Distributed Networked Sensing and Control Systems: Robust Estimation and Real-time Control

Songhwai Oh

Distributed Networked Sensing and Control Systems: Robust Estimation and Real-time Control

by

Songhwai Oh

B.S. (University of California, Berkeley) 1995
M.S. (University of California, Berkeley) 2003

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Shankar Sastry, Chair
Professor Stuart Russell
Professor Bin Yu

Fall 2006

The dissertation of Songhwai Oh is approved:

Professor Shankar Sastry, Chair	Date
Professor Stuart Russell	Date
Professor Bin Yu	Date

University of California, Berkeley

Fall 2006

Distributed Networked Sensing and Control Systems: Robust Estimation and Real-time Control

Abstract

Distributed Networked Sensing and Control Systems:
Robust Estimation and Real-time Control
by
Songhwai Oh
Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences
University of California, Berkeley
Professor Shankar Sastry, Chair

There is a growing interest in distributed networked sensing and control systems, such as wireless sensor networks, networked control systems, distributed control systems, multi-agent systems, and heterogeneous sensor networks. A distributed networked sensing and control system consists of a number of autonomous agents. Information among these agents is usually shared by wireless communication. However, each agent is resource-constrained, *e.g.*, it may have limited processing power, storage capacity, and communication bandwidth. These constraints create measurement inconsistency and communication unreliability and they are the major obstacles in realizing an autonomous distributed networked sensing and control system which is capable of real-time situation understanding and control.

In this thesis, we take an advantage of spatio-temporal correlation among the neighboring agents and develop robust real-time algorithms for situation understanding and control. We also design and implement a real-time situation understanding and control system using wireless sensor networks.

For this purpose, we use the mathematical frameworks of multi-target tracking and pursuit evasion games. Multi-target tracking is a general framework which can be used to describe many estimation and inference problems appearing in distributed networked sensing and control systems. The pursuit evasion game is a mathematical framework for many challenging control problems and it can be viewed as the worst-case control problem.

The thesis starts with the simplest distributed estimation problem in a sensor network. After showing that the method cannot be applied to more general multi-target tracking problems, we develop a general Bayesian framework for multi-target tracking problems. The Bayesian framework allows a method which is robust against inconsistency in measurements and missing measurements due to communication unreliability. Since the exact computation of Bayesian estimates is a time-consuming task, we develop an approximate method, called Markov chain Monte Carlo data association, to efficiently solve the data association problems appearing in multi-target tracking problems.

Markov chain Monte Carlo data association is also used to improve the robustness of the multi-target tracking methodology by compactly managing identities of multiple objects using the identity-mass-flow framework. We then develop a real-time hierarchical control system with multiple layers of data fusion to solve the multi-target tracking and pursuit evasion games using a distributed networked sensing and control system. This thesis presents the first demonstration of multi-target tracking using a wireless sensor network without relying on classification.

We also present a general framework for modeling a distributed networked control system consisting of multiple agents communicating over a lossy communication channel. We describe exact and approximate filtering methods to estimate states of a distributed networked control system. In addition, we describe how to find a communication control which stabilizes a distributed networked control system.

Professor Shankar Sastry, Chair Date

Acknowledgements

This thesis is made possible by all my teachers, colleagues, and friends and I would like to thank them all.

First and foremost, I would like to thank my advisor, Professor Shankar Sastry, for his guidance and encouragement. I have been always inspired by his enthusiasm toward science and the depth and breath of his research. His interests, advices, and insightful suggestions have helped me complete this thesis.

I also would like to thank Professor Stuart Russell. My research, including this thesis, has benefited tremendously from many discussions with him. His expertise and ingenious questions and suggestions have been inspirational and kept me motivated. I also would like to thank Professor Bin Yu for being in my qualification exam and dissertation committees. Her expertise and comments have helped me greatly in making this thesis.

During my graduate years in Berkeley, I have had a chance to work with many extraordinary researchers. I thank them all. Luca Schenato helped me with applying my research to many practical problems and I thank him for all the comments and discussions I had with him. I also thank Inseok Hwang for giving me an opportunity to apply my work to problems in aeronautics. I also would like to thank Phoebus Chen for his dedication in research and letting me work with him. I also would like to thank Jin Kim for introducing me to the multiple target tracking problem and many delightful discussions. I also would like to thank Professor Pravin Varaiya, Professor Ruzena Bajcsy, and Professor Laurent El Ghaoui for helping me with many aspects of my research.

I also would like to thank all my friends in the Robotics and Intelligent Machines Laboratory, including Parvez Ahmmad, Alessandro Abate, Aaron Ames, Hoam Chung, Mike Eklund, Christopher Geyer, Jianghai Hu, John Koo, Jongho Lee, Marci Meingast, Tanya Roosta, Shawn Shaffert, David Shim, Bruno Sinopoli, Jonathan Sprinkle, Todd Templeton, and Bonnie Zhu. I would like to thank everyone who worked for the NEST project and made the project and final experiment successful: Professor David Culler, Prabal Dutta, Eric Fraser, Mike Howard, Jonathan Hui, Jaein Jeong, August Joki, Sukun Kim, Philip Levis, Michael Manzo, Joseph Polastre, Travis Pynn, Peter Ray, Cory Sharp, Jay Taneja, Gilman Tolle, Robert Szewczyk, and Kamin Whitehouse. I also would like to thank my fellow graduate students in EECS at U.C. Berkeley. It was a great pleasure to work with so many extraordinary individuals and I am looking forward to continue working with them for many more years to come.

Lastly, but most importantly, I would like to thank my parents for their endless love and encouragement.

To my parents

Contents

Chapter 1

Introduction

1.1 Sensor Networks

Recently, we have been witnessing dramatic advances in micro-electromechanical sensors (MEMS), digital signal processing (DSP) capabilities, computing, and low-power wireless radios which are revolutionizing our ability to build massively distributed, easily deployed, self-calibrating, disposable, wireless sensor networks [Estrin *et al.*, 2002; Akyidliz *et al.*, 2002; Gharavi and Kumar, 2003]. Soon, the fabrication and commercialization of inexpensive millimeter-scale autonomous electromechanical devices containing a wide range of sensors, including acoustic, vibration, acceleration, pressure, temperature, humidity, magnetic, and biochemical sensors, will be readily available [Warnake *et al.*, 2002]. These potentially mobile devices, called "nodes", are provided with their own power supply [Roundy *et al.*, 2004] and can communicate with neighboring sensor nodes via low-power wireless communication to form a wireless ad-hoc sensor network with up to 100,000 nodes [Estrin *et al.*, 2001; Culler *et al.*, 2004]. Sensor networks can offer access to an unprecedented quantity of information about our environment, bringing about a revolution in the amount of control an individual has over his environment. The ever-decreasing cost of hardware and steady improvements in software will make sensor networks ubiquitous in many aspects of our lives [TR, 2003]. Throughout this thesis, the term *sensor networks* refers to the *wireless sensor networks*.

Applications

The applications of sensor networks include, but not limited to, environmental monitoring [Szewczyk *et al.*, 2004; Zhang *et al.*, 2004], structural health monitoring [Pakzad *et al.*, 2005], healthcare [Lorincz *et al.*, 2004], building comfort control [Kintner-Meyer and Conant, 2005], traffic control [Nekovee, 2005], wildfire monitoring [Doolina and Sitara, 2005], manufacturing and plant automation [Willig *et al.*, 2005], service robotics [LaMarca *et al.*, 2002], and surveillance systems [Brooks *et al.*, 2004; Arora *et al.*, 2004]. A sensor network is an attractive solution for many applications due to its flexibility, wireless communication capability, and the small size of each sensor node. Considering the fact that the installation costs can represent 20% to 80% of the overall cost of data acquisition and control systems [Kintner-Meyer and Conant, 2005], the cost reduced by using a wireless sensor network alone will have a significant impact on every application.

Based on how sensor data is collected and used, applications of sensor networks can be categorized as following:

1. *Data collection*: Unlike the traditional sensors, many of these small sensor nodes in a sensor network can be placed near the source of interest, providing a dense set of data. This makes a sensor network an ideal tool for collecting data for many scientific studies. Examples include ZebraNet for monitoring the migration pattern of zebras [Zhang *et al.*, 2004], habitat monitoring on Great Duck Island and microclimate monitoring of California redwood forests [Szewczyk *et al.*, 2004].

2. *Event detection*: In data collection, a sensor network is solely used to collect sensor data and an analysis of the collected data is done at a later time. But an event

detection system uses a sensor network to collect sensor data and detect abnormal behaviors, such as structural health monitoring of bridges [Pakzad *et al.*, 2005], vital sign monitoring in health care [Lorincz *et al.*, 2004], wildfire monitoring [Doolina and Sitara, 2005], and surveillance systems [Brooks *et al.*, 2004; Arora *et al.*, 2004].

3. *Real-time situation understanding and control*: A real-time situation understanding and control system is similar to an event detection system. But instead of simply reporting a detection of an abnormal behavior, a real-time situation understanding and control system makes decisions within the system. Examples include manufacturing and plant automation [Willig *et al.*, 2005], traffic control [Nekovee, 2005], building comfort control [Kintner-Meyer and Conant, 2005], service robotics [LaMarca *et al.*, 2002], and surveillance systems. It can be argued that the realization of real-time situation understanding and control systems using sensor networks will have a dramatic societal impact on our lives.

Challenges

Each sensor node is resource-constrained. The limited supply of power and other constraints, such as manufacturing costs and limited package sizes, limit the capabilities of each sensor node. For example, a typical sensor node has limited processing power, storage capacity, and communication bandwidth. These constraints create measurement inconsistency, such as noise and false alarms, and communication unreliability, such as transmission failures and delays. Hence, they are the major challenges in realizing a sensor network system. The impact of these constraints is greater as we introduce more decision-making capabilities into a sensor network. This is why there have been some successful deployments of data collection systems and event detection systems while there has been no real-time situation understanding and control system using a sensor network.

As we introduce more real-time decision-making capabilities into a sensor network, we need to have a better understanding of the situation in real-time. Considering measurement inconsistency and communication unreliability inherent in sensor networks, the task of real-time situation understanding and decision-making seems to be an unreachable goal. However, the abundant number of spatially spread sensors and a carefully designed system based on robust real-time algorithms can make real-time situation understanding and decision-making possible using sensor networks; and this is the goal of this thesis. The main objectives of this thesis are

- development of robust real-time algorithms for situation understanding and control;
- design and implementation of a real-time situation understanding and control system using wireless sensor networks; and
- evaluation of real-time situation understanding and control systems.

There are other challenges in developing sensor network system that are not addressed in this thesis. On the hardware side, we need an inexpensive sensor node which operates with low power consumption for a long-term deployment. On the software side, we need reliable and robust communication protocols, time synchronization, and calibration of a large-scale distributed system. Other challenges include security and privacy introduced by using the wireless communication medium.

1.2 Distributed Networked Sensing and Control Systems

In this thesis, a wide range of distributed and networked systems are considered, such as sensor networks, networked control systems, distributed control systems, and multi-agent systems. The methodology developed in this thesis is currently applied to heterogeneous sensor networks. A heterogeneous sensor network is a combination of high-bandwidth, low-bandwidth, and mobile sensors. The high-bandwidth sensors include the image sensors for in-depth situation awareness and recognition. In this thesis, we use the term *distributed networked sensing and control systems* to refer all such systems.

Robust Estimation and Real-time Control

As mentioned earlier, the main objectives of this thesis are the evaluation and development of a real-time situation understanding and control system. In order to formulate real-time situation understanding problems in sensor networks, we need a sound mathematical framework. For this purpose, we use the mathematical framework of multi-target tracking throughout this thesis. Multi-target tracking is a general framework which can be used to describe many estimation or inference problems appearing in sensor networks. In multi-target tracking, there are many stochastic or random processes whose states are stochastically updated. Based on noisy measurements from these processes, the goal of multi-target tracking is to accurately estimate the states of all processes. In general, two stochastic models are used in multi-target tracking: dynamic and measurement models. The multi-target tracking problem includes single-target tracking and static process estimation problems. Multi-target tracking enjoys the flexibility of allowing different measurement models for different sensors and different dynamic models for different stochastic or random processes. Since the general formulation of multi-target tracking does not assume a constant number of processes, this is truly the general mathematical framework for estimation and inference problems in sensor networks.

In order to overcome measurement inconsistency and communication unreliability of a distributed networked sensing and control system, we need a robust estimation algorithm. In this thesis, we formulate the estimation problem using the Bayesian framework based on carefully designed prior models. The Bayesian framework allows a method which is robust against inconsistency in measurements and missing measurements due to communication unreliability. Since the exact computation of Bayesian estimates is a time-consuming task, this thesis proposes an efficient and good approximation method.

Formulating a general mathematical model for real-time control systems is more challenging since a different control application requires a different control system. For this reason, we use the framework of pursuit evasion games to model real-time situation understanding and control problems. The pursuit evasion game can be viewed as the worst-case control problem since the pursuers must find the best controls to pursue evaders while the equally capable evaders try to avoid the pursuers with their best efforts. One can argue that the most control problems are easier than pursuit evasion games. Hence, if one can solve the pursuit evasion games, she can also solve the other control problems.

Since each sensor node or agent communicates with one another using the wireless communication channel, one must be able to quantify and evaluate the effect of communication loss in estimation and control. This problem is treated in the last chapter of this thesis.

1.3 Overview of the Thesis

Chapter 2 describes a distributed tracking algorithm for sensor networks. We assume a network of sparsely located binary sensors and the tracking problem is formulated as a hidden state estimation problem in a hidden Markov model (HMM) over the finite state space of sensors. An optimal distributed tracking algorithm is derived using the Viterbi algorithm. Then we show provably good pruning strategies for scalability and the conditions under which the algorithm is robust against false detections. Since the computation and storage of track information are done in a completely distributed manner, the method is robust against node failures and transmission failures. However, we find that the formulation can not be easily extended to multi-target tracking problem. This chapter gives a motivation for developing a general framework for multi-target tracking problems discussed in Chapter 3. This chapter is based on [Oh and Sastry, 2005b].

In Chapter 3, we develop a general Bayesian framework for multi-target tracking problems. This framework handles uncertainties in identity, number of targets, track initiation and termination times. As discussed earlier, the mathematical framework of multi-target tracking is well-suited for inference problems in sensor networks. Based on this framework, Chapter 4 develops an efficient algorithm for solving multi-target tracking problems.

Chapter 4 describes the Markov chain Monte Carlo data association (MCMCDA) method for solving the data association problems appearing in multi-target tracking problems. When the number of targets is fixed, the single-scan version of MCMCDA approximates joint probabilistic data association (JPDA). Although the exact computation of association probabilities in JPDA is NP-hard, we prove that the single-scan MCMCDA algorithm provides a fully polynomial randomized approximation scheme for JPDA. For general multi-target tracking problems, in which unknown numbers of targets appear and disappear at random times, we present a multi-scan MCMCDA algorithm that approximates the optimal Bayesian filter based on the general framework developed in Chapter 3. We also present extensive simulation studies supporting the theoretical results and performance of MCMCDA. MCMCDA is used as a building block of the surveillance system described in Chapter 5 and Chapter 6. This chapter is based on [Oh *et al.*, 2006b].

Chapter 5 considers the problem of tracking multiple targets and managing their identities in sensor networks and proposes a scalable distributed multi-target tracking and identity management (DMTIM) algorithm that can track an unknown number of targets and manage their identities efficiently in a distributed sensor network environment. A decision based on a single set of tracks may be risky since tracks do not fully exhibit the uncertainty in the identities of targets accumulated from continuous interactions among crossing or nearby targets. DMTIM compactly manages identities of targets based on the *identity-mass-flow* framework. This framework prevents exponential growth in computation and storage of target-track association probabilities. Using identity and track fusion, DMTIM maintains consistent identities and tracks among neighboring sensors. This chapter is based on [Oh *et al.*, 2006a].

In Chapter 6, the multi-target tracking algorithm developed in Chapter 4 is applied to a real-time control system using sensor networks. In particular, we consider the problem of pursuit evasion games (PEGs). The main challenge in developing a real-time control system using sensor networks is the inconsistency in sensor measurements due to packet loss, communication delay, and false detections. We address this challenge by developing a real-time hierarchical control system, named *LochNess*, which decouples the estimation of evader states from the control of pursuers via multiple layers of data fusion. The control system *LochNess* is evaluated in simulation and successfully demonstrated using a large-scale outdoor sensor network deployment. To the author's best knowledge, it is the first demonstration of multi-target tracking using a sensor network without relying on classification. This chapter is based on [Oh *et al.*, 2007].

Chapter 7 describes a distributed networked control system (DNCS) consisting of multiple agents communicating over a lossy communication channel, *e.g.*, wireless channel. First, we discuss the state estimation of a DNCS and propose approximate filtering algorithms. While the time complexity of the exact method can be exponential in the number of possible communication link configurations, the time complexity of an approximate method is not dependent on the number of possible configurations. We also derive a condition under which the stable state estimation is not possible for the general case with an arbitrary number of lossy communication links. Lastly, the problem of finding a communication control which stabilizes a DNCS is considered. The stabilizing communication control problem seeks the acceptable ranges of packet loss rates at which the overall system is stable. Efficient algorithms based on convex optimization are developed for solving the stabilizing communication control problem. This chapter is partially based on [Oh and Sastry, 2006].

CHAPTER 2

DISTRIBUTED TRACKING IN SENSOR NETWORKS

In this chapter, we take the minimalist approach and use the binary sensor model, in which each sensor reports only a binary value indicating whether an object is present near the sensor or not. The tracking problem in sensor networks is formulated as a hidden state estimation problem in a hidden Markov model (HMM) over the finite state space of sensors. Then an optimal distributed tracking algorithm is derived using the Viterbi algorithm [Rabiner, 1989]. We then show provably good pruning strategies for scalability and the conditions under which the algorithm is robust against false detections. The algorithm is also extended to handle non-disjoint sensing regions and to track multiple moving objects. Since the computation and storage of track information are done in a completely distributed manner, the method is robust against node failures and transmission failures. In addition, the use of binary sensors makes the proposed algorithm suitable for many sensor network applications such as location-based services. The algorithm presented in this chapter is well-suitable for indoor tracking problems where the movement of an object cannot be easily modeled and sensor network self-localization is difficult.

Based on the fomulation developed in this chapter, we discuss issues with a completely distributed algorithm for a general multi-target tracking problem in a sensor network. The general multi-target tracking problem is rigorously fomulated in Chapter 3, an algorithm for solving general multi-target tracking problems is given in Chapter 4, and a real-time tracking system for sensor networks is described in Chapter 6.

2.1 Problem Formulation

Suppose there are N sensor nodes, $s_1, \dots, s_N$. We denote the sensing region of sensor node s_i by C_i and assume that the sensing region of each sensor node is disjoint from the sensing regions of other sensor nodes. A sensor node s_i is *passage connected* to a sensor node s_j if there is a path from the sensing region of s_i to the sensing region of s_j such that an object can traverse. A sensor node s_i is *direct passage connected* to a sensor node s_j if s_i is passage connected to s_j and there is a path from s_i to s_j which does not intersect sensing regions of the remaining sensors. We assume that the passage connectivity (direct passage connectivity) is symmetric, *i.e.*, if s_i is passage connected (direct passage connected) to s_j, then s_j is passage connected (direct passage connected) to s_i. Note that when there is no confusion, we will denote sensor s_i by its index i.

