time intercept guidance for tactical missiles", *Control Theory and Advanced Thechnology*, vol. 4, no. 2, pp. 251-263, 1988.

[30] P. Soueres and J.-P. Laumond, "Shortest paths synthesis for a car-like robot", *IEEE Transactions on Automatic Control*, vol. 41 no. 5, pp. 672688, 1996.

[31] A. M. Shkel and V. J. Lumelsky, "Classification of the Dubins set", *Robotics and Autonomous Systems*, vol. 34, pp. 179-202, 2001.

[32] G. Desaulniers and F. Soumis, "An effcient algorithm to find a shortest path for a car-like robot", *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 819-828, 1995.

[33] H. J. Sussmann and W. Tang, "Shortest paths for the reeds-shepp car : A worked out example of the use of geometric techniques in nonlinear optimal control", sycon-91-10, Rutgers Univs., 1991.

[34] J. D. Boissonnat, A. Cerezo and J. Leblond, "Shortest paths of bounded curvature in the plane", *in Proc. IEEE International Conf. Robotics and Automation*, pp. 2315-2320, 1992.

[35] T. L. Song and S. J. Shin, "Time-Optimal Impact Angle Control for Vertical Plan Engagement", *IEEE Trans. Aerospace and Electronic Systems*, vol. 35, No. 2, pp. 738-742, 1999.

[36] A. Balluchi, A. Bicchi, B. Piccoli and P. Soueres, "Stability and robustness of optimal synthesis for route tracking by Dubins' vehicle", *in Proc.* 39^{th} *IEEE Conf. Decision and Control*, pp. 581-586, 2000.

[37] G. Yang and V. Kapila, "A dynamic-programming-styled algorithm for time-optimal multi-agent task assignment", *Proc. IEEE Conference on Decision and Control*, vol.2, pp. 1959-1964, 2001.

[38] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints", *in Proc. IEEE Conference on Decision and Control*, vol.2, pp. 1301-1306, 2002.

[39] J.-P. Laumond, *Robot Motion Planning and Contorl* Srpinger, 1998.

[40] J. Bellingham, A. Richard and J. P. How, "Receding horizon control of autonomous aerial vehicle", *in Proc. American Control Conference*, pp. 3741-3746, Anchorage, AK, May, 2002.

[41] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Cooperative path planning for multiple UAVs in dynamic and uncertain environments", *in Proc. the* 41^{st} *IEEE International Conf. Decision and Control*, vol. 3 , pp. 2816-2822, 2002.

[42] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots", *in Proc. IEEE International Conf. Robotics and Automation*, pp. 2619-2626, 2002.

[43] A. Bicchi and L. Pallottino, "On optimal cooperative conflict resolution for air traffic management systmes", *IEEE Trans. Intelligent Transportation Systems*, vol. 1, no. 4, pp. 212-222, 2000.

[44] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals", *IEEE Trans. Robotics and Automation*, vol. 14, no. 6, pp. 912-925, 1998.

[45] I. K. Nikolos, K.P. Valavanis, N. C. Tsourveloudis and A. N. Kostaras, " Evolutionary algorithm based offline/online path planner for UAV navigation", *IEEE Trans. System, Man, and Cybernetics - Part B*, vol. 33, no. 6, pp. 898-912, 2003.

[46] J. Howlett, M. Goodrich and T. McLain, "Learning real-time A* path planner for sensing closely-spaced targets from an aircraft", *Proc. AIAA Guidance, Navigation, and Control Conference*, no. 2003-5338, Austin, TX, Aug. 2003.

[47] K. Savla, E. Frazzoli and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins vehicle", *in Proc. American Control Conference*, pp. 791-796, June, 2005.

[48] L. Few, "The shortest path and shortest road through n points", *Mathematika*, vol. 2, no. 1, pp. 141-144, 1955.