Definition 1. *A* passage connectivity graph *of sensor nodes $s_1, \dots, s_N$ is a graph $G = (V, E)$, where $V = \{1, \dots, N\}$ and $(u, v) \in E$ if and only if u and v are direct passage connected.*

Let $G = (V, E)$ be the passage connectivity graph of sensor nodes $s_1, \dots, s_N$. An example of a passage connectivity graph is shown in Figure 2.1. Notice that since $u \in V$ is direct passage connected to u, $(u, u) \in E$. Throughout this chapter, without loss of generality, we assume that G is a connected graph, since, if G is disconnected, we can consider each partition of G separately. Let $X_t \in \{1, \dots, N\}$ be the location of an object

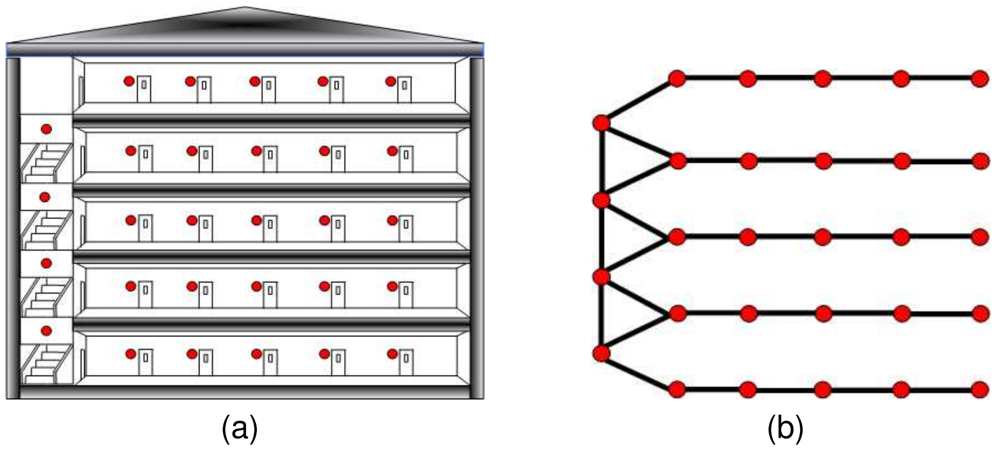


Figure 2.1: (a) An example of indoor sensor network deployment (solid circles are sensors). (b) The passage connectivity graph of the sensor network shown in (a).

on G at time t for $t \in \{1, \ldots, T\}$, where T is the time horizon. An object appears at $i \in V$ with the initial state probability $\pi_i = P(X_1 = i)$. If $X_t = i$, an object is located in C_i at time t.

Let $\mathrm{NB}_i = \{j \in V : (i, j) \in E\}$ be the neighborhood of i. We assume that the evolution of an object on G is Markovian such that $P(X_{t+1} = j | X_t = i, X_{t-1}, \ldots, X_1) = P(X_{t+1} = j | X_t = i) := p_{ij}$ for all t. Let $P_{\mathrm{t}} = [p_{ij}]$ be the transition probability matrix. For each i, $\sum_{j \in \mathrm{NB}_i} p_{ij} = 1$ and $p_{ij} = 0$ for $j \notin \mathrm{NB}_i$. Notice that our framework can handle more complex dynamics of a moving object by using a k-th order Markov chain, for $k \geq 2$. For each sensor node s_i, if an object is in C_i, it is detected with probability η_i. Let $Y_t^i \in \{0, 1\}$ be the observation made by s_i at time t. When an object is in C_i at time t, the object is detected, $Y_t^i = 1$, with probability η_i, and the object is not detected, $Y_t^i = 0$, with probability $1 - \eta_i$. For now we assume there are no false detections. We show the robustness of our algorithm in the presence of false detections in Section 2.4. The model parameters, π, P_{t}, and η, can be learned from historic data using Baum-Welch method [Rabiner, 1989], hence, it is safe to assume that they are known in advance. We also assume that node u can communicate with node v reliably in timely manner if $(u, v) \in E$.

Since the sensing regions are disjoint, we can assume that each sensor makes an independent observation at each time. Let $Y_t = (Y_t^1, \ldots, Y_t^N)^T$, $Y_{1:T} = \{Y_1, \ldots, Y_T\}$ and $X_{1:T} = \{X_1, \ldots, X_T\}$. The joint distribution is

$$P(X_{1:T}, Y_{1:T}) = P(X_1) \prod_{t=2}^{T} P(X_t|X_{t-1}) \prod_{t=1}^{T} P(Y_t|X_t), \tag{2.1}$$

where

$$P(Y_t|X_t) = (\eta_{X_t})^{Y_t^{X_t}} (1 - \eta_{X_t})^{1-Y_t^{X_t}}. \tag{2.2}$$

Now the tracking problem on a graph is to find the most probable trajectory of a moving object on the graph given observations $y_{1:T}$, *i.e.*, finding the maximum a posteriori (MAP) solution $x^*_{1:T}$:

$$\begin{aligned} x^*_{1:T} &= \arg\max_{x_{1:T}} P(x_{1:T}|y_{1:T}) \\ &= \arg\max_{x_{1:T}} \frac{P(y_{1:T}|x_{1:T})P(x_{1:T})}{P(y_{1:T})} \\ &= \arg\max_{x_{1:T}} P(y_{1:T}|x_{1:T})P(x_{1:T}) \\ &= \arg\max_{x_{1:T}} P(x_{1:T}, y_{1:T}). \end{aligned} \tag{2.3}$$

The Bayes rule is used for the second equality and the normalization constant $P(y_{1:T})$ is dropped in the third equality since it does not change the maximization problem.

2.2 Optimal Distributed Tracking Algorithm

The most probable sequence of hidden states $x^*_{1:T}$ given observations $y_{1:T}$ can be efficiently found by the Viterbi algorithm [Rabiner, 1989]. We first define

$$\delta_t(i) = \max_{x_1,\ldots,x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t = i, y_{1:t}), \tag{2.4}$$

for $i = 1, \ldots, N$ and $t = 1, \ldots, T$. $\delta_t(i)$ is the maximum probability of a single path ending in state i at time t, given observations $y_{1;t}$. Once we have all $\delta_T(i)$, we can trace backward from the state which maximizes $\delta_T(i)$ to find the sequence $x^*_{1:T}$. For more detail about the Viterbi algorithm, see [Rabiner, 1989].

We compactly denote the event $\{x_t = i\}$ by z_t^i. Notice that

$$\begin{aligned}
\delta_t(i) &= \max_{x_{1:t-1}} P(x_{1:t-1}, z_t^i, y_{1:t}) \\
&= \max_{x_{1:t-1}} P(z_t^i|x_{t-1})P(y_t|z_t^i)P(x_{1:t-1}, y_{1:t-1}) \\
&= \left[\max_{x_{1:t-1}} P(z_t^i|x_{t-1})P(x_{1:t-1}, y_{1:t-1})\right] P(y_t|z_t^i) \\
&= \left[\max_{j\in\{1,\ldots,N\}} P(z_t^i|z_{t-1}^j) \max_{x_{1:t-2}} P(x_{1:t-2}, z_{t-1}^j, y_{1:t-1})\right] P(y_t|z_t^i) \\
&= \left[\max_{j\in \mathrm{NB}_i} P(z_t^i|z_{t-1}^j)\delta_{t-1}(j)\right] P(y_t|z_t^i),
\end{aligned} \tag{2.5}$$

where the domain of the maximization is reduced from $\{1, \ldots, N\}$ to NB_i in the last equality since $P(z_t^i|z_{t-1}^j) = 0$ for $j \notin \mathrm{NB}_i$ ($i \notin \mathrm{NB}_j$ by symmetry). Hence, in order to compute $\delta_t(i)$, we only need $\{\delta_{t-1}(j) : j \in \mathrm{NB}_i\}$ from the neighbors of s_i, transition probabilities and the likelihood $P(y_t|z_t^i)$.

Since $P(y_t^i = 1|z_t^i) = \eta_i$ and $P(y_t^i = 0|z_t^i) = 1 - \eta_i$, (2.5) can be further simplified as

$$\delta_t(i) = \left[\max_{j\in \mathrm{NB}_i} P(z_t^i|z_{t-1}^j)\delta_{t-1}(j)\right] \eta_i^{y_t^i}(1-\eta_i)^{1-y_t^i}. \tag{2.6}$$

Thus, we can compute $\delta_t(i)$ with only local information. The distributed implementation to find the most probable trajectory of a moving object is composed of two steps: distributed tracking and distributed backtracking.

Distributed tracking is described in Algorithm 1 and distributed backtracking is described in Algorithm 2. In Algorithm 1, $\psi_t(i)$ records the best previous location of an object to be used by Algorithm 2. The tracking step of the algorithm is graphically il-

Algorithm 1 Distributed Tracking (node i at time t)

```
1: if $t = 1$ then
2:     $\delta_1(i) = \pi_i \eta_i^{y_1^i} (1 - \eta_i)^{1-y_1^i}$
3:     $\psi_1(i) = 0$
4:     transmit $\delta_1(i)$ to $\text{NB}_i$
5: else if $t > 1$ then
6:     receive $\{\delta_{t-1}(j) : j \in \text{NB}_i\}$
7:     $\delta_t(i) = \left[\max_{j \in \text{NB}_i} P(z_t^i | z_{t-1}^j) \delta_{t-1}(j)\right] \eta_i^{y_t^i} (1 - \eta_i)^{1-y_t^i}$
8:     $\psi_t(i) = \arg\max_{j \in \text{NB}_i} P(z_t^i | z_{t-1}^j) \delta_{t-1}(j)$
9:     transmit $\delta_t(i)$ to $\text{NB}_i$
10: end if
```

Algorithm 2 Distributed Backtracking (node i at time t)

```
1: if $t = T$ then
2:     $\phi_T(i) = \delta_T(i)$
3:     transmit $(\phi_T(i), x_T^*(i) = \{i\})$ to $\psi_T(i)$
4: else if $1 \leq t < T$ then
5:     receive $D_i = \{(\phi_{t+1}(j), x_{t+1:T}^*(j)) : j \in J_i\}$, where $J_i = \{j \in \text{NB}_i : \psi_{t+1}(j) = i\}$
6:     if $D_i \neq \emptyset$ then
7:         $j = \arg\max_{k \in D_i} \phi_{t+1}(k)$
8:         $\phi_t(i) = \phi_{t+1}(j)$
9:         $x_{t+1:T}^*(i) = x_{t+1:T}^*(j)$
10:        $x_t^*(i) = i$
11:        if $t > 1$ then
12:            transmit $(\phi_t(i), x_{t:T}^*(i))$ to $\psi_t(i)$
13:        else if $t = 1$ then
14:            transmit $x_{1:T}^*(i)$ to $s_\text{b}$
15:        end if
16:    end if
17: end if
```

lustrated in Figure 2.2. In Figure 2.2(a), node i receives δ_{t-1} from its neighbors. After computing $\delta_t(i)$ and $\psi_t(i)$, node i transmits $\delta_t(i)$ to its neighbors (Figure 2.2(b)).

For distributed backtracking, we suppose that the sensor node s_b requests a trajectory of the moving object at time T. Since trajectory information $\psi_t(i)$ is stored in a distributed manner, we backtrack to recover the most probable trajectory by collecting vertices along

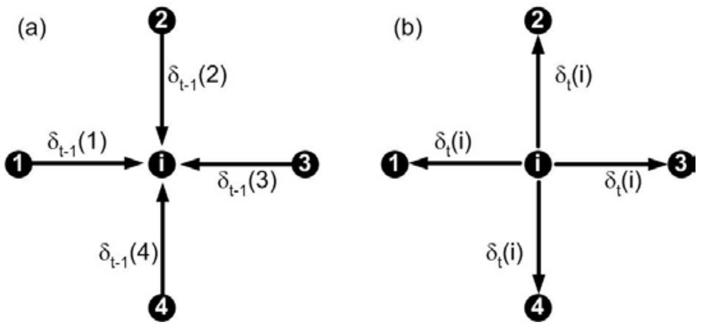


Figure 2.2: A graphical illustration of tracking. (a) Node i receives δ_{t-1} from its neighbors (line 6 of Algorithm 1). (b) Node i transmits $\delta_t(i)$ to its neighbors (line 4 and 9 of Algorithm 1).

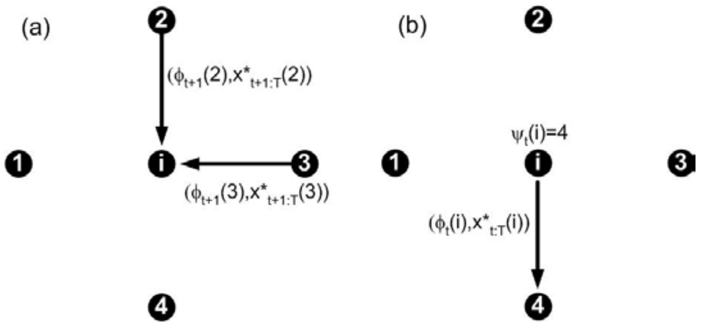


Figure 2.3: A graphical illustration of backtracking. (a) Node i receives $(\phi_{t+1}, x^*_{t+1:T})$ from its neighbors (line 5 of Algorithm 2). (b) Node i transmits $(\phi_t(i), x^*_{t:T}(i))$ to $\psi_t(i)$ (line 3 and 12 of Algorithm 2).

the path directed by $\psi_t(i)$ as described in Algorithm 2. In Algorithm 2, the trajectory $x^*_{t:T}(i)$ is simply concatenated at each step and a complete trajectory is available when the backtracking is terminated. However, an alternative approach is to transmit x^*_t to s_b at each t and transmit only $\phi_t(j)$ to $\psi_t(i)$ so that the transmission packets are of the same size. The backtracking step of the algorithm is graphically illustrated in Figure 2.3. In Figure 2.3(a), node i receives $(\phi_{t+1}, x^*_{t+1:T})$ from its neighbors. After line 10 of Algorithm 2, node i transmits $(\phi_t(i), x^*_{t:T}(i))$ to $\psi_t(i)$ (Figure 2.3(b)).

Notice that in order to carry out calculations described in Algorithm 1 and Algorithm 2, node i needs to know only π_i, η_i, and P_{t} restricted to NB_i and the remaining variables are computed on the fly. Hence, the calculations required for tracking and backtracking can be done in a completely distributed manner. The algorithm implements the Viterbi algorithm [Rabiner, 1989] and finds the most probable trajectory $x^*_{1:T}$. Since the computation and storage of track information are done in a completely distributed manner, the method is robust against node failures and transmission failures. If d is the maximum degree of the graph G, the number of transmissions at each time is bounded above by dN and does not depend on T. But the memory required to keep track information increases as T increases. For a large graph, the number of sensors involved in tracking increases as T increases. Hence, in order to bound the memory requirement and the number of sensors involved in tracking and make the algorithm scalable, we need a method to prune away unlikely tracks. For this purpose, we present provably good pruning strategies in Section 2.3.

2.3 Pruning

The main idea behind the pruning strategies discussed in this section is to prune a trajectory hypothesis if the number of detections is low. Since pruning is performed, based solely on local information, it is important to analyze the performance loss of the algorithm for the chosen pruning strategy. We first present the updated tracking and backtracking steps and then describe pruning strategies such that, for given $\epsilon > 0$, the algorithm finds an optimal trajectory $x^*_{1:T}$ with probability at least $1 - \epsilon$. An effect of pruning is shown in Figure 2.4 where the pruning strategy given in Theorem 1 is used.

Distributed tracking and backtracking with pruning are described in Algorithm 3 and Algorithm 4, respectively. They are modifications of Algorithm 1 and Algorithm 2. In Algorithm 3, a variable $\varphi_t(i)$ is introduced to keep the number of detections along the path directed by $\psi_t(i)$. A track hypothesis is pruned away if $g(t, \varphi_t(i)) = 1$ and the choice of $g(t, \varphi_t(i))$ is discussed below. $\delta_t(i)$ and $\psi_t(i)$ are all initialized to zeros.

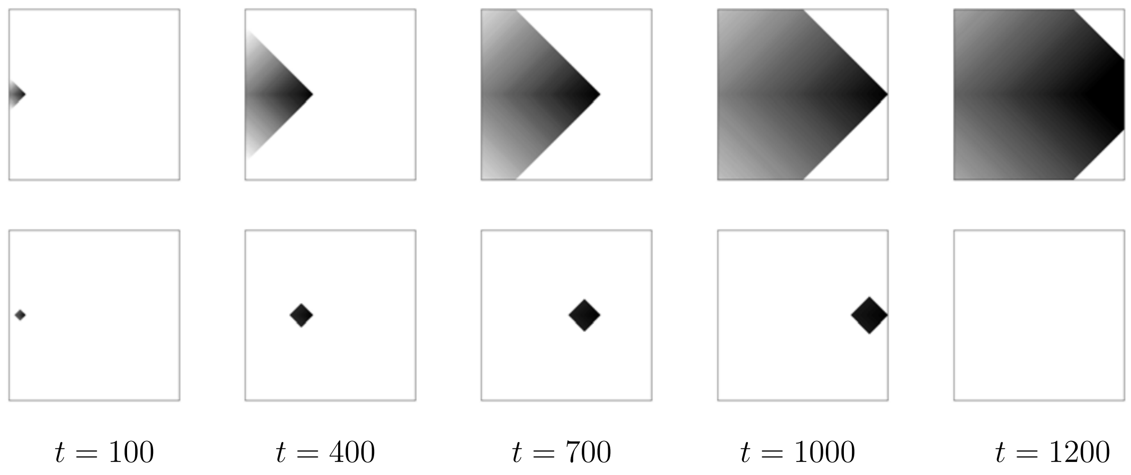


Figure 2.4: An effect of pruning: an object moves horizontally along the center row of 1000×1000 sensor grid ($\eta = .7$). A sensor node involved in tracking is painted with darker color. (Top) Without pruning. (Bottom) With pruning ($t_0 = 100$, $s = 100$ and $\epsilon = .05$).

We describe two pruning strategies such that, for given $\epsilon > 0$, the algorithm finds an optimal trajectory $x^*_{1:T}$ with probability at least $1 - \epsilon$. Let W_t be the event that the algorithm terminated at time t returns an incorrect solution, *i.e.*, when a correct solution is pruned away.

Theorem 1 (Finite Horizon)**.** *Let G be a passage connectivity graph of sensor nodes $s_1, \dots, s_N$. Suppose that $\eta_i > 0$, for all $i = 1, \dots, N$, and let $\eta = \min \eta_i$. For $\epsilon > 0$ and $T > 0$, choose $s \in \mathbb{N}$, such that $\frac{T}{\epsilon} \exp\left(-\frac{\eta t_0}{2}\right) < s \le T$ for some $t_0 \ge 1$, and choose $\theta_t \le \eta t \left(1 - \sqrt{\frac{2}{\eta t} \log\left(\frac{T}{s\epsilon}\right)}\right)$. If the following pruning strategy,*

$$g(t, \varphi_t(i)) = \begin{cases} 1 & \text{if } \varphi_t(i) < \theta_t \text{ for } t = ks \ge t_0, k \in \mathbb{N} \\ 0 & \text{otherwise}, \end{cases} \tag{2.7}$$

is used, then $P(W_T) < \epsilon$.

Proof: See Appendix A.1. ■

Algorithm 3 Distributed Tracking with Pruning (node i at time t)

```
 1: if $t = 1$ then
 2:     $\delta_1(i) = \pi_i \eta_i^{y_1^i}(1-\eta_i)^{1-y_1^i}$
 3:     $\psi_1(i) = 0$
 4:     $\varphi_1(i) = y_1^i$
 5:     transmit $(\delta_1(i), \varphi_1(i))$ to NB$_i$
 6: else if $t > 1$ then
 7:     receive $D_i = \{\delta_{t-1}(j), \varphi_{t-1}(j) : j \in \text{NB}_i\}$
 8:     if $D_i \neq \emptyset$ then
 9:         $\delta_t(i) = \left[\max_{j\in\text{NB}_i} P(z_t^i|z_{t-1}^j)\delta_{t-1}(j)\right] \eta_i^{y_t^i}(1-\eta_i)^{1-y_t^i}$
10:         $\psi_t(i) = \arg\max_{j\in\text{NB}_i} P(z_t^i|z_{t-1}^j)\delta_{t-1}(j)$
11:         $\varphi_t(i) = \varphi_{t-1}(\psi_t(i)) + y_t^i$
12:         if $g(t, \varphi_t(i)) = 0$ then
13:             transmit $(\delta_t(i), \varphi_t(i))$ to NB$_i$
14:         end if
15:     end if
16: end if
```

The following theorem shows an existence of a pruning strategy when $T \to \infty$.

Theorem 2 (Infinite Horizon)**.** *Let G be a passage connectivity graph of sensor nodes $s_1, \ldots, s_N$. Suppose that $\eta_i > 0$, for all $i = 1, \ldots, N$, and let $\eta = \min \eta_i$. For $\epsilon > 0$, choose $s \in \mathbb{N}$, such that $s > \frac{2}{\eta}\log\left(\frac{1+\epsilon}{\epsilon}\right)$, and choose $\theta_t \leq \eta t \left(1 - \sqrt{\frac{2}{\eta s}\log\left(\frac{1+\epsilon}{\epsilon}\right)}\right)$. If the following pruning strategy*

$$g(t, \varphi_t(i)) = \begin{cases} 1 & \textit{if } \varphi_t(i) < \theta_t \textit{ for } t = ks \geq s, k \in \mathbb{N} \\ 0 & \textit{otherwise}, \end{cases} \tag{2.8}$$

is used, then $P(W_\infty) < \epsilon$.

Proof: See Appendix A.2. ■

Algorithm 4 Distributed Backtracking with Pruning (node i at time t)

```
1: **if** $t = T$ **then**
2:     $\phi_T(i) = \delta_T(i)$
3:     **if** $\phi_T(i) > 0$ **then**
4:         transmit $(\phi_T(i), x^*_T(i) = \{i\})$ to $\psi_T(i)$
5:     **end if**
6: **else if** $1 \leq t < T$ **then**
7:     receive $D_i = \{(\phi_{t+1}(j), x^*_{t+1:T}(j)) : j \in J_i\}$, where $J_i = \{j \in \text{NB}_i : \psi_{t+1}(j) = i\}$
8:     **if** $D_i \neq \emptyset$ **then**
9:         $j = \arg\max_{k \in D_i} \phi_{t+1}(k)$
10:        $\phi_t(i) = \phi_{t+1}(j)$
11:        $x^*_{t+1:T}(i) = x^*_{t+1:T}(j)$
12:        $x^*_t(i) = i$
13:        **if** $t > 1$ **then**
14:            transmit $(\phi_t(i), x^*_{t:T}(i))$ to $\psi_t(i)$
15:        **else if** $t = 1$ **then**
16:            transmit $x^*_{1:T}(i)$ to $s_\text{b}$
17:        **end if**
18:    **end if**
19: **end if**
```

2.4 Robustness

In previous sections, we have assumed that there are no false detections in our observation model. In this section, we study the robustness of the algorithm in the presence of false detections. We assume that the algorithm is run without pruning so if a pruning strategy is used, unless stated otherwise, the arguments here should be understood in a probabilistic sense, *i.e.*, given $\epsilon > 0$ and a pruning strategy given in Section 2.3, the arguments hold with probability larger than $1-\epsilon$. Let ξ_i be the false detection probability for the sensor s_i. Thus, when an object is not in C_i at time t, the sensor s_i makes a false detection, $Y^i_t = 1$, with probability ξ_i and $Y^i_t = 0$ with probability $1 - \xi_i$. We need to update the joint distribution

(2.1) as following to allow false detections.

$$P(X_{1:T}, Y_{1:T}) = P(X_1) \prod_{t=2}^{T} P(X_t|X_{t-1}) \prod_{t=1}^{T} P(Y_t|X_t), \tag{2.9}$$

where

$$\begin{aligned} P(Y_t|X_t) &= \prod_{i=1}^{N} P(Y_t^i|X_t) \\ &= (\eta_{X_t})^{Y_t^{X_t}} (1-\eta_{X_t})^{1-Y_t^{X_t}} \prod_{j \neq X_t} (\xi_j)^{Y_t^j} (1-\xi_j)^{1-Y_t^j}. \end{aligned} \tag{2.10}$$

Unfortunately, we can no longer compute the likelihood $P(Y_t|X_t)$ with only local information as shown in (2.2). We need observations from all sensors in order to compute the likelihood and it is difficult to guarantee the optimality of any distributed algorithm. We focus on the case, in which the detection probabilities and the false detection probabilities are uniform, for its simplicity and show the optimality conditions for the algorithm given in Section 2.2 in the presence of false detections.

Theorem 3. *Let G be a passage connectivity graph of sensor nodes $s_1, \dots, s_N$. Suppose that $\eta = \eta_i \geq .5$ and $\xi = \xi_i \leq .5$, for all $i = 1, \dots, N$ and $m = \min P(x_{1:T}) > 0$. Let $q_{1:T}$ be the solution computed by the algorithm given in Section 2.2 and let $x^*_{1:T}$ be the optimal solution, an MAP estimate to (2.9) given $y_{1:T}$. Then the following inequalities hold*

$$P(q_{1:T}, y_{1:T}) \leq P(x^*_{1:T}, y_{1:T}) \leq \alpha P(q_{1:T}, y_{1:T}), \tag{2.11}$$

where $\alpha = \left(\frac{1-\xi}{\xi}\right)^{\frac{\log(P(q_{1:T})/m)}{\log(\eta/(1-\eta))}}$.

Proof: See Appendix A.3. ■

In Theorem 3, $\alpha \geq 1$ since $\xi \leq .5$. To get a tighter bound, we want α to be as small as possible. Notice that when $P(q_{1:T})$ is small, α is small and we can be confident that the estimate $q_{1:T}$ from the algorithm given in Section 2.2 is close to the MAP estimate $x^*_{1:T}$. An extreme case is when $\xi = .5$. Then $\alpha = 1$ and $q_{1:T} = x^*_{1:T}$. However, (2.11) is the worst-case bound. To increase the performance of the algorithm for an average case, we require small ξ. Hence, we assume that ξ is fixed at some small value and discuss approaches to make the worst-case bound tight.

Corollary 1. *Let G be a passage connectivity graph of sensor nodes $s_1, \dots, s_N$. Suppose that G is a regular graph with a uniform initial state distribution and uniform transition probabilities. Then, under the conditions specified in Theorem 3, $q_{1:T} = x^*_{1:T}$.*

Proof: G is a regular graph with a uniform initial state distribution and uniform transition probabilities so we have a uniform prior. In particular, $P(q_{1:T}) = P(x^*_{1:T})$. Hence, $\alpha = 1$ and $q_{1:T} = x^*_{1:T}$. ■

Corollary 2. *Assume the conditions specified in Theorem 3. For $\epsilon_1 > 0$, let*

$$c = \exp\left(\log r \log\left(\frac{1-\xi}{\xi}\right) / \log(1+\epsilon_1)\right), \tag{2.12}$$

where $r = \max P(x_{1:T}) / \min P(x_{1:T})$. If $\eta \geq \frac{c}{1+c}$, then $P(q_{1:T}, y_{1:T}) \leq P(x^*_{1:T}, y_{1:T}) \leq (1+\epsilon_1)P(q_{1:T}, y_{1:T})$.

Proof: See Appendix A.4. ■

Hence, an ideal case is when r is small and η is big (as specified in Corollary 2). For example, when $r = 1000$ and $\xi = .45$, it requires $\eta \geq .8808$ for $\epsilon_1 = 1$ and $\eta \geq .7793$ for $\epsilon_1 = 2$. But if $r = 1000$ and $\xi = .3$, we need $\eta \geq .9998$ for $\epsilon_1 = 1$ and $\eta \geq .9952$ for $\epsilon_1 = 2$. We note that, for binary sensors, the detection probability can be boosted to a desired value by deploying multiple sensors over the same surveillance region.

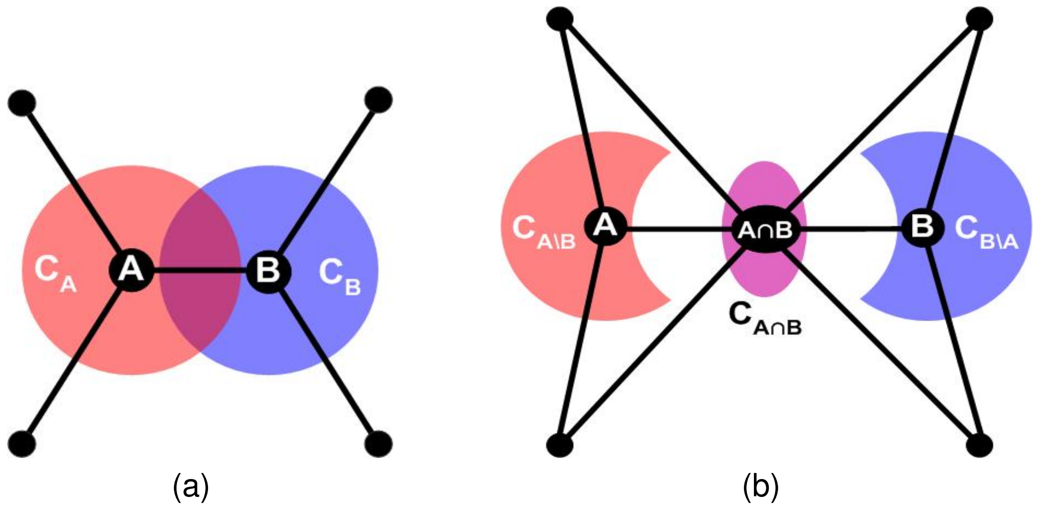


Figure 2.5: (a) An example of non-disjoint sensing regions. Sensor nodes are shown in dots and disks around sensors represent sensing regions. (b) A partition of sensing regions from the example shown in (a).

2.5 Non-disjoint Sensing Regions

Until now, we have assumed that the sensing regions of sensors are disjoint, leading to independent observations. However, this assumption may not be satisfied in all cases. In some cases, we may want to introduce more sensors to improve the detection probability. In this section, we describe how to handle non-disjoint sensing regions.

Suppose that G is a passage connectivity graph. For the simplicity of exposition, we consider the case when the sensing regions of two sensors A and B are not disjoint (see Figure 2.5(a)). However, the method can be applied to the case with more than two sensors. Consider all disjoint segments of $C_A \cup C_B$, *i.e.*, $C_{A \setminus B} = C_A \setminus C_B$, $C_{B \setminus A} = C_B \setminus C_A$ and $C_{A \cap B} = C_A \cap C_B$ (see Figure 2.5(b)). Recall that C_A is the sensing region of sensor A. In this example, an extra vertex $A \cap B$ is added into G along with new edges from $A \cap B$ (see Figure 2.5(b)). The sensor nodes A and B exchange observations and the necessary computations at $A \cap B$ can be distributed between A and B. The additional parameters of this updated graph can be learned from historic data using Baum-Welch method as mentioned earlier. However, if the sensor model of each sensor node is known, we can compute the sensor models of each disjoint segment. Let η_A and η_B be the detection probabilities of sensor A and sensor B, respectively, and let ξ_A and ξ_B be the false detection probabilities of sensor A and sensor B, respectively. Let Y_A and Y_B be the observations made by sensor A and sensor B, respectively, and let X be the true position of the object. Then the sensor models of disjoint segments can be computed as shown in Table 2.1. Now tracking can be done as discussed in Section 2.2 and 2.3.

Table 2.1: Sensor model $P(Y_A, Y_B|X)$ of disjoint segments in Figure 2.5(b)

	$Y_A = 0, Y_B = 0$	$Y_A = 0, Y_B = 1$	$Y_A = 1, Y_B = 0$	$Y_A = 1, Y_B = 1$
$X \in C_{A\setminus B}$	$(1-\eta_A)(1-\xi_B)$	$(1-\eta_A)\xi_B$	$\eta_A(1-\xi_B)$	$\eta_A\xi_B$
$X \in C_{B\setminus A}$	$(1-\xi_A)(1-\eta_B)$	$(1-\xi_A)\eta_B$	$\xi_A(1-\eta_B)$	$\xi_A\eta_B$
$X \in C_{A\cap B}$	$(1-\eta_A)(1-\eta_B)$	$(1-\eta_A)\eta_B$	$\eta_A(1-\eta_B)$	$\eta_A\eta_B$

2.6 Simulation Results on Multiple Object Tracking

Suppose there are K independently moving objects and assume that each sensor can identify one object from another. Let Y_t^i be an observation made by sensor s_i at time t, which takes a value from a power set of $\{1,\ldots,K\}$. Then we can express the sensor model as, for all $k = 1,\ldots,K$ and $i = 1,\ldots,N$,

$$\begin{aligned} P(Y_t^i \ni k | X_t^k = i) &= \eta_i^k \\ P(Y_t^i \not\ni k | X_t^k = i) &= 1 - \eta_i^k, \end{aligned} \tag{2.13}$$

where $X_t^k \in \{1,\ldots,N\}$ is the location of the object k at time t and η_i^k is the detection probability at sensor i for object k. Then we can individually track each object using the algorithm given in Section 2.2 and 2.3.

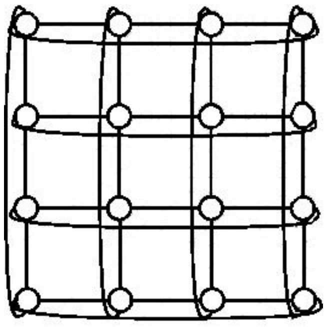


Figure 2.6: A toroidal grid

For simulation, we consider a toroidal grid passage connectivity graph G. An example of a toroidal grid is shown in Figure 2.6. We assume there are two objects and they all have the uniform initial state distribution while $p_{ii} = .1$ and $p_{ij} = .9/|\text{NB}_i \setminus \{i\}|$ for $j \in \text{NB}_i \setminus \{i\}$. The detection probabilities are randomly chosen from $[.7, 1)$. However, we consider the situation described in [Shin *et al.*, 2003; Liu *et al.*, 2004], in which a sensor does not correctly classify an object when multiple objects are present in the same sensing region. So when multiple objects are present in the same sensing region, we assume that the corresponding sensor does not report detection. We approximate this situation by the sensing model (2.13). Two scenarios are considered and they are shown in Figure 2.7(a) and 2.7(c). The estimated tracks are shown in Figure 2.7(b) and 2.7(d). The estimated tracks are the same as the true trajectories when objects are detected while the estimated tracks follow the path with maximum transition probabilities and minimum detection probabilities when objects are not detected.

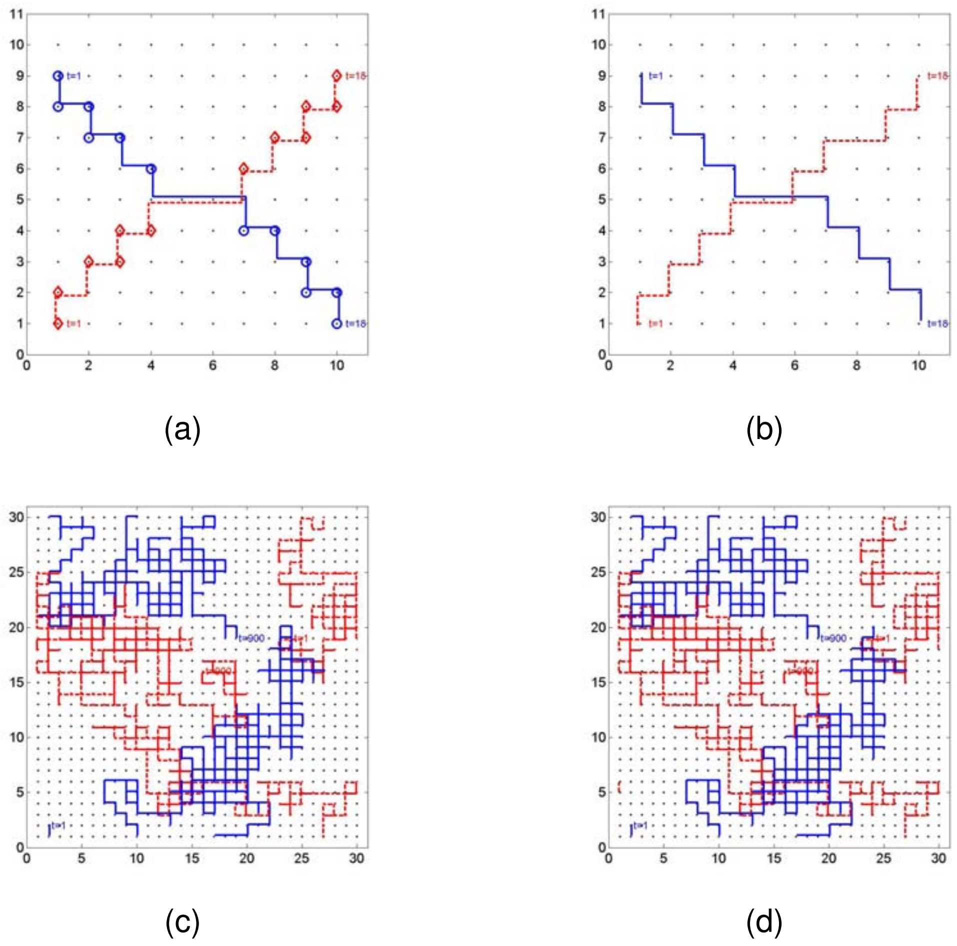


Figure 2.7: (a) Scenario 1 (10×10 toroidal grid, trajectory of object 1 in solid line, trajectory of object 2 in dashed line, observations from object 1 in circles, observations from object 2 in diamonds). (b) Estimated tracks for scenario 1 (trajectory of object 1 in solid line, trajectory of object 2 in dashed line). (c) Scenario 2 (30×30 toroidal grid). (d) Estimated tracks for scenario 2.

2.7 Issues with Tracking Multiple Objects in Sensor Networks

As described in Section 2.4, in the presence of false alarms, the evaluation of the likelihood (2.10) requires observations from all sensors. Hence, it is difficult to compute the exact likelihood using only local information and there is no efficient distributed algorithm for the exact likelihood computation. However, Section 2.4 shows that we can still find a good estimate in some cases using only local information. In fact, this idea is extended in Chapter 6 to approximately eliminate false detections when computing the likelihood of binary detections. In summary, although we cannot efficiently compute the exact likelihood using only local information, there are methods to minimize the effect of false alarms.