[49] L. E. Parker and B. A. Emmons, "Cooperative multi-robot observation of multiple moving targets", *in Proc. IEEE International Conf. Robotics and Automation*, pp. 2082-2089, 1997.

[50] B. Jung and G. S. Sukhatme, "A region-based approach for cooperative multi-target tracking in a structured environment", *in Proc. IEEE International Conference on Intelligent Robots and Systems*, pp. 2764-2769, 2002.

[51] J. Cortes, S. Martinez, T. Karatas and F. Bullo, "Coverage control for mobile sensing networks", *IEEE Trans. Robotics and Automation*, vol. 20, no. 2, pp.243-255, 2004.

[52] Z. Tang and U. Ozguner, "Motion Planning for Multi-target Surveillance with Multiple Mobile Sensor Agents", To appear in *IEEE Trans. Robotics*, 2005.

[53] R. Morselli and R. Zanasi, "Positioning trajectory generator with nonlinear constraints", *in Proc. IEEE International Conference on Control Applications*, pp 1177-1182, Glasgow, Scotland, U.K., Sept. 2002.

[54] A. Turnau, M. Szymkat and A. Korytowski, "Robust near time-optimal trajectory planning by intermediate targets assignment", *in Proc. IEEE International Conference on Control Applications*, pp. 1159-1164, Glasgow, Scotland, U.K., Sept. 2002.

[55] D. Walker, T. McLain and J. Howlett, "Cooperative UAV target assignment using distributed calculation of target-task tours", *Theory and Algorithms for Cooperative Control*, World Scientific, 2004.

[56] M. Orlov, "Efficient generation of set partitions", http://www.cs.bgu.ac.il/~orlovm/papers/partitions.pdf, 2002.

[57] M. Abramowitz and I. A. Stegun, "Stirling numbers of the second kind", *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing, New York: Dover, pp. 824-825, 1972.

[58] K. G. Murty, "An algorithm for finding all the assignments in order of increasing cost, *Operations Research*, vol. 16, pp. 682-687, 1968.

[59] M. L. Miller, H.S. Stone and I.J. Cox, "Optimizing Murty's ranked assignment method", *IEEE Trans. Aerospace and Electronic System*, vol. 33, no. 3, pp. 851-86, 1997.

[60] D. Eppstein, "Finding the k shortest paths", *SIAM Journal of Computing*, vol. 28, no. 2, pp. 652-673, 1999.

[61] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford, England: Oxford University Press, 1995.

[62] M. S. Grewal, *Kalman Filtering: Theory and Practice*, Englewood Cliffs, NJ, Prentice-Hall, 1993.

[63] H. A. P. Blom, "An efficient decision-making-free filter for processes with abrupt changes", *IFAC Symp. Identification and System Parameter Estimation*, York, UK, July 1985.

[64] H. A. P. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for system with Markovian switching coefficients", *IEEE Trans. Automatic Control*, vol. 33, No. 8, pp. 780-783, 1988.

[65] A. Satish, R. L. Kashyap, "Multiple target tracking using maximum likelihood principle", *IEEE Trans. Signal Processing*, vol. 43, no. 7, pp. 1677-1695, 1995.

[66] L. D. Stone, C. D. Barlow and T. L. Corwin, *Bayesian Multiple Target Tracking,* Artech House, 1999.

[67] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, Norwood, MA 1999.

[68] K. Kastella, "Finite difference methods for nonlinear filtering and automatic target recognition", *Multitarget-Multisensor Tracking: Appications and Advances*, vol. III, pp. 233-258, Artech House, 2000.

[69] S. Challa and Y. Bar-Shalom, "Nonlinear filter design using Fokker-Planck-Kolmogorov probability density evolutions", *IEEE Trans. Aerospace and Electronic Systems*, vol. 36, no. 1, pp. 309-315, 2000.

[70] J. R. Layne and M. J. Eilders, "Robust stochastic estimation: automated prediction and search for UAV's", AFRL Tech. Report, LRIR 88SN01COR, 2002.