Now, let us consider the problem of tracking multiple objects. Although we show a simulation study of tracking multiple objects in the previous section, we need an ability to classify or identify an object. In general, this is an extremely difficult task and there is always a positive probability of misclassification even with a high-end sensing system. Hence, there is always uncertainty about the identity of an object. Because of this uncertainty, measurements are correlated and one cannot derive a multi-target tracking algorithm using only local information. Considering the possibility of false alarms and missing measurements and uncertainty about the appearance and disappearance times of an object, the problem gets even more complex. One can think of the multi-target tracking problem as a sequential hypothesis testing of exponentially increasing hypotheses with time. Measurements from neighboring sensors are not sufficient to initiate, maintain, disambiguate, and terminate tracks of multiple objects in the presence of clutter; it requires measurements from distant sensors. The method developed in this chapter cannot handle the general multi-target tracking problems and we need a general framework to develop a more robust tracking algorithm. This is the topic of Chapter 3.

2.8 Summary

In this chapter, we have described a novel tracking algorithm for sensor network. The tracking problem is formulated as a hidden state estimation problem in a hidden Markov model (HMM) over the finite state space of sensors. Then an optimal distributed tracking algorithm is derived using the Viterbi algorithm. The algorithm finds an optimal solution in a fully distributed manner. Furthermore, the algorithm is based on the simple binary sensors and does not depend on sensor network localization. We have described provably good pruning strategies for scalability of the algorithm and showed the conditions under which the algorithm is robust against false detections. We have also presented extensions of the algorithm to handle non-disjoint sensing regions and to track multiple objects. Since the computation and storage of track information are done in a completely distributed manner, the method is robust against node failures and transmission failures.

Chapter 3

General Multi-target Tracking Problems

Multi-target tracking plays an important role in many areas of engineering such as surveillance [Bar-Shalom and Fortmann, 1988], computer vision [Cox, 1993; Dellaert *et al.*, 2003], signal processing [Xie and Evans, 1991], network and computer security [Cybenko *et al.*, 2004], and sensor networks [Oh *et al.*, 2007]. In the standard setup, some indistinguishable targets move continuously in a given region, typically independently according to a known, Markovian process. Targets arise at random in space and time, persist for a random length of time, and then cease to exist; the sequence of states that a target follows during its lifetime is called a *track*. The positions of moving targets are measured, either at random intervals or, more typically, in periodic *scans* that measure the positions of all targets simultaneously. The position measurements are noisy and occur with detection probability less than one, and there is a noise background of spurious position reports, *i.e.*, false alarms or clutter.

The essence of the multi-target tracking problem is to find tracks from the noisy measurements. Now, if the sequence of measurements associated with each target is known, multi-target tracking (at least under the assumption of independent motion) reduces to a set of state estimation problems. Unfortunately, the association between measurements and targets is unknown. The *data association* problem is to work out which measurements were generated by which targets; more precisely, we require a partition of measurements such that each element of a partition is a collection of measurements generated by a single target or clutter [Sittler, 1964]. In the general case, uncertainty as to the correct association is unavoidable.

This chapter develops a general framework for multi-target tracking problems in order to handle uncertainties in identity, number of targets, track initiation and termination times. The framework developed in this chapter is used in Chapter 4 to develop an effient algorithm for solving multi-target tracking problems.

3.1 Problem Formulation

Let $T \in \mathbb{Z}^+$ be the duration of surveillance. Let K be the number of objects that appear in the surveillance region $\mathcal{R}$ during the surveillance period. Each object k moves in $\mathcal{R}$ for some duration $[t_\mathrm{i}^k, t_\mathrm{f}^k] \subseteq [1, T]$. Notice that the exact values of K and $\{t_\mathrm{i}^k, t_\mathrm{f}^k\}$ are unknown. Each object arises at a random position in $\mathcal{R}$ at t_i^k, moves independently around $\mathcal{R}$ until t_f^k and disappears. At each time, an existing target persists with probability $1 - p_\mathrm{z}$ and disappears with probability p_z. The number of objects arising at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $\lambda_\mathrm{b} V$ where λ_b is the birth rate of new objects per unit time, per unit volume, and V is the volume of $\mathcal{R}$. The initial position of a new object is uniformly distributed over $\mathcal{R}$.

Let $F^k : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be the discrete-time dynamics of the object k, where n_x is the dimension of the state variable, and let $x_t^k \in \mathbb{R}^{n_x}$ be the state of the object k at time t. The object k moves according to

$$x_{t+1}^k = F^k(x_t^k) + w_t^k, \qquad \text{for } t = t_{\mathrm{i}}^k, \ldots, t_{\mathrm{f}}^k - 1, \tag{3.1}$$

where $w_t^k \in \mathbb{R}^{n_x}$ are white noise processes. The white noise process is included to model non-rectilinear motions of targets. The noisy observation (or measurement[1]) of the state of the object is measured with a detection probability p_{d}. Notice that, with probability $1 - p_{\mathrm{d}}$, the object is not detected and we call this a missing observation. There are also false alarms and the number of false alarms has a Poisson distribution with a parameter $\lambda_{\mathrm{f}} V$ where λ_{f} is the false alarm rate per unit time, per unit volume. Let n_t be the number of observations at time t, including both noisy observations and false alarms. Let $y_t^j \in \mathbb{R}^{n_y}$ be the j^{th} observation at time t for $j = 1, \ldots, n_t$, where n_y is the dimension of each observation vector. Each object generates a unique observation at each sampling time if it is detected. Let $H^j : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ be the observation model. Then the observations are generated as follows:

$$y_t^j = \begin{cases} H^j(x_t^k) + v_t^j & \text{if } j^{\text{th}} \text{ observation is from } x_t^k \\ u_t & \text{otherwise,} \end{cases} \tag{3.2}$$

where $v_t^j \in \mathbb{R}^{n_y}$ are white noise processes and $u_t \sim \mathrm{Unif}(\mathcal{R})$ is a random process for false alarms.

We assume that targets are indistinguishable in this thesis, but if observations include target type or attribute information, the state variable can be extended to include target type information. The parameters p_{z}, p_{d}, λ_{b} and λ_{f} have been widely used in many multi-target tracking applications [Bar-Shalom and Fortmann, 1988; Kurien, 1990]. Our experimental and simulation results show that our tracking algorithm is robust against changes in these parameters in most cases. See Section 4.3.3.4 for simulation results on the robustness of the algorithm against changes in λ_{b}.

[1]The terms *observation* and *measurement* are used interchangeably in this thesis.

The multi-target tracking problem is to estimate K, $\{t_{\mathrm{i}}^k, t_{\mathrm{f}}^k\}$ and $\{x_t^k : t_{\mathrm{i}}^k \leq t \leq t_{\mathrm{f}}^k\}$, for $k = 1, \ldots, K$, from observations.

3.2 Probabilistic Model

In order to compute Bayesian estimates to the multi-target tracking problem, we need to first specify the probabilistic model of multi-target tracking. This section describes the probabilistic model and derives a formula for computing the posterior (up to a normalizing constant). If ω is the parameter of the probabilistic model of multi-target tracking with the prior distribution $P(\omega)$, Y is a set of all measurements, and $P(Y|\omega)$ is the likelihood of Y given ω, then the posterior $P(\omega|Y)$ can be represented in the following form using Bayes rule:

$$P(\omega|Y) = \frac{P(Y|\omega)P(\omega)}{P(Y)}. \tag{3.3}$$

In our model, ω is an association event, *i.e.*, a partition of measurements such that each element of a partition is a collection of measurements generated by a single target or clutter [Sittler, 1964]. Since, in general, there is no closed-form formula for computing $P(Y)$, one can only compute $P(\omega|Y)$ up to a normalizing constant, which requires computation of $P(Y|\omega)$ and $P(\omega)$. For a fixed association event ω, $P(Y|\omega)$ can be compute by solving a set of single-target tracking problems. Hence, we focus our attention to the derivation of $P(\omega)$ in this section.

A special care is necessary when defining $P(\omega)$: $P(\omega)$ must be defined independently from Y. Since ω can be defined based on the size of measurements, ω is first defined for all possible measurement sizes. Let μ be a T-dimensional vector, *i.e.*, $\mu = [\mu_1, \ldots, \mu_T]^{\mathrm{T}}$, representing the possible numbers of measurements from $t = 1$ to $t = T$, where $\mu_t \in \mathcal{Z} = \{0, 1, 2, \ldots, M\}$ and $M < \infty$ is some large number[2]. For each value of μ, define a set of

[2]The finiteness of M guarantees that $P(\omega)$ has a proper probability distribution. But it does not restrict our model. Since the number of measurements per scan is always finite and M does not appear in computation

measurement indices $\Upsilon_t^\mu = \{(t,1),(t,2),\ldots,(t,\mu_t)\}$ for $\mu_t > 0$, where (t,i) is an index to the i^{th} measurement at time t, and $\Upsilon_t^\mu = \emptyset$ for $\mu_t = 0$. Now let $\Upsilon^\mu = \cup_{t=1}^T \Upsilon_t^\mu$ be an index set to a set of measurements whose size matches μ and the set $\{\Upsilon^\mu : \mu \in \mathcal{Z}^T\}$ contains all possible index sets.

For each μ, let Ω^μ be a collection of partitions of Υ^μ such that, for $\omega \in \Omega^\mu$, $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$, where τ_0 is a set of indices to false alarms and τ_k is a set of indices to measurements from target k, for $k = 1,\ldots,K$. More formally, $\omega \in \Omega^\mu$ is defined as following:

1. $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$;
2. $\bigcup_{k=0}^K \tau_k = \Upsilon^\mu$ and $\tau_i \cap \tau_j = \emptyset$ for $i \neq j$;
3. τ_0 is a set of indices to false alarms;
4. $|\tau_k \cap \Upsilon_t^\mu| \leq 1$ for $k = 1,\ldots,K$ and $t = 1,\ldots,T$; and
5. $|\tau_k| \geq 2$ for $k = 1,\ldots,K$.

Here, $K = K(\omega)$ is the number of tracks for the given partition $\omega \in \Omega^\mu$ and $|S|$ denotes the cardinality of the set S. We call τ_k a track when there is no confusion although the actual track is the set of estimated states from the observations indexed by τ_k. (We assume there is a deterministic function that returns a set of estimated states given a set of observations, so no distinction is required.) The fourth requirement says that a track can have at most one observation at each time, but, in the case of multiple sensors with overlapping sensing regions, we can easily relax this requirement to allow multiple observations per track. A track is assumed to contain at least two observations since we cannot distinguish a track with a single observation from a false alarm, assuming $\lambda_{\text{f}} > 0$. For special cases, in which $p_{\text{d}} = 1$ or $\lambda_{\text{f}} = 0$, the definition of Ω^μ can be adjusted accordingly.

of the posterior (3.10), we can always assume that M is some number larger than the maximum possible number of measurements per scan.

Example 1. *Let $T = 5$ and $\mu = [2, 2, 2, 2, 2]^T$; then*

$$\Upsilon^{\mu} = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), (4, 1), (4, 2), (5, 1), (5, 2)\}.$$

An example of $\omega \in \Omega^{\mu}$ is $\omega = \{\tau_0, \tau_1, \tau_2\}$, where

$$\begin{aligned} \tau_0 &= \{(3, 2)\} \\ \tau_1 &= \{(1, 1), (2, 1), (3, 1), (4, 1), (5, 1)\} \\ \tau_2 &= \{(1, 2), (2, 2), (4, 2), (5, 2)\}. \end{aligned}$$

This example is shown in Figure 3.1. $\square$

Now let $\widetilde{\Omega} = \{\omega \in \Omega^{\mu} : \mu \in \mathcal{Z}^T\}$. Notice that $\mu = \mu(\omega)$ is a deterministic function of $\omega \in \widetilde{\Omega}$. In addition, we can compute the following numbers from $\omega \in \widetilde{\Omega}$:

- e_t, the number of targets present at time t with $e_0 = 0$;
- z_t, the number of targets terminated at time t;
- a_t, the number of new targets at time t; and
- d_t, the number of detected targets at time t.

Since these numbers are deterministic functions of $\omega \in \widetilde{\Omega}$, we have

$$P(\omega) = P(\omega, \mathcal{N}) = P(\omega|\mathcal{N})P(\mathcal{N}), \tag{3.4}$$

where $\mathcal{N} = \{\mu_t, e_t, z_t, a_t, d_t : 1 \leq t \leq T\}$. Based on the target termination, target detection, new target arrival, and false alarm models described in Section 3.1, we have

$$\begin{aligned} P(\mathcal{N}) &= \prod_{t=1}^{T} \left[\binom{e_{t-1}}{z_t} p_{\text{z}}^{z_t} (1-p_{\text{z}})^{e_{t-1}-z_t} \binom{e_{t-1}-z_t+a_t}{d_t} p_{\text{d}}^{d_t} (1-p_{\text{d}})^{e_{t-1}-z_t+a_t-d_t} \right. \\ &\times \left. \frac{(\lambda_{\text{b}}V)^{a_t}}{a_t!} \exp(-\lambda_{\text{b}}V) \frac{(\lambda_{\text{f}}V)^{\mu_t-d_t}}{(\mu_t-d_t)!} \exp(-\lambda_{\text{f}}V) \right]. \end{aligned} \tag{3.5}$$

Since $\omega \in \widetilde{\Omega}$ with the same $\mathcal{N}$ are indistinguishable, *i.e.*, invariant under permutation of target indices, they are exchangeable and we assign a uniform prior on them. Hence,

$$P(\omega|\mathcal{N}) \propto \prod_{t=1}^{T} \left[\binom{e_{t-1}}{z_t} \binom{e_{t-1}-z_t+a_t}{d_t} \binom{\mu_t}{d_t} \binom{d_t}{a_t} (d_t-a_t)! \right]^{-1}. \tag{3.6}$$

Combining (3.5) and (3.6), the prior of $\omega \in \widetilde{\Omega}$ can be simplified as

$$P(\omega) \propto \prod_{t=1}^{T} \frac{1}{\mu_t!} p_{\text{z}}^{z_t} (1-p_{\text{z}})^{e_{t-1}-z_t} p_{\text{d}}^{d_t} (1-p_{\text{d}})^{e_{t-1}-z_t+a_t-d_t} (\lambda_{\text{b}}V)^{a_t} (\lambda_{\text{f}}V)^{\mu_t-d_t}. \tag{3.7}$$

We simplify (3.7) by letting $c_t = e_{t-1} - z_t$ be the number of targets from time $t-1$ that have not terminated at time t, $g_t = e_{t-1} - z_t + a_t - d_t$ be the number of undetected targets, and $f_t = \mu_t - d_t$ be the number of false alarms. Then, the prior model (3.7) becomes

$$P(\omega) \propto \prod_{t=1}^{T} \frac{1}{\mu_t!} p_{\text{z}}^{z_t} (1-p_{\text{z}})^{c_t} p_{\text{d}}^{d_t} (1-p_{\text{d}})^{g_t} (\lambda_{\text{b}}V)^{a_t} (\lambda_{\text{f}}V)^{f_t}. \tag{3.8}$$

Let $Y_t = \{y_t^j : j = 1, \ldots, n_t\}$ be all measurements at time t and $Y = \{Y_t : 1 \leq t \leq T\}$ be all measurements from $t = 1$ to $t = T$. Applying Bayes rule, the posterior of $\omega \in \widetilde{\Omega}$

becomes:

$$\begin{aligned} P(\omega|Y) &\propto P(Y|\omega)P(\omega) \\ &\propto P(Y|\omega)\prod_{t=1}^{\mathrm{T}} \frac{1}{\mu_t!} p_{\mathrm{z}}^{z_t}(1-p_{\mathrm{z}})^{c_t} p_{\mathrm{d}}^{d_t}(1-p_{\mathrm{d}})^{g_t}(\lambda_{\mathrm{b}}V)^{a_t}(\lambda_{\mathrm{f}}V)^{f_t}, \end{aligned} \tag{3.9}$$

where $P(Y|\omega)$ is the likelihood of observations Y given $\omega \in \widetilde{\Omega}$.

It is important to notice that $P(Y|\omega) = 0$ if $\mu(\omega) \neq n(Y)$, where

$$n(Y) = [n_1(Y), \ldots, n_T(Y)]^{\mathrm{T}}$$

denotes the number of measurements at each time in Y. Hence, we can restrict our attention to those $\omega \in \widetilde{\Omega}$ with $\mu(\omega) = n(Y)$. This crucial observation makes the numerous computations based on (3.9) practical. The set of all possible associations is now defined as $\Omega := \Omega^{n(Y)} = \{\omega \in \widetilde{\Omega} : \mu(\omega) = n(Y)\}$ and Ω is used instead of $\widetilde{\Omega}$ throughout this thesis. Thus, it is convenient to view Ω as a collection of partitions of Y. An example of one such partition is shown in Figure 3.1.

The posterior (3.9) can be further simplified as

$$P(\omega|Y) \propto P(Y|\omega)\prod_{t=1}^{\mathrm{T}} p_{\mathrm{z}}^{z_t}(1-p_{\mathrm{z}})^{c_t} p_{\mathrm{d}}^{d_t}(1-p_{\mathrm{d}})^{g_t}(\lambda_{\mathrm{b}}V)^{a_t}(\lambda_{\mathrm{f}}V)^{f_t}, \tag{3.10}$$

where the term $\prod_{t=1}^{T} V^{a_t+f_t}$ will be canceled out by the matching initial state and false alarm densities in $P(Y|\omega)$. The likelihood $P(Y|\omega)$ can be computed based on the chosen dynamic and measurement models. For example, the computation of $P(Y|\omega)$ for linear dynamic and measurement models can be found in [Oh *et al.*, 2004].

The posterior $P(\omega|Y)$ can be applied to both MAP and Bayesian approaches to solve

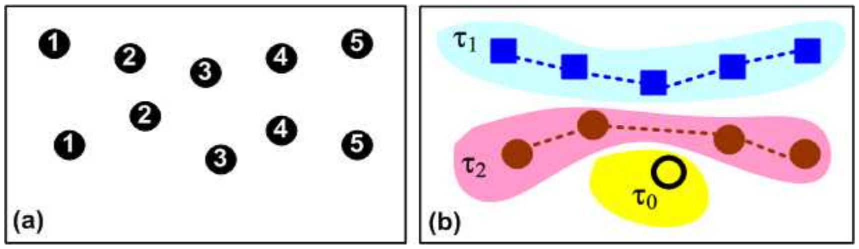


Figure 3.1: (a) An example of observations Y (each circle represents an observation and numbers represent observation times). (b) An example of a partition ω of Y. This ω is also described in Example 1.

the multi-target tracking problem. In the MAP approach, we first seek for $\hat{\omega}$ such that

$$\hat{\omega} = \arg\max_{\omega \in \Omega} P(\omega|Y). \tag{3.11}$$

Then the states of the targets are estimated based on $\hat{\omega}$. In the Bayesian approach, we look for Bayesian estimates of parameters. For instance, if we are interested in estimating the state x_t^k of target k, the Bayesian estimate of x_t^k is (when the mean squared error is used as a risk function):

$$\hat{x}_t^k = \sum_{\omega \in \Omega} \int x_t^k P(x_t^k|\omega, Y) P(\omega|Y) dx_t^k. \tag{3.12}$$

Notice that it considers the contribution of all ω when computing $\hat{x}_t^k$, whereas the MAP approach uses only $\hat{\omega}$. It is important to note that when the number of targets is not fixed, a unique labeling of each target is required to find $\hat{x}_t^k$ under the Bayesian approach (see Section 4.3.3.5 for an example). The method proposed in this thesis (Algorithm 7) can be used to find both MAP and Bayesian estimates to the multi-target tracking problem.

CHAPTER 4

MARKOV CHAIN MONTE CARLO DATA ASSOCIATION

Based on the general framework for multi-target tracking problems developed in Chapter 3, this chapter develops an efficient algorithm for solving general multi-target tracking problems. Multi-target tracking algorithms are often categorized according to the objective function that they purport to optimize:

- *Heuristic* approaches typically involve no explicit objective function. For example, the greedy nearest-neighbor filter (NNF) [Bar-Shalom and Fortmann, 1988] processes the new measurements in some order and associates each with the target whose predicted position is closest, thereby selecting a single association after each scan. Although effective under benign conditions, the NNF gives order-dependent results and breaks down under more difficult circumstances.

- *Maximum a posteriori* (MAP) approaches find the most probable association, given the measurements made so far, and estimate tracks given this association.

- The *Bayesian* approaches generate optimal filtering predictions by summing over all possible associations, weighted by their probabilities. Under certain distributional assumptions (*e.g.*, linear–Gaussian models), the optimal Bayesian filter can be shown to minimize the mean squared error in the track estimates. For this reason, approaches that sum over multiple associations are sometimes called *minimum mean square error* (MMSE) approaches.

MAP approaches and some heuristic approaches use the Bayesian formalism. However, we distinguish them from the Bayesian approaches in which parameters are treated as random variables, following the interpretation given in [Jordan, 2004]. In fact, MAP is considered as a *frequentist* approach [Jordan, 2004].

Tracking algorithms can also be categorized by the way in which they process measurements:

- *Single-scan* algorithms estimate the current states of targets based on their previously computed tracks and the current scan of measurements.
- *Multi-scan* algorithms may revisit past scans when processing each new scan, and can thereby revise previous association decisions in the light of new evidence.

MAP approaches include the well-known *multiple hypothesis tracking* (MHT) algorithm [Reid, 1979]. MHT is a multi-scan tracking algorithm that maintains multiple hypotheses associating past measurements with targets. When a new set of measurements arrives, a new set of hypotheses is formed from each previous hypothesis. The algorithm returns a hypothesis with the highest posterior as a solution. MHT is categorized as a "deferred logic" method [Poore, 1995] in which the decision about forming a new track or removing an existing track is delayed until enough measurements are collected. MHT is capable of initiating and terminating a varying number of tracks and is suitable for autonomous surveillance applications. The main disadvantage of MHT in its pure form is its computational complexity since the number of hypotheses grows exponentially over time. Various heuristic methods have been developed to control this growth [Reid, 1979; Kurien, 1990; Cox and Hingorani, 1996]; but these methods are applied at the expense of sacrificing the MAP property. Other MAP approaches have been tried besides MHT, including 0-1 integer programming [Morefield, 1971] and multidimensional assignment [Poore, 1995]. As the latter reference shows, the underlying MAP data association problem is NP-hard, so we do not expect to find efficient, exact algorithms.

Exact Bayesian data association is even less tractable than the MAP computation. Several "pseudo-Bayesian" methods have been proposed, of which the best-known is the *joint probabilistic data association* (JPDA) filter [Bar-Shalom and Fortmann, 1988]. JPDA is a suboptimal single-scan approximation to the optimal Bayesian filter; it can also be viewed as an assumed-density filter in which the joint state estimate is always a single set of tracks for a "known" set of targets. At each time step, instead of finding a single best association between measurements and tracks, JPDA enumerates all possible associations and computes association probabilities $\{\beta_{jk}\}$, where β_{jk} is the probability that j^{th} measurement extends the k^{th} track. Given an association, the state of a target is estimated by a filtering algorithm and this conditional state estimate is weighted by the association probability. Then the state of a target is estimated by summing over the weighted conditional estimates. JPDA has proved very effective in cluttered environments compared with NNF [Bar-Shalom and Fortmann, 1988]. The exact calculation of association probabilities $\{\beta_{jk}\}$ in JPDA, which requires the summation over all association event probabilities, is NP-hard [Collins and Uhlmann, 1992] since the related problem of finding the permanent of a matrix is #P-complete [Valiant, 1979]. A #P-complete problem is computationally equivalent to computing the number of accepting computations of a polynomial-time nondeterministic Turing machine and #P contains NP [Jerrum and Sinclair, 1996]. Some heuristic approaches to approximate JPDA include a "cheap" JPDA algorithm [Fitzgerald, 1990], "suboptimal" JPDA [Roecker and Phillis, 1993] and "near-optimal" JPDA [Roecker, 1994]. In [Huang and Russell, 1997], a single-scan data association problem is considered and a leave-one-out heuristic is developed to avoid the enumeration of all possible associations.

This chapter develops a real-time multi-target tracking method called Markov chain Monte Carlo data association (MCMCDA). Unlike MHT and JPDA, MCMCDA is a true approximation scheme for the optimal Bayesian filter; *i.e.*, when run with unlimited resources, it converges to the Bayesian solution. As the name suggests, MCMCDA uses Markov chain Monte Carlo (MCMC) sampling instead of summing over all possible associations. MCMC was first used to solve data association problems by Pasula *et al.* [Pasula *et al.*, 1999; Pasula, 2003], who showed it to be effective for multi-camera traffic surveillance problems involving hundreds of vehicles. More recently, in [Cong *et al.*, 2004], MCMC was used to approximate the association probabilities in JPDA and was shown to outperform Fitzgerald's cheap JPDA. MCMC has also been used for problems that are roughly isomorphic to the data association problem, including state estimation in the switching Kalman filter [Bergman and Doucet, 2000] and stereo correspondence in computer vision [Dellaert *et al.*, 2003]. MCMCDA goes beyond these contributions by incorporating missing measurements, false alarms and an ability to initiate and terminate tracks, so that the algorithm can be applied to the full range of data association problems. In addition, MCMCDA is robust against the uncertainty in the number of targets.

This chapter has two main technical results. The first is a theorem showing that, when the number of targets is fixed, single-scan MCMCDA is a fully polynomial randomized approximation scheme for JPDA. More specifically, for any $\epsilon > 0$ and any $0 < \eta < 0.5$, the algorithm finds "good estimates" with probability at least $1 - \eta$ in time complexity $O(\epsilon^{-2} \log \eta^{-1} N(N \log N + \log(\epsilon^{-1})))$, where N is the number of measurements per scan. (The precise meaning of good estimates is defined in Section 4.2.3.) The theorem is based on the seminal work of Jerrum and Sinclair [Jerrum and Sinclair, 1996], who designed an MCMC algorithm for approximating the permanent of a matrix and developed new techniques for analyzing its rate of convergence. As mentioned above, the relationship between JPDA and computing the permanent was identified by Collins and Uhlmann [Collins and

Uhlmann, 1992]; the connection to the polynomial-time approximation theorems of Jerrum and Sinclair was first suggested by Pasula *et al.* [Pasula *et al.*, 1999]. Although our proof has the same structure as that of Jerrum and Sinclair, substantial technical work was required to complete the mapping from computing the permanent to solving JPDA, including the usage of gating conditions that ensure appropriate lower bounds on individual association probabilities. In addition, we also present simulation results supporting our theoretical results.

Our second technical result is the complete specification of the transition structure for a multi-scan version of MCMCDA that includes detection failure, false alarms, and track initiation and termination. We prove that the resulting algorithm converges to the full Bayesian solution. We also provide the first extensive experimental investigation of MCMCDA's performance on classical data association problems. We demonstrate remarkably effective real-time performance compared to MHT under extreme conditions, such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates.

But first, we describe the Markov chain Monte Carlo (MCMC) method.

4.1 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) plays a significant role in many fields such as physics, statistics, economics, finance, and engineering [Gilks *et al.*, 1996; Beichl and Sullivan, 2000; Eraker, 2001]. The MCMC method includes algorithms such as Gibbs sampling [Geman and Geman, 1984] and the Metropolis-Hastings algorithm [Metropolis *et al.*, 1953; Hastings, 1970]. Beichl and Sullivan described the Metropolis-Hastings algorithm as "the most successful and influential of all the members of ... the *Monte Carlo Method*" [Beichl and Sullivan, 2000]. MCMC techniques have been applied to complex probability distribution integration problems, counting problems, and combinatorial optimization problems [Beichl and Sullivan, 2000]. In some cases, MCMC is the only known general algorithm that finds a good approximate solution to a complex problem in polynomial time [Jerrum and Sinclair, 1996].

MCMC is a general method to generate samples from a distribution π on a space Ω by constructing a Markov chain $\mathcal{M}$ with states $\omega \in \Omega$ and stationary distribution $\pi(\omega)$. We now describe an MCMC algorithm known as the Metropolis-Hastings algorithm. If we are at state $\omega \in \Omega$, we propose $\omega' \in \Omega$ following the proposal distribution $q(\omega, \omega')$. The move is accepted with an acceptance probability $A(\omega, \omega')$ where

$$A(\omega, \omega') = \min\left(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}\right), \tag{4.1}$$

otherwise the sampler stays at ω. With this construction, the detailed balance condition is satisfied, *i.e.*, for all $\omega, \omega' \in \Omega$ with $\omega' \neq \omega$,

$$Q(\omega, \omega') = \pi(\omega)P(\omega, \omega') = \pi(\omega')P(\omega', \omega), \tag{4.2}$$

where $P(\omega, \omega') = q(\omega, \omega')A(\omega, \omega')$ is the transition probability from ω to ω'.

A Markov chain is *irreducible* when every state is accessible from every other state. A Markov chain is *periodic* if there exists at least one state to which the Markov chain returns with a fixed time period greater than one. A Markov chain is *aperiodic* if no such state exists. If $\mathcal{M}$ is irreducible and aperiodic, then $\mathcal{M}$ converges to its stationary distribution by the ergodic theorem [Roberts, 1996]. Hence, for any bounded function f, the sample mean $\hat{f} = \frac{1}{T}\sum_{t=1}^{T} f(\omega_t)$ converges to $\mathbb{E}_\pi f(\omega)$ as $T \to \infty$, where ω_t is the state of $\mathcal{M}$ at time t and $\mathbb{E}_\pi f(\omega)$ is the expected value of $f(\omega)$ with respect to measure π. Notice that (4.1) requires only the ability to compute the ratio $\pi(\omega')/\pi(\omega)$, avoiding the need to normalize π, and this is why MCMC, especially the Metropolis-Hastings algorithm, can be applied to a wide range of applications.

An ergodic chain $\mathcal{M}$ on state space Ω converges to its stationary distribution asymptotically. But a practical question is how fast $\mathcal{M}$ approaches stationarity. One way to measure the rate of convergence of $\mathcal{M}$ to stationarity is the "mixing time" of the Markov chain. Let P be the transition probabilities of $\mathcal{M}$ and let $P^t_\omega(\cdot)$ be the distribution of the state at time t given that $\mathcal{M}$ is started from the initial state $\omega \in \Omega$. If π is the stationary distribution of $\mathcal{M}$, then the *total variation distance* at time t with initial state ω is defined as

$$\Delta_\omega(t) = \|P^t_\omega - \pi\|_{\text{tv}} = \max_{S \subset \Omega} |P^t_\omega(S) - \pi(S)| = \frac{1}{2} \sum_{y \in \Omega} |P^t_\omega(y) - \pi(y)|. \tag{4.3}$$

The rate of convergence of $\mathcal{M}$ to stationarity can be measured by the *mixing time*

$$\tau_\omega(\epsilon) = \min\{t : \Delta_\omega(s) \leq \epsilon \text{ for all } s \geq t\}. \tag{4.4}$$

After the mixing time $\tau_\omega(\epsilon)$, $P^t_\omega(\cdot)$ for $t \geq \tau_\omega(\epsilon)$ is very close to the stationary distribution π.

One approach to bound $\tau_\omega(\epsilon)$ of a Markov chain with a complex structure is the canonical path method [Jerrum and Sinclair, 1996]. We use the canonical path method to bound $\tau_\omega(\epsilon)$ of the Markov chain simulated by the MCMCDA algorithm given in Section 4.2. For the remainder of this section, we describe the canonical path method.

For a finite, reversible and ergodic Markov chain $\mathcal{M}$ with state space Ω, consider an undirected graph $G = (V, E)$ where $V = \Omega$ and $E = \{(x, y) : Q(x, y) > 0\}$ (recall the definition of $Q(\cdot, \cdot)$ from (4.2)). So an edge $(x, y) \in E$ indicates that the Markov chain $\mathcal{M}$ can make a transition from x to y or from y to x in a single step. For each ordered pair $(x, y) \in \Omega^2$, the canonical path γ_{xy} is a simple path[1] from x to y in G. In terms of $\mathcal{M}$, the canonical path γ_{xy} is a sequence of legal transitions from x to y in $\mathcal{M}$. Let $\Gamma = \{\gamma_{xy} : x, y \in \Omega\}$ be the set of all canonical paths. Now the mixing time of the chain is

[1]A simple path in a graph is a path with no repeated vertices.

related to the *maximum edge loading*:

$$\bar{\rho} = \bar{\rho}(\Gamma) = \max_{e} \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y)|\gamma_{xy}|, \tag{4.5}$$

where $|\gamma_{xy}|$ denotes the length of the path γ_{xy}. If $\bar{\rho}$ is not so big, *i.e.*, no single edge is overloaded, then the Markov chain can mix rapidly. The main result for the canonical path method is as follows [Jerrum and Sinclair, 1996; Diaconis and Stroock, 1991]:

Theorem 4. *Let $\mathcal{M}$ be a finite, reversible, ergodic Markov chain with loop probabilities $P(x,x) \geq \frac{1}{2}$ for all states x. Let Γ be a set of canonical paths with maximum edge loading $\bar{\rho}$. Then the mixing time of $\mathcal{M}$ satisfies $\tau_\omega(\epsilon) \leq \bar{\rho}(\log \pi(x)^{-1} + \log \epsilon^{-1})$, for any choice of initial state ω.*

4.2 Single-scan MCMCDA

In this section, we consider a special case of the multi-target tracking problem described in Chapter 3, in which the number of targets is fixed and known, and propose the single-scan MCMCDA. Then, we prove that the single-scan MCMCDA algorithm finds an approximate solution to JPDA in polynomial time.

4.2.1 Single-scan Bayesian Filter

The single-scan Bayesian filter is a generalization of the JPDA filter [Bar-Shalom and Fortmann, 1988] for the general dynamics and measurement models defined in Chapter 3. JPDA has been traditionally used with the Kalman filter, assuming linear–Gaussian models, *i.e.*, linear dynamic and measurement models and white Gaussian noise processes [Bar-Shalom and Fortmann, 1988]. Recently, JPDA has also been applied with a nonlinear filtering algorithm such as particle filters [Schulz *et al.*, 2001]. The single-scan MCMCDA filter described in Section 4.2.2 is based on the single-scan Bayesian filter but differs from the JPDA filter in that the association probabilities are approximated by MCMC. The description of the single-scan MCMCDA filter for linear–Gaussian models is given in [Oh and Sastry, 2005a].

The single-scan Bayesian filter processes each measurement scan sequentially and computes the posterior distribution of the current states of targets based on the current measurements and the posterior distribution computed at the previous scan. The states of targets can be estimated from the posterior distribution. Since not all past measurements are considered when computing the posterior distribution, the single-scan Bayesian filter, including the JPDA filter, is suboptimal [Bar-Shalom and Fortmann, 1988]. In addition, when the dynamics and measurement model are nonlinear or the noise processes are non-Gaussian, the posterior distribution can be only approximated using techniques such as linearization, unscented filtering [Julier and Uhlmann, 2004], interacting multiple models [Blom and Bar-Shalom, 1988], or particle filters [de Freitas and Gordon, 2001]. Notice that, for linear–Gaussian models such as those used in [Bar-Shalom and Fortmann, 1988; Oh and Sastry, 2005a], the posterior is a Gaussian distribution which is completely described by its mean and variance.

Now suppose that we have the posterior distribution $\hat{P}(X_{t-1}^k | y_{1:t-1})$ computed from the previous time $t-1$, for each target k, where X_{t-1}^k is the state of target k at time $t-1$, $y_{1:t-1} = \{y_1, \ldots, y_{t-1}\}$ is a set of measurements, and $\hat{P}(X|y)$ is an approximation of the distribution $P(X|y)$. At time t, the following three steps show how the single-scan Bayesian filter computes $\hat{P}(X_t^k | y_{1:t})$ from the new measurements y_t and the previous posterior distribution $\hat{P}(X_{t-1}^k | y_{1:t-1})$. In the following description of the single-scan Bayesian filter, we follow the notations defined in Chapter 3, except that random variables are denoted by capital letters.

Step 1 (Prediction): For each k, compute the distribution

$$\begin{aligned} \hat{P}(X_t^k|y_{1:t-1}) &:= \int P(X_t^k|x_{t-1}^k, y_{1:t-1})\hat{P}(x_{t-1}^k|y_{1:t-1})dx_{t-1}^k \\ &= \int P(X_t^k|x_{t-1}^k)\hat{P}(x_{t-1}^k|y_{1:t-1})dx_{t-1}^k, \end{aligned} \tag{4.6}$$

where the Markovian assumption is used in the second equality and $P(X_t^k|x_{t-1}^k)$ is determined by the noise process w_t^k in (3.1).

Step 2 (Measurement Validation): For each k and j, compute the distribution

$$\begin{aligned} \hat{P}^k(Y_t^j|y_{1:t-1}) &:= \int P(Y_t^j|x_t^k, y_{1:t-1})\hat{P}(x_t^k|y_{1:t-1})dx_t^k \\ &= \int P(Y_t^j|x_t^k)\hat{P}(x_t^k|y_{1:t-1})dx_t^k, \end{aligned} \tag{4.7}$$

where the second equality uses the fact the current observation is independent of previous observations given the current state and $P(Y_t^j|x_t^k)$ is determined by the noise process v_t^j in (3.2). Here, $\hat{P}^k(y_t^j|y_{1:t-1})$ is the probability density of having observation y_t^j given $y_{1:t-1}$, when y_t^j is a measurement originated from target k. (Again, for linear–Gaussian models, $\hat{P}^k(Y_t^j|y_{1:t-1})$ is a Gaussian distribution and completely determined by its mean and variance.) As in JPDA, we validate measurements and use only validated measurements when estimating states of targets. The measurement y_t^j is validated for target k, if and only if

$$\hat{P}^k(y_t^j|y_{1:t-1}) \geq \delta^k, \tag{4.8}$$

where δ^k are appropriate thresholds. An example of validation is shown in Figure 4.1.