[71] B. O. Koopman, *Search and Screening: General Principle with Historical Applications*, Pergamon Press Inc., Elmsford, NY, 1979.

[72] L. D. Stone, *Theory of Optimal Search* (2nd ed.), Operations Research Society of America, ORSA Books, Arlington, VA, 1989.

[73] S. S. Browm, "Optimal search for a moving target in discrete time and space", *Operations Research*, vol. 28, no. 6, pp. 1275-1289, Nov. 1980.

[74] L. Tierney and J. B. Kadane, "Surveillance search for a moving target", *Operations Research*, vol. 31, no. 4, pp. 720-738, July 1983.

[75] A. R. Washburn, "Search for a moving target: the FAB algorithm", *Operations Research*, vol. 31, no. 4, pp. 739-751, July 1983.

[76] J. P. Le Cadre and G. Souris, "Searching Tracks", *IEEE Trans. Aerospace and Electronic Systems*, vol. 36, no. 4, Oct. 2000.

[77] V. V. Popovich, Y. A. Ivakin and S. S. Shaida, "Theory of search for moving objects", *in Proc. Oceans'02 MTS/IEEE*, vol. 3, pp. 1319-1329, Oct. 2002.

[78] M. M. Polycarpou, Y. Yang and K. M. Passino, "A cooperative search framework for distributed agents", *in Proc IEEE International Symposium on Intelligent Control*, pp. 1-6, 2001.

[79] M. Baum and K. Passino, "A search-theoretic approach to cooperative control for uninhabited air vehicles", *in Proc. AIAA GNC conf.*, Aug. 2002.

[80] L. D. Jacques, "Search, classification, and attack decisions for cooperative wide area search munitions", *in Proc. Cooperative Control Workshop*, Floria, Dec. 2001.

[81] D. Enns, D. Bugajski and S. Pratt, "Guidance and control for cooperative search", *in Proc. American Control Conference*, vol. 3, pp. 1923-1929, Anchorage, AK, May 2002.

[82] M. Flint, M. Polycarpou and E. Fernandez-Gaucherand, "Cooperative control for multiple autonomous UAV's searching for targets", *in Proc. the 41st IEEE Conf. Decision and Control*, vol.3, pp. 2823-2828, Dec. 2002.

[83] U. Y. Ogras, O. H. Dagci and U. Ozguner, "Cooperative control of mobile robots for target search", *in Proc. the 41st IEEE Conf. Mechatronic*, pp. 123-128, June 2003.

[84] D. J. Pack and B. E. Mullins, "Toward finding an universal search algorithm for swarm robots", *in Proc. IEEE/RSJ Conf. Intelligent Robots and Systmes*, vol.2, pp. 1945-1950, Oct. 2003.

[85] M. Flint, E. Fernandez-Gaucherand and M. Polycarpou, "Cooperative control for UAV's searching risky environments for targets", *in Proc. the 41st IEEE Conf. Decision and Control*, vol.4, pp. 3567-3572, Dec. 2003.

[86] R. W. Beard and T. W. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints", *in Proc. the 41st IEEE Conf. Decision and Control*, vol.1, pp. 25-30, Dec. 2003.

[87] C. K. Cheng and G. Leng, "Cooperative search algorithm for distributed autonomous robots", *in Proc. IEEE/RSJ Conf. Intelligent Robots and Systmes*, vol.1, pp. 394-399, Oct. 2004.

[88] P. B. Sujit and D. Ghose, "Search using multiple UAVs with flight time constraints", *IEEE Trans. Aerospace and Electronic Systems*, vol. 40, no. 2, Apr. 2004.

[89] Z. Tang and U. Ozguner, "Sensor fusion for target track maintenance with multiple UAVs based on Bayesian filtering method and Hospitability Map", *in Proc. the 41st IEEE Conf. Decision and Control*, vol.1, pp. 19-24, Dec. 2003.