Step 3 (State Estimation): Let Ω be a set of all feasible (joint) association events at time t. For notational convenience, the subscript t is dropped when there is no confusion. For each $\omega \in \Omega$, $\omega = \{(j,k)\}$, where (j,k) denotes an event that observation j is associated with target k. An association event ω is feasible when (i) for each $(j,k) \in \omega$, y_t^j is validated

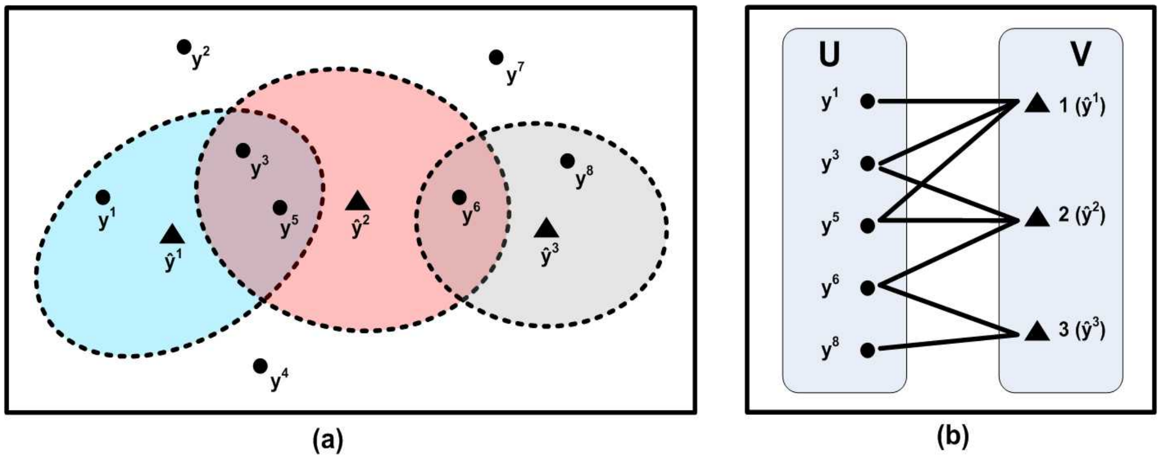


Figure 4.1: (a) An example of measurement validation. For this 2D example, $\hat{P}^k(Y_t^j|y_{1:t-1})$ has a Gaussian distribution with mean $\hat{y}^k$ for $k = 1, 2, 3$ (shown as a solid triangle). Measurements $\{y^j : j = 1, 2, \dots, 8\}$ are shown as disks. A measurement is validated for target k if it is inside the shaded region centered at $\hat{y}^k$. (b) Measurement validation encoded as a bipartite graph $G = (U, V, E)$. An edge between $y^j \in U$ and $k \in V$ indicates that measurement y^j is validated for target k and $(y^j, k) \in E$. (The subscript t is omitted.)

for target k; (ii) an observation is associated with at most one target; and (iii) a target is associated with at most one observation.

Let $N = n_t$ be the number of validated observations. We encode the feasible association events in a bipartite graph. Let $G = (U, V, E)$ be a bipartite graph, where $U = \{y_t^j : 1 \leq j \leq N\}$ is a vertex set of validated observations, $V = \{k : 1 \leq k \leq K\}$ is a vertex set of target indices, and $E = \{(u, v) : u \in U, v \in V, \hat{P}^v(u|y_{1:t-1}) \geq \delta^v\}$. An edge $(u, v) \in E$ indicates that observation u is validated for target v according to (4.8). An example of measurement validation encoded as a bipartite graph is shown in Figure 4.1(b). A feasible association event is a *matching* in G, *i.e.*, a subset $M \subset E$ such that no two edges in M share a vertex. The set of all feasible association events Ω can be represented as $\Omega = M_0(G) \cup \cdots \cup M_K(G)$, where $M_k(G)$ is a set of k-matchings in G. Some examples of matchings or feasible association events are shown in Figure 4.2.

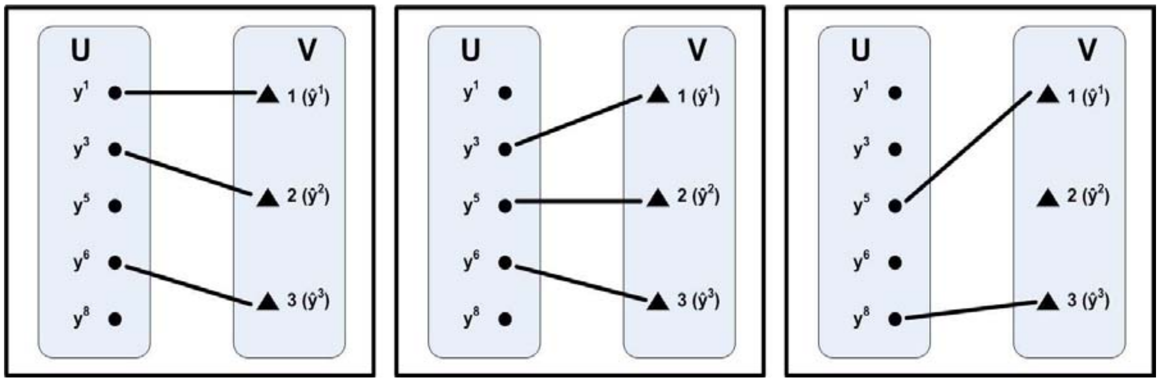


Figure 4.2: Examples of matchings (feasible association events) based on the measurement validation example given in Figure 4.1.

Now using the total probability theorem, we can compute the approximate distribution as:

$$\hat{P}(X_t^k|y_{1:t}) := \sum_{\omega\in\Omega} \hat{P}(X_t^k|\omega, y_{1:t})\hat{P}(\omega|y_{1:t}) = \sum_{j=0}^{N} \beta_{jk}\hat{P}(X_t^k|\omega_{jk}, y_{1:t}), \tag{4.9}$$

where ω_{jk} denotes the event $\{\omega \ni (j,k)\}$, ω_{0k} denotes the event that no observation is associated with target k, and β_{jk} is an association probability, such that,

$$\beta_{jk} = \hat{P}(\omega_{jk}|y_{1:t}) = \sum_{\omega:(j,k)\in\omega} \hat{P}(\omega|y_{1:t}). \tag{4.10}$$

$\hat{P}(X_t^k|\omega_{jk}, y_{1:t})$ in (4.9) can be easily computed by considering it as a single target estimation problem with a single observation. Hence, the computation of $\hat{P}(X_t^k|y_{1:t})$ reduces to the computation of β_{jk}. The computation of β_{jk} requires a summation over the posterior, hence the enumeration of all association events. In JPDA, $\mathbb{E}(X_t^k|y_{1:t})$ is estimated in the same manner as (4.9) and JPDA is a method for estimating expectations such as $\mathbb{E}(X_t^k|y_{1:t})$ using the association probabilities $\{\beta_{jk}\}$ in the presence of identity uncertainty. As mentioned earlier, the exact calculation of $\{\beta_{jk}\}$ in JPDA is NP-hard [Collins and Uhlmann, 1992] and this is the major drawback of JPDA. In the following sections, we describe the single-scan MCMCDA filter which approximates the association probabilities $\{\beta_{jk}\}$ and prove that the running time of the algorithm is polynomial in the size of the problem.

4.2.2 Single-scan MCMCDA Filter

The single-scan MCMCDA filter shares the same filtering steps of the single-scan Bayesian filter described in Section 4.2.1, except that the association probabilities $\{\beta_{jk}\}$ in (4.10) are approximated using MCMC. Since the filtering steps are already described in Section 4.2.1, we only describe how the single-scan MCMCDA filter approximates $\{\beta_{jk}\}$ in this section.

Based on the parametric false alarm model described in Section 3.1 and the derivation similar to (3.8), for each $\omega \in \Omega$, the prior $P(\omega)$ can be written as

$$P(\omega) \propto (\lambda_{\mathrm{f}} V)^{N-|\omega|} p_{\mathrm{d}}^{|\omega|} (1-p_{\mathrm{d}})^{K-|\omega|}. \tag{4.11}$$

Then, the posterior of $\omega \in \Omega$ can be written as

$$\begin{aligned}
P(\omega|y_{1:t}) &\overset{(a)}{=} \frac{1}{Z_0} P(\omega|y_{1:t-1}) P(y_t|\omega, y_{1:t-1}) \\
&\overset{(b)}{=} \frac{1}{Z_0} P(\omega) P(y_t|\omega, y_{1:t-1}) \\
&\overset{(c)}{\approx} \frac{1}{Z} P(\omega) \hat{P}(y_t|\omega, y_{1:t-1}) \\
&\overset{(d)}{=} \frac{1}{Z} \lambda_{\mathrm{f}}^{N-|\omega|} p_{\mathrm{d}}^{|\omega|} (1-p_{\mathrm{d}})^{K-|\omega|} \prod_{(u,v)\in\omega} \hat{P}^{v}(u|y_{1:t-1}) \\
&=: \hat{P}(\omega|y_{1:t}),
\end{aligned} \tag{4.12}$$

where Z_0 and Z are normalizing constants; Bayes rule is used in (a); (b) follows from the fact that ω is independent of $y_{1:t-1}$; (c) follows from the fact that $\hat{P}(y_t|\omega, y_{1:t-1})$ is an approximation of $P(y_t|\omega, y_{1:t-1})$; and the prior (4.11) is used in (d) and $V^{N-|\omega|}$ is canceled

Algorithm 5 Single-scan MCMCDA

Input: $G = (U, V, E), n_{\text{mc}}, n_{\text{bi}}, \theta = \{\{\hat{P}^v(u|y_{1:t-1})\}, \lambda_{\text{f}}, p_{\text{d}}, K, N\}$
Output: $\{\hat{\beta}_{jk}\}$

```
 1: $\hat{\beta}_{jk} = 0$ for all $j$ and $k$
 2: choose $\omega_0$ randomly from $\Omega$
 3: for $n = 1$ to $n_{\text{mc}}$ do
 4:     $\omega_n$ = Single-scan MCMCDA.single-step($G, \omega_{n-1}, \theta$) (see Algorithm 6)
 5:     if $n > n_{\text{bi}}$ then
 6:         for each $(y^j, k) \in \omega_n$ do
 7:             $\hat{\beta}_{jk} = \hat{\beta}_{jk} + 1/(n_{\text{mc}} - n_{\text{bi}})$
 8:         end for
 9:     end if
10: end for
```

by the matching false alarm density in $\hat{P}(y_t|\omega, y_{1:t-1})$.

The MCMC data association (MCMCDA) algorithm is an MCMC algorithm whose state space is the set of all feasible association events Ω and whose stationary distribution is the posterior $\hat{P}(\omega|y_{1:t})$ (4.12). The single-scan MCMCDA algorithm is shown in Algorithm 5 along with its MCMC step described in Algorithm 6. The inputs to Algorithm 5 are the graph G, the number of samples n_{mc}, the number of burn-in samples n_{bi}, and θ. The input θ contains likelihoods $\{\hat{P}^v(u|y_{1:t-1})\}$ and model parameters $\lambda_{\text{f}}, p_{\text{d}}, K$, and N. Algorithm 5 computes the approximate association probabilities $\{\hat{\beta}_{jk}\}$, which can be used in (4.9) to compute the approximate posterior distribution $\hat{P}(X_t^k|y_{1:t})$. In Algorithm 6, since we have a uniform proposal distribution, $A(\omega, \omega') = \min\left(1, \frac{\pi(\omega')}{\pi(\omega)}\right)$, where $\pi(\omega) = \hat{P}(\omega|y_{1:t})$ from (4.12). Notice that, in line 2 of Algorithm 6, a self-loop transition probability of 0.5 introduced to make the analysis easier (see p.18 of [Jerrum and Sinclair, 1993] for more detail). In practice, however, the self-loop transition probability in line 2 can be set to 0 for faster convergence.

Algorithm 6 Single-scan MCMCDA.single-step

```
Input: $G = (U, V, E), \omega, \theta$
Output: $\omega_{\text{new}}$
1:  sample $U$ from Unif$[0, 1]$
2:  if $U < \frac{1}{2}$ then
3:      $\omega' = \omega$
4:  else
5:      choose $e = (u, v) \in E$ uniformly at random
6:      if $e \in \omega$ then
7:          $\omega' = \omega - e$            (deletion move)
8:      else if both $u$ and $v$ are unmatched in $\omega$ then
9:          $\omega' = \omega + e$            (addition move)
10:     else if exactly one of $u$ and $v$ is matched in $\omega$ and $e'$ is the matching edge then
11:         $\omega' = \omega + e - e'$       (switch move)
12:     else
13:         $\omega' = \omega$
14:     end if
15: end if
16: $\omega = \omega'$ with probability $A(\omega, \omega')$
17: $\omega_{\text{new}} = \omega$
```

4.2.3 Analysis

Let $\mathcal{M}$ be the Markov chain simulated by Algorithm 6. Since the self-loop probability is nonzero, $\mathcal{M}$ is aperiodic. It can be easily seen that $\mathcal{M}$ is irreducible, *i.e.*, all states communicate, for example via the empty matching. In addition, the transitions described in Algorithm 6 satisfy the detailed balance condition (4.2) so $\mathcal{M}$ is reversible. Hence, by the ergodic theorem, the chain converges to its stationary distribution [Roberts, 1996].

Let us first take a look at the complexity of the problem. As noted earlier, the state space of the Markov chain $\mathcal{M}$ is $\Omega = M_0(G) \cup \cdots \cup M_K(G)$. For each k, $|M_k(G)| \leq \binom{K}{k} \frac{N!}{(N-k)!}$ with equality if the subgraph of G with the k chosen vertices in V is a complete bipartite graph, *i.e.*, all observations are validated for all k chosen targets. Hence, we can bound the

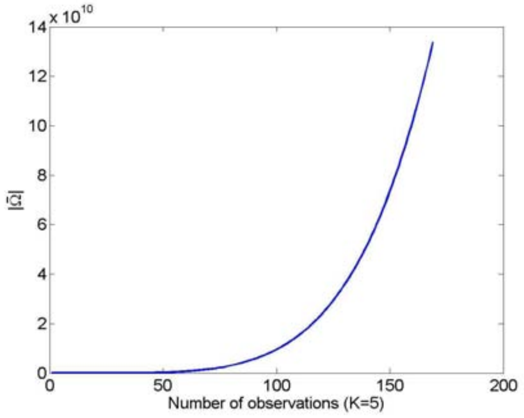


Figure 4.3: $\bar{\Omega}$ as a function of the number of observations when $K = 5$

size of Ω as

$$|\Omega| = |M_0(G)| + \cdots + |M_K(G)| \leq \sum_{k=0}^{K} \binom{K}{k} \frac{N!}{(N-k)!} =: \bar{\Omega}. \tag{4.13}$$

Figure 4.3 shows this bound for $K = 5$ as a function of the number of observations. Certainly, the size of the state space grows exponentially as the number of targets or the number of observations increases, hence the exact calculation of JPDA by enumeration is not feasible when the number of targets or the number of observations is large. (See Section 4.2.4 for experimental results.)

We assume that each likelihood term can be bounded as $\underline{L} \leq \hat{P}^v(u|y_{1:t-1}) \leq \bar{L}$, for all $(u, v) \in E$. The lower bound $\underline{L} = \min \delta^k$ is due to the measurement validation. In JPDA, the measurement validation is used to reduce the number of feasible association events. However, we later find that it is required to approximate the association probabilities in polynomial time. The upper bound $\bar{L}$ can be precomputed based on $\hat{P}^v(u|y_{1:t-1})$. Here, we are making a reasonable assumption that $\hat{P}^v(u|y_{1:t-1}) \leq \bar{L} < \infty$ for all $(u, v) \in E$. (An example of $\bar{L}$ for linear–Gaussian models can be found in [Oh and Sastry, 2005a].)

The following theorems show that the single-scan MCMCDA algorithm provides a fully polynomial randomized approximation scheme for JPDA.

Theorem 5. *Suppose that $\lambda_f > 0$ and $0 < p_d < 1$. Then the mixing time of the Markov chain $\mathcal{M}$ is bounded by*

$$\tau_\omega(\epsilon) \leq 4R^4K^2N(m_0(K,N) + \log \epsilon^{-1}) \tag{4.14}$$

for all $\omega \in \Omega$, where

$$\begin{aligned} R &= \max\left\{1, \frac{p_d\bar{L}}{\lambda_f(1-p_d)}, \frac{\lambda_f(1-p_d)}{\underline{L}p_d}\right\}, \\ m_0(K,N) &= K\log\frac{m_1}{m_2} + \log\frac{m_3(K,N)}{m_4(K,N)} + \sum_{k=1}^{K+1}\log k + \sum_{n=1}^{N}\log n \end{aligned}$$

with $m_1 = \max\{1, \bar{L}\}$, $m_2 = \min\{1, \underline{L}\}$, $m_3(K,N) = \max_{0\leq k\leq K}\{\lambda_f^{N-k}p_d^k(1-p_d)^{K-k}\}$, and $m_4(K,N) = \min_{0\leq k\leq K}\{\lambda_f^{N-k}p_d^k(1-p_d)^{K-k}\}$.

Proof: See Appendix B.1. ■

Remark 1. *If $0.5 < p_d < 1$ and $\lambda_f < 1-p_d$, then $m_3(K,N) = \lambda_f^{N-K}p_d^K$ and $m_4(K,N) = \lambda_f^N(1-p_d)^K$. So $m_3(K,N)/m_4(K,N) = \left(\frac{p_d}{\lambda_f(1-p_d)}\right)^K$ and K is the only remaining exponent.*

Remark 2. *Let $\bar{\tau}(\epsilon)$ be the upper bound found in Theorem 5. $\bar{\tau}(\epsilon)$ is polynomial in K and N. If $m_3(K,N)/m_4(K,N)$ does not grow fast,* e.g.*, Remark 1, $\bar{\tau}(\epsilon) = O(K^2N(K\log K + N\log N + \log\epsilon^{-1}))$. If K is fixed, $\bar{\tau}(\epsilon) = O(N(N\log N + \log\epsilon^{-1}))$.*

Let $p(\omega)$ be the distribution of the states of $\mathcal{M}$ after simulating Algorithm 6 for at least $\bar{\tau}(\epsilon)$ steps. Then the total variation distance satisfies $\|p-\pi\|_{\text{tv}} \leq \epsilon$. So we can sample from p to estimate $\{\beta_{jk}\}$. However, there is a small bias in our estimates since we are not sampling directly from π. The following theorem gives an upper bound on the number of samples needed for finding good estimates.

Theorem 6. *Let $0 < \epsilon_1, \epsilon_2 \leq 1$ and $0 < \eta < 0.5$. Suppose that $\|p - \pi\|_{tv} \leq \epsilon$ for $\epsilon \leq \epsilon_1\epsilon_2/8$. Then, with a total of $504\epsilon_1^{-2}\epsilon_2^{-1}\lceil \log \eta^{-1} \rceil$ samples from p, we can find estimates $\hat{\beta}_{jk}$ for β_{jk} with probability at least $1-\eta$, such that, for $\beta_{jk} \geq \epsilon_2$, $\hat{\beta}_{jk}$ estimates β_{jk} within ratio $1+\epsilon_1$,* i.e., *$(1-\epsilon_1)\beta_{jk} \leq \hat{\beta}_{jk} \leq (1+\epsilon_1)\beta_{jk}$, and, for $\beta_{jk} < \epsilon_2$, $|\hat{\beta}_{jk} - \beta_{jk}| \leq (1+\epsilon_1)\epsilon_2$.*

Proof: See Appendix B.2. ■

Remark 3. *Following Remark 2, for fixed K, $\bar{\tau}(\epsilon) = O(N(N \log N + \log \epsilon^{-1}))$. Combining this fact with Theorem 6, the time complexity of the overall procedure is*

$$\tilde{n}_{mc} = O(\epsilon_1^{-2}\epsilon_2^{-1} \log \eta^{-1} N(N \log N + \log(\epsilon_1^{-1}\epsilon_2^{-1}))). \tag{4.15}$$

Hence, with a total of $\tilde{n}_{mc}$ samples, Algorithm 6 finds estimates $\hat{\beta}_{jk}$ for β_{jk} with probability at least $1-\eta$, such that, for $\beta_{jk} \geq \epsilon_2$, $\hat{\beta}_{jk}$ estimates β_{jk} within ratio $1+\epsilon_1$, and, for $\beta_{jk} < \epsilon_2$, $|\hat{\beta}_{jk} - \beta_{jk}| \leq (1+\epsilon_1)\epsilon_2$. We can simplify further by letting $\epsilon_0 = \epsilon_1\epsilon_2$. Then the time complexity is $O(\epsilon_0^{-2} \log \eta^{-1} N(N \log N + \log(\epsilon_0^{-1})))$.