[90] M. Gage and R. S. Hamilton, "The heat equation shrinking convex plane curves", *Journal of Differential Geometry*, vol. 23, no. 1, pp. 6996, 1986.

[91] M. Grayson, "The heat equation shrinks embedded plane curves to round points", *Journal of Differential Geometry*, vol. 26, no. 2, pp. 285-314, 1987.

[92] B. Chow and D. Tsai, "Geometric expansion of convex plan curves", *Journal of Differential Geometry*, vol 44, no. 2, pp. 312-330, Sept. 1996.

[93] M. Spivak, *A compredensive introduction to differential geometry*, vols. I-V, Second edition, Publish or perish, Inc., Wilmington, DE, 1979.

[94] G. Unal, D. Nain, G. Ben-Arous, N. Shimkin, A. Tannenbaum and O. Zeitouni, "Algorithms for stochastic approximations of curvature flows", in *proc. International Conf. Image Processing*, vol. 2, pp. 651-654, Sept. 2003

[95] G. G. Magaril-Ilyave and V. M. Tikhomirov, *Convex Analysis: Theory and Applications*, American Mathematical Society, Providence, RI 2003.

[96] R.C. Yates, *A Handbook on Curves and Their Properties*, J. W. Edwards Ann Arbor, MI, 1952.

[97] F. Jr. Ayres, *Theory and Problems of Matrices*, New York, Schaum, 1962.

[98] M. A. Bhati, *Practical Optimization Methods with Mathematica Applications*, New York: Springer-Verlag, 2000.

[99] Y. Bar-Shalom, H. Chen and M, Mallick, "One-step solution for the multi-step out-of-sequence-measurement problem in tracking", *IEEE Trans. Aerospace and Electronic Systems*, vol. 40, no. 1, 2004.

[100] M. Mallick, T. Kirubarajan and S. Arulampalam, "Out-of-sequence measurement processing for tracking ground target using particle filters", *in Proc. IEEE Conf. Aerospace Conference*, vol. 4, pp. 1809-1818, 2002.

[101] L. Hong; S. Cong; D. Wicker, "Distributed multirate interacting multiple model fusion (DMRIMMF) with application to out-of-sequence GMTI data", *IEEE Trans. Automatic Control*, vol. 49, no. 1, pp. 102-107, 2004.

[102] A. Doucet, N. de Freitas and N. Gordon (Eds.), "Sequential Monte Carlo Methods in Practice", Springer, New York, 2001.

[103] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson and P. J. Nordlund, "Particle filters for positioning, navigation, and tracking", *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 425-437, 2002.

[104] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filtering for online nonlinear/non-Gaussian Bayesian tracking", *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174-188, 2002.

[105] S. McGinnity and G. W. Irwin, "Multiple model bootstrap filter for maneuvering target tracking", *IEEE Trans. Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 1006-1012, 2000.

[106] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[107] J. Geweke, "Bayesian inference in econometric models using Monte Carlo integration", *Econometrica*, vol. 24, pp. 1317-1399, 1989.

[108] H. Chen, T. Kirubarajan and Y. Bar-Shalom, "Performance limits of track-to-track fusion versus centralized estimation: theory and application, *IEEE Trans. Aerospace and Electronic Systems*, vol. 39, no. 2, pp. 386-400, 2003.

[109] N. N. Okello and S. Challa, "Joint sensor registration and track-to-track fusion for distributed trackers", *IEEE Trans. Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 808-823, 2004.

[110] C. Chong, S. Mori, W. H. Barker, and C. K. Chang, "Architectures and algorithms for track association and fusion", *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 1, pp. 5-13, 2000.

[111] D. Schulz, W. Burgard, D. Fox and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle fitlers and statistical data association", *in Proc. IEEE Conf. Robotics and Automation*, pp. 1665-1670, 2001