4.2.4 Simulation Results

A simple case is chosen to demonstrate single-scan MCMCDA, in which there are five predicted observations and 12 actual observations over a 4×4 two-dimensional region (see Figure 4.4(a)). $\hat{P}^k(y_t^j|y_{1:t-1})$ has a Gaussian distribution with zero mean and covariance $B^k = \text{diag}(0.5, 0.5)$ for all k. The other parameters used in this simulation are: $\lambda_\text{f} = 0.5$, $p_\text{d} = 0.8$, and $\delta^k = p((y_t^j - \hat{y}^k)^\text{T}(B^k)^{-1}(y_t^j - \hat{y}^k) = 4)$ for all k. Each predicted observation has at least 10 validated measurements in this case.

The true values of $\{\beta_{jk}\}$ are computed using JPDA. In order to study the convergence of the single-scan MCMCDA algorithm, we ran 100 independent runs with initial states randomly chosen from Ω. For each run, two types of estimates are made at each MCMC step: (type $r=1$) $\hat{\beta}_{jk}^1$, which are computed after $\bar{\tau}(\epsilon)$ burn-in samples; and (type $r=2$)

$\hat{\beta}^2_{jk}$, which are computed after $10,000$ burn-in samples. Let $\hat{\beta}^r_{jk}(m,n)$ be the estimate made at the n^{th} MCMC step for the m^{th} run, for type $r \in \{1,2\}$. $\bar{\tau}(\epsilon)$ is computed using $\epsilon_1 = 0.1$, $\epsilon_2 = 0.01$, $\eta = 0.05$, and $\epsilon = \epsilon_1\epsilon_2/8$. In this case, $\bar{\tau}(\epsilon) = 5.25 \times 10^6$ and $\beta_{jk} \geq \epsilon_2$ for all (j,k) pairs. The results are shown in Figure 4.4(b). The figure shows a pair of envelopes, one for each type of estimate. The top curve of an envelope plots the maximum approximation ratio

$$R^r_{\text{max}}(n) = \max_{m=1,\ldots,100} \max_{jk} \frac{\hat{\beta}^r_{jk}(m,n)}{\beta_{jk}} \tag{4.16}$$

and the bottom curve plots the minimum approximation ratio

$$R^r_{\text{min}}(n) = \min_{m=1,\ldots,100} \min_{jk} \frac{\hat{\beta}^r_{jk}(m,n)}{\beta_{jk}} \tag{4.17}$$

for $r \in \{1,2\}$. The envelope for type $r = 1$ starts from $n = \bar{\tau}(\epsilon)$ and type $r = 2$ starts from $n = 10,000$.

A couple of observations can be made from Figure 4.4(b). Based on Theorem 6, it requires at least $\tilde{n}_{\text{mc}} = 1.5 \times 10^7$ samples to make sure that the estimate using $\bar{\tau}(\epsilon)$ burn-in samples approximates the true value with ratio less than ϵ_1, *i.e.*, the envelop $(R^r_{\text{max}}(n), R^r_{\text{min}}(n))$ is completely contained in $(1+\epsilon_1, 1-\epsilon_1)$ for $n > \tilde{n}_{\text{mc}}$. But both estimates achieve the approximation ratio of ϵ_1 much faster than $\tilde{n}_{\text{mc}}$. As frequently observed in many practical applications of MCMC, we observe that the algorithm requires a significantly smaller number of burn-in samples than what the theorem requires. This is a reasonable observation since the theorem is based on the worst-case analysis. For this example, 10,000 burn-in samples were enough, *i.e.*, 500 times less than $\bar{\tau}(\epsilon)$. Lastly, in theory, we find good approximations with probability at least $1-\eta$. But we observe that estimates from all 100 runs are within the approximation ratio of ϵ_1.

Next, we added one more predicted observation at $[\sqrt{2}/2, \sqrt{2}/2]^{\text{T}}$ to the previous example. There are more than 5×10^5 association events for this example and it was difficult

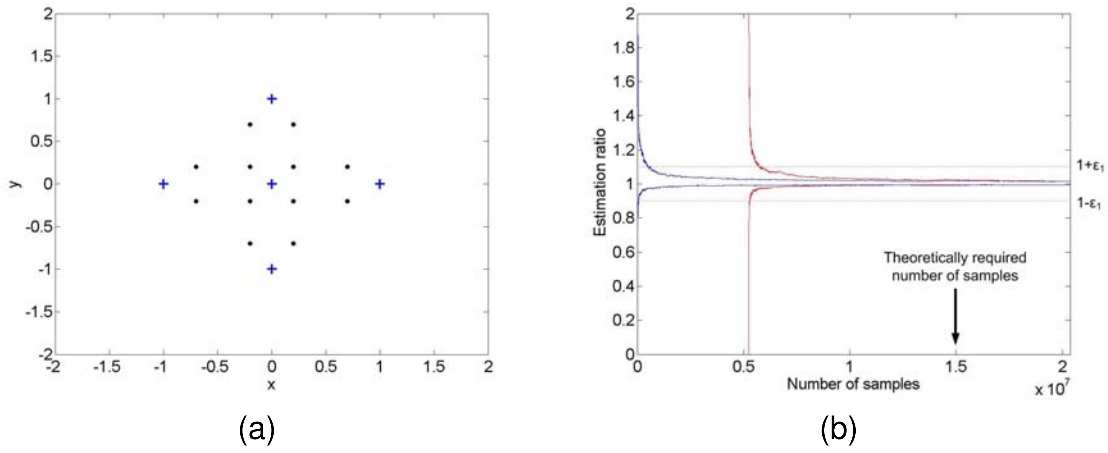


Figure 4.4: (a) Predicted observations (crosses) and actual observations (dots). (b) Maximum and minimum approximation ratios (R^r_{max}, R^r) for two types of estimates ($r \in \{1, 2\}$) computed from 100 independent single-scan MCMCDA runs. The approximation ratios for $r = 1$ start from $\bar{\tau}(\epsilon) = 5.25 \times 10^6$ and the approximation ratios for $r = 2$ start from $10,000$. The dotted ϵ_1-tube centered at 1 represents the goal approximation ratio ϵ_1. If $(R^r_{\text{max}}, R^r_{\text{min}})$ is completely contained in $(1 + \epsilon_1, 1 - \epsilon_1)$, we have achieved our goal approximation ratio. Theoretically, it requires at least $\tilde{n}_{\text{mc}} = 1.5 \times 10^7$ samples to make sure $(R^r_{\text{max}}(n), R^r_{\text{min}}(n))$ is contained in $(1 + \epsilon_1, 1 - \epsilon_1)$ for $n > \tilde{n}_{\text{mc}}$. But both estimates achieve the approximation ratio of ϵ_1 much faster than $\tilde{n}_{\text{mc}}$.

to compute the association probabilities using JPDA. But the single-scan MCMCDA algorithm found good estimates with run-time compatible to the previous example.

4.3 Multi-scan MCMCDA

In this section, we present an algorithm for solving the multi-target tracking problem described in Chapter 3. The algorithm is presented in Section 4.3.1 and its performance is compared against MHT in Section 4.3.3.

4.3.1 Multi-scan MCMCDA Algorithm

The multi-scan MCMCDA algorithm is described in Algorithm 7. It is an MCMC algorithm whose state space is Ω as defined in Section 3.2 and whose stationary distribution is the posterior (3.10). The proposal distribution for MCMCDA consists of eight moves grouped into five types as follows: (1) birth/death move pair; (2) split/merge move pair; (3) extension/reduction move pair; (4) track update move; and (5) track switch move. (See Figure 4.5.) We index each move by an integer such that $m = 1$ for a birth move, $m = 2$ for a death move and so on. The move m is chosen randomly from the distribution $\xi_K(m)$ where K is the number of tracks of the current partition ω. When there is no track, we can only propose a birth move, so we set $\xi_0(m = 1) = 1$ and 0 for all other moves. When there is only a single target, we cannot propose a merge or track switch move, so $\xi_1(m = 4) = \xi_1(m = 8) = 0$. For other values of K and m, we assume $\xi_K(m) > 0$. The inputs for MCMCDA are the set of all observations Y, the number of samples n_{mc}, the initial state ω_{init}, and model parameters $p_{\text{z}}, p_{\text{d}}$, and λ_{b}. When we want to estimate $\mathbb{E}_\pi X$ of a bounded function $X : \Omega \to \mathbb{R}^n$, MCMCDA can also take the function X as an input. At each step of the algorithm, ω is the current state of the Markov chain. The acceptance probability $A(\omega, \omega')$ is defined in (4.1) where $\pi(\omega) = P(\omega|Y)$ from (3.10). The output $\hat{X}$ approximates the Bayesian posterior expectation $\mathbb{E}_\pi X$ and $\hat{\omega}$ approximates the MAP estimate $\arg\max P(\omega|Y)$. The computation of $\hat{\omega}$ can be viewed as simulated annealing [Kirkpatrick *et al.*, 1983] at a constant temperature. Notice that MCMCDA can provide both MAP and Bayesian solutions to the multi-target tracking problem.

An MCMC algorithm can be specialized and made more efficient by incorporating domain-specific knowledge. In multi-target tracking, we can make two assumptions: (1) the maximum directional speed of any target in $\mathcal{R}$ is less than some $\bar{v}$; and (2) the number of consecutive missing observations of any track is less than some $\bar{d}$. The first assumption is reasonable in a surveillance scenario since, in many cases, the maximum speed of a

Algorithm 7 Multi-scan MCMCDA

```
Input: $Y, n_{\text{mc}}, \omega_{\text{init}}, p_{\text{z}}, p_{\text{d}}, \lambda_{\text{b}}, X : \Omega \to \mathbb{R}^n$
Output: $\hat{\omega}, \hat{X}$
1: $\omega = \omega_{\text{init}}$; $\hat{\omega} = \omega_{\text{init}}$; $\hat{X} = 0$
2: for $n = 1$ to $n_{\text{mc}}$ do
3:     propose $\omega'$ based on $\omega$ (see sections 4.3.1.1 to 4.3.1.5)
4:     sample $u$ from Unif$[0, 1]$
5:     $\omega = \omega'$ if $u < A(\omega, \omega')$
6:     $\hat{\omega} = \omega$ if $P(\omega|Y)/P(\hat{\omega}|Y) > 1$
7:     $\hat{X} = \frac{n}{n+1}\hat{X} + \frac{1}{n+1}X(\omega)$
8: end for
```

vehicle is generally known based on the vehicle type and terrain conditions. The second assumption is a user-defined parameter. Let $p_{\text{dt}}(s) = 1 - (1 - p_{\text{d}})^s$ be the probability that an object is observed at least once out of s measurement times. Then, for given $\bar{p}_{\text{dt}}$, we set $\bar{d} \geq \log(1 - \bar{p}_{\text{dt}})/\log(1 - p_{\text{d}})$ to detect a track with probability at least $\bar{p}_{\text{dt}}$. For example, given $p_{\text{d}} = 0.7$ and $\bar{p}_{\text{dt}} = 0.99$, a track is detected with probability larger than 0.99 for $\bar{d} \geq 4$. We will now assume that these two new conditions are added to the definition of Ω so each element $\omega \in \Omega$ satisfies these two additional assumptions.

The incorporation of the constraints $\bar{v}$ and $\bar{d}$ is carried out by devising a data structure, which groups temporally separated observations based on distances with respect to the constraints. This data structure is formally defined as following:

$$L_d(y_t^j) = \{y_{t+d}^k \in y_{t+d} : \varphi(y_t^j, y_{t+d}^k) \leq d \cdot \bar{v}\} \tag{4.18}$$

for $d = 1, \ldots, \bar{d}$, $j = 1, \ldots, n_t$ and $t = 1, \ldots, T-1$. Here $\varphi : \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is an appropriate metric, *e.g.*, for a Cartesian coordinate system, φ is induced by the Euclidean norm. L_d is used in Algorithm 7 to propose a new state ω' from the current state ω and the parameter d allows missing observations. The use of this data structure makes the algorithm more scalable since distant observations will be considered separately and makes the computations of the proposal distribution easier. It is similar to the clustering technique

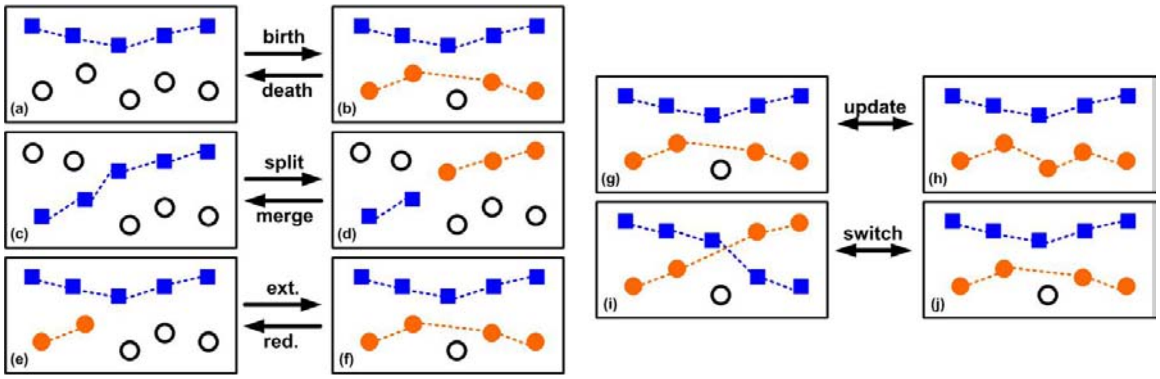


Figure 4.5: Graphical illustration of MCMCDA moves (associations are indicated by dotted lines and hollow circles are false alarms). Each move proposes a new joint association event ω' which is a modification of the current joint association event ω. The birth move proposes ω' by forming a new track from the set of false alarms ($(a) \rightarrow (b)$). The death move proposes ω' by combining one of the existing tracks into the set of false alarms ($(b) \rightarrow (a)$). The split move splits a track from ω into two tracks ($(c) \rightarrow (d)$) while the merge move combines two tracks in ω into a single track ($(d) \rightarrow (c)$). The extension move extends an existing track in ω ($(e) \rightarrow (f)$) and the reduction move reduces an existing track in ω ($(f) \rightarrow (e)$). The track update move chooses a track in ω and assigns different measurements from the set of false alarms ($(g) \leftrightarrow (h)$). The track switch move chooses two track from ω and switches some measurement-to-track associations ($(i) \leftrightarrow (j)$).

used in MHT but L_d in MCMCDA is fixed for a given set of observations.

We now describe each move of the sampler in detail. First, let $\zeta(d)$ be a distribution of a random variable d taking values from $\{1, 2, \ldots, \bar{d}\}$. We assume the current state of the chain is $\omega = \omega^0 \cup \omega^1 \in \Omega$, where $\omega^0 = \{\tau_0\}$ and $\omega^1 = \{\tau_1, \ldots, \tau_K\}$. The proposed partition is denoted by $\omega' = \omega'^0 \cup \omega'^1 \in \Omega$. Note the abuse of notation below with indexing of time, *i.e.*, when we say $\tau(t_i)$, t_i means the time at which a target corresponding to the track τ is observed i times.

4.3.1.1 Birth and Death Moves (Figure 4.5 , $a \leftrightarrow b$)

For a birth move, we increase the number of tracks from K to $K' = K + 1$ and select t_1 uniformly at random (u.a.r.) from $\{1, \ldots, T-1\}$ as an appearance time of a new track. Let $\tau_{K'}$ be the track of this new object. Then we choose d_1 from the distribution ζ. Let $L^1_{d_1} = \{y^j_{t_1} : L_{d_1}(y^j_{t_1}) \neq \emptyset, y^j_{t_1} \notin \tau_k(t_1), j = 1, \ldots, n_{t_1}, k = 1, \ldots, K\}$. $L^1_{d_1}$ is a set of observations at t_1 such that, for any $y \in L^1_{d_1}$, y does not belong to other tracks and $L_{d_1}(y)$ is not empty. We choose $\tau_{K'}(t_1)$ u.a.r. from $L^1_{d_1}$. If $L^1_{d_1}$ is empty, the move is rejected since the move is not reversible. Once the initial observation is chosen, we then choose the subsequent observations for the track $\tau_{K'}$. For $i = 2, 3, \ldots$, we choose d_i from ζ and choose $\tau_{K'}(t_i)$ u.a.r. from $L_{d_i}(\tau_{K'}(t_{i-1})) \setminus \{\tau_k(t_{i-1} + d_i) : k = 1, \ldots, K\}$ unless this set is empty. But, for $i = 3, 4, \ldots$, the process of adding observations to $\tau_{K'}$ terminates with probability γ, where $0 < \gamma < 1$. If $|\tau_{K'}| \leq 1$, the move is rejected. We then propose this modified partition where $\omega'^1 = \omega^1 \cup \{\tau_{K'}\}$ and $\omega'^0 = \{\tau_0 \setminus \tau_{K'}\}$. For a death move, we simply choose k u.a.r. from $\{1, \ldots, K\}$ and delete the k^{th} track and propose a new partition where $\omega'^1 = \omega^1 \setminus \{\tau_k\}$ and $\omega'^0 = \{\tau_0 \cup \tau_k\}$.

4.3.1.2 Split and Merge Moves (Figure 4.5 , $c \leftrightarrow d$)

For a split move, we select $\tau_s(t_r)$ u.a.r. from $\{\tau_k(t_i) : |\tau_k| \geq 4, i = 2, \ldots, |\tau_k| - 2, k = 1, \ldots, K\}$. Then we split the track τ_s into τ_{s_1} and τ_{s_2} such that $\tau_{s_1} = \{\tau_s(t_i) : i = 1, \ldots, r\}$ and $\tau_{s_2} = \{\tau_s(t_i) : i = r + 1, \ldots, |\tau_s|\}$. The modified track partition becomes $\omega'^1 = (\omega^1 \setminus \{\tau_s\}) \cup \{\tau_{s_1}\} \cup \{\tau_{s_2}\}$ and $\omega'^0 = \omega^0$. For a merge move, we consider the following set of possible merge move pairs:

$$M = \{(\tau_{k_1}(t_f), \tau_{k_2}(t_1)) : \tau_{k_2}(t_1) \in L_{t_1 - t_f}(\tau_{k_1}(t_f)), f = |\tau_{k_1}| \text{ for } k_1 \neq k_2, 1 \leq k_1, k_2 \leq K\}.$$

We select a pair $(\tau_{s_1}(t_f), \tau_{s_2}(t_1))$ u.a.r. from M. The tracks are combined into a single track $\tau_s = \tau_{s_1} \cup \tau_{s_2}$. Then we propose a new partition where $\omega'^1 = (\omega^1 \setminus (\{\tau_{s_1}\} \cup \{\tau_{s_2}\})) \cup \{\tau_s\}$ and $\omega'^0 = \omega^0$.

4.3.1.3 Extension and Reduction Moves (Figure 4.5 , $e \leftrightarrow f$)

In a track extension move, we select a track τ u.a.r. from K available tracks in ω. We reassign observations for τ after the disappearance time $t_{|\tau|}$ as done in the track birth move. For a track reduction move, we select a track τ u.a.r. from K available tracks in ω and r u.a.r. from $\{2, \ldots, |\tau| - 1\}$. We shorten the track τ to $\{\tau(t_1), \ldots, \tau(t_r)\}$ by removing the observations assigned to τ after the time t_{r+1}.

4.3.1.4 Track Update Move (Figure 4.5 , $g \leftrightarrow h$)

In a track update move, we select a track τ u.a.r. from K available tracks in ω. Then we pick r u.a.r. from $\{1, 2, \ldots, |\tau|\}$ and reassign observations for τ after the time t_r as done in the track birth move.

4.3.1.5 Track Switch Move (Figure 4.5 , $i \leftrightarrow j$)

For a track switch move, we select a pair of observations $(\tau_{k_1}(t_p), \tau_{k_2}(t_q))$ from two different tracks such that, $\tau_{k_1}(t_{p+1}) \in L_d(\tau_{k_2}(t_q))$ and $\tau_{k_2}(t_{q+1}) \in L_{d'}(\tau_{k_1}(t_p))$, where $d = t_{p+1} - t_q$, $d' = t_{q+1} - t_p$ and $0 < d, d' \leq \bar{d}$. Then we let

$$\begin{aligned}
\tau_{k_1} &= \{\tau_{k_1}(t_1), \ldots, \tau_{k_1}(t_p), \tau_{k_2}(t_{q+1}), \ldots, \tau_{k_2}(t_{|\tau_{k_2}|})\} \\
\tau_{k_2} &= \{\tau_{k_2}(t_1), \ldots, \tau_{k_2}(t_q), \tau_{k_1}(t_{p+1}), \ldots, \tau_{k_1}(t_{|\tau_{k_1}|})\}.
\end{aligned}$$

The main result of this section is that MCMCDA is an optimal Bayesian filter in the limit. Let $\mathcal{M}$ be the Markov chain specified by Algorithm 7. Then we have:

Theorem 7. *Suppose that* $0 < p_z, p_d < 1$ *and* $\lambda_b, \lambda_f > 0$. *If* $\zeta(d) > 0$, *for all* $d \in \{1, \ldots, \bar{d}\}$, *then the Markov chain* $\mathcal{M}$ *is ergodic and* $\hat{X} \to \mathbb{E}_\pi X$ *as* $n_{mc} \to \infty$.

See Appendix B.3 for the proof of the theorem and see Section 4.3.3.5 for a numerical demonstration of the theorem.

4.3.2 Online MCMCDA

The MCMCDA algorithm described in previous section is a batch algorithm and its computational complexity grows as more measurements are collected. Since recent measurements are more relevant to the current states, good estimates of the current states can still be found from recent measurements. Based on this idea, we propose an online MCMCDA algorithm whose estimates are based on measurements from a window of time $[t_{\text{curr}} - t_{\text{win}} + 1, \ldots, t_{\text{curr}}]$, where t_{curr} is the current time and t_{win} is the size of a window. Hence, at all times, only a finite number of measurements are kept by the algorithm. This online implementation of MCMCDA shown in Algorithm 8 is suboptimal because it considers only a subset of past measurements.

At each time step, we use the previous MAP estimate to initialize MCMCDA and run MCMCDA on the measurements $Y_{\text{w}}(t_{\text{curr}}) = \{y_t^j : 1 \le j \le n_t, t_{\text{curr}} - t_{\text{win}} + 1 \le t \le t_{\text{curr}}\}$ belonging to the current window. At time $t_{\text{curr}} + 1$, the measurements at time $t_{\text{curr}} - t_{\text{win}} + 1$ are removed from Y_{w} and a set of newly arrived measurements $Y_{\text{new}}(t_{\text{curr}})$ is appended to $Y_{\text{w}}(t_{\text{curr}})$. Any delayed measurements are inserted into the appropriate slots. Then, we initialize the Markov chain with the previously estimated tracks and execute Algorithm 7 on $Y_{\text{w}}(t_{\text{curr}})$. The algorithm is summarized in Algorithm 8. The inputs for online MCMCDA at time t_{curr} are the previous MAP estimate $\hat{\omega}(t_{\text{curr}} - 1)$, the existing set of measurements $Y_{\text{w}}(t_{\text{curr}} - 1)$, and the set of new measurements $Y_{\text{new}}(t_{\text{curr}})$. The other inputs are the same as Algorithm 7. Other estimates such as state estimates can be computed using the function X or $\hat{\omega}(t_{\text{curr}})$. The simulation results using online MCMCDA can be found in Section 4.3.3.4.

Algorithm 8 Online MCMCDA (at time t_{curr})

```
Input: $\hat{\omega}(t_{\text{curr}}-1), Y_{\text{w}}(t_{\text{curr}}-1), Y_{\text{new}}(t_{\text{curr}}), n_{\text{mc}}, p_{\text{z}}, p_{\text{d}}, \lambda_{\text{b}}, X : \Omega \to \mathbb{R}^n$
Output: $\hat{\omega}(t_{\text{curr}}), Y_{\text{w}}(t_{\text{curr}})$
1: $Y_{\text{w}}(t_{\text{curr}}) = \{y_t^j \in Y_{\text{w}}(t_{\text{curr}}-1) : t_{\text{curr}} - t_{\text{win}} + 1 \le t \le t_{\text{curr}}\}$
2: add new measurements $Y_{\text{new}}(t_{\text{curr}})$ into $Y_{\text{w}}(t_{\text{curr}})$
3: $\omega_{\text{init}} = \{\tau(t) \in \hat{\omega}(t_{\text{curr}}-1) : t_{\text{curr}} - t_{\text{win}} + 1 \le t \le t_{\text{curr}}\}$
4: $\hat{\omega}(t_{\text{curr}}) = \text{Multi-scan MCMCDA}(Y_{\text{w}}(t_{\text{curr}}), n_{\text{mc}}, \omega_{\text{init}}, p_{\text{z}}, p_{\text{d}}, \lambda_{\text{b}}, X)$ (see Algorithm 7)
```

4.3.3 Simulation Results

In this section, the performance of multi-scan MCMCDA is evaluated and compared against MHT. We consider surveillance over a rectangular region on a plane, $\mathcal{R} = [0, 1000] \times [0, 1000]$. The state vector is $x = [x, y, \dot{x}, \dot{y}]^{\text{T}}$ where (x, y) is a position on $\mathcal{R}$ along the usual x and y axes and $(\dot{x}, \dot{y})$ is a velocity vector. Linear dynamics and a linear measurement model are used:

$$x_{t+1}^k = A x_t^k + G w_t^k \qquad\qquad y_t^j = C x_t^k + v_t^j \tag{4.19}$$

where

$$A = \begin{bmatrix} 1 & 0 & T_{\text{s}} & 0 \\ 0 & 1 & 0 & T_{\text{s}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G = \begin{bmatrix} T_{\text{s}}^2/2 & 0 \\ 0 & T_{\text{s}}^2/2 \\ T_{\text{s}} & 0 \\ 0 & T_{\text{s}} \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^{\text{T}},$$

and T_{s} is the sampling period, w_t^k is a zero-mean Gaussian process with covariance $Q = \text{diag}(100, 100)$, and v_t^j is a zero-mean Gaussian process with covariance $R = \text{diag}(100, 100)$.

The complexity of multi-target tracking problems can be measured by several metrics: (1) the intensity of the false alarm rate λ_{f}; (2) the detection probability p_{d}; and (3) the density of tracks. The problem gets more challenging with increasing λ_{f}, decreasing p_{d}, and increasing density of tracks. The number of tracks itself may not make the problem more difficult if they are scattered apart. The difficulty arises when there are many tracks crossing and moving close to each other; this is when the ambiguity of data association is greatest. Hence, we consider only situations in which tracks move very closely so we can control the density of tracks by the number of tracks.

We study the performance of the MCMCDA algorithm against the greedy algorithm and MHT by varying the parameters listed above. To make the comparison easier, we take the MAP approach, in which the states of targets are estimated from $\hat{\omega}$ computed from Algorithm 7. The greedy algorithm is a batch-mode, nearest-neighbor, multi-target tracking algorithm. Initially, all observations are unmarked. Unmarked observations are considered false alarms. The algorithm first picks two unmarked observations at different times to estimate an initial state. Then it forms a candidate track by picking unmarked observations which are the nearest to the predicted states for the subsequent time steps. The candidate track is validated as a track and observations associated to the candidate track are marked if the marginal of the candidate track exceeds a threshold. The process is repeated until no more tracks can be found. For a more detailed description of the greedy algorithm, see [Oh, 2003].

Based on our model described above and in Chapter 3, we have generated different scenarios. In particular, in all cases, except for the online tracking case, half of the new objects appear from the bottom left quadrant of $\mathcal{R}$ and the other half appear from the the bottom right quadrant. (The actual initial positions are chosen randomly from a 200×200 region in each quadrant.) They all move diagonally so that each group of tracks crosses the other group in the middle of $\mathcal{R}$. The targets also move very close to each other and there are crossovers within each group. All targets are present from $t = 1$ to $t = T$. An example of a scenario with 100 targets is shown in Figure 4.6. In order to measure the density of tracks, the distance between every pair of targets at each time is computed from a test case ($K = 100$) used in Section 4.3.3.1. Figure 4.7 shows two histograms

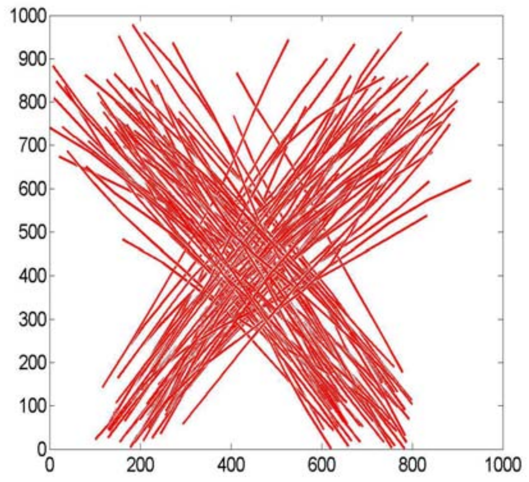
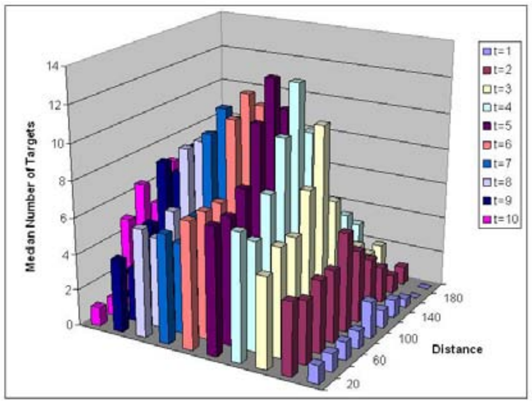


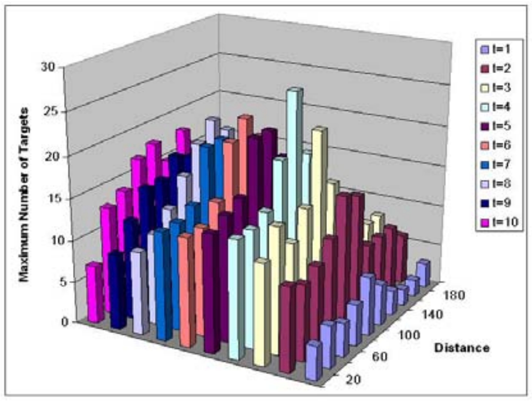
Figure 4.6: A scenario used for the experiment in Section 4.3.3.1 ($K = 100$). A solid line represents a trajectory of a target. A half of new objects appears from the left bottom quadrant of $\mathcal{R}$ and the other half of new objects appears from the the right bottom quadrant. The actual initial positions are chosen randomly from a 200×200 region in each quadrant.

created from these pairwise distances. Figure 4.7(a) shows the median number of targets at different distance ranges while Figure 4.7(b) shows the maximum number of targets at different distance ranges. In this example, between $t = 5$ and $t = 6$, the median number of neighboring targets with distance less than 20 is 7 and some targets have 14 neighboring targets. For distance less than 100, the median number of neighboring targets is 46 and some targets have 90 neighboring targets. Considering that the covariance matrices are $Q = \text{diag}(100, 100)$ and $R = \text{diag}(100, 100)$, the distance of length 100 is about 3.5 times the standard deviation of the joint noise process in two-dimension; hence, this is a highly dense environment.

Since the number of targets is not fixed, it is difficult to compare algorithms using a standard criterion such as the mean square error. Instead, we use a standard performance measure, called F_1, used in the information retrieval literature [van Rijsbergen, 1979; Yang



(a) Histogram of the median number of targets at different distances



(b) Histogram of the maximum number of targets at different distances

Figure 4.7: Histograms of the number of targets from the test case ($K = 100$) used in Section 4.3.3.1 at different target-to-target distances at each simulation time (t). Each simulation time (t) is shown in a different color. The height of each bar represents the number of targets with distances belonging to the corresponding range.

and Liu, 1999]. The F_1 measure is defined in terms of recall and precision. Recall is the ratio of correct associations made by an algorithm divided by the total number of correct associations. Precision is the ratio of correct associations made by an algorithm divided by the total number of associations made by the algorithm. The F_1 measure is a harmonic mean between recall (r) and precision (p) with an equal weight and defined as:

$$F_1(r,p) = \frac{2rp}{r+p}. \tag{4.20}$$

Recall and precision measure the *effectiveness* of an algorithm [van Rijsbergen, 1979]; the higher the value of the F_1 measure, the more effective the algorithm is.

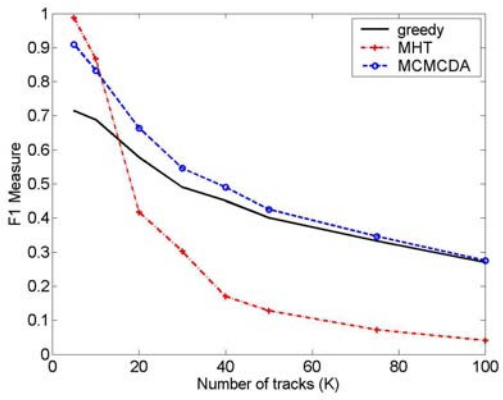
Both MCMCDA and greedy algorithms are written in C++ with MATLAB interfaces. We have used the C++ implementation of MHT [Cox,], which implements pruning, gating, clustering, N-scan-back logic and k-best hypotheses. The parameters for MHT are finetuned so that it gives similar performance to that of MCMCDA when there are 10 targets: the maximum number of hypotheses in a group is 1,000, the maximum track tree depth is 5, and the maximum Mahalanobis distance is 11.8. All simulations are run on a PC with a 2.6-GHz Intel processor.

4.3.3.1 Number of Tracks

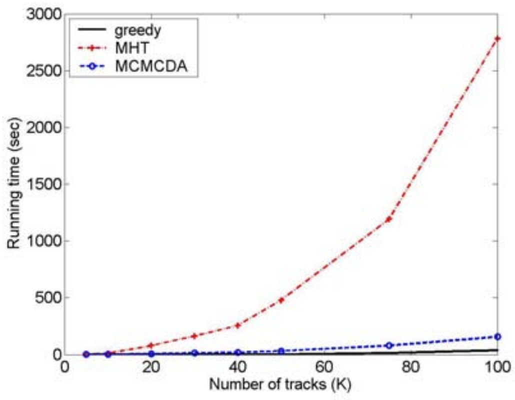
In this experiment, we vary K from 5 to 100. The other parameters are held fixed: $\mathcal{R} = [0, 1000] \times [0, 1000]$, $T = 10$, $p_{\mathrm{d}} = 0.9$, $\lambda_{\mathrm{f}} V = 1$, $\bar{d} = 1$, $\bar{v} = 100$ unit lengths per unit time. A uniform mass function is used for each $\xi_k(\cdot)$ and $\zeta(d)$ is computed based on p_{d}. For each value of K, we randomly generated 10 test cases. The initial state of MCMCDA is calculated with the greedy algorithm and 50,000 samples are used in MCMCDA. For each K, the average F_1 measure and running time are computed from the 10 test cases (for MCMCDA, we also average over 10 runs per test case). The average F_1 measure computed at different K is shown in Figure 4.8(a). The average running times of three algorithms are shown in Figure 4.8(b) (the running time of MCMCDA includes the initialization step). Although the maximum number of hypotheses of 1,000 per group is a large number, with increasing numbers of tracks, the performance of MHT deteriorates due to pruning. But both greedy and MCMCDA maintain good performance, although the greedy algorithm detects fewer tracks for large K. However, we later find that the performance of the greedy algorithm is poor against missing detections as shown in Section 4.3.3.3. Another striking difference is that the running times of both greedy and MCMCDA are significantly less than that of MHT.

4.3.3.2 False Alarms

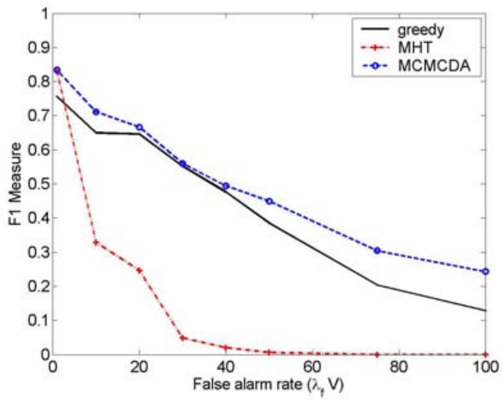
Now the settings are the same as Section 4.3.3.1 but we vary the false alarm rates while the number of tracks is fixed at $K = 10$. The false alarm rates are varied from $\lambda_{\mathrm{f}} V = 1$



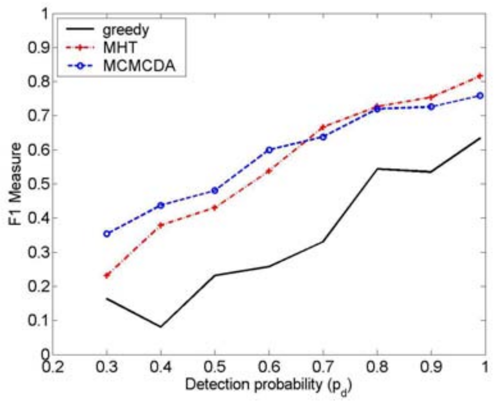
(a) F_1 measure as a function of K



(b) Average running time as a function of K



(c) F_1 measure as a function of $\lambda_{\mathrm{f}}V$



(d) F_1 measure as a function of p_{d}

Figure 4.8: Simulation results

to $\lambda_{\mathrm{f}}V = 100$ with an increment of 10. For each value of $\lambda_{\mathrm{f}}V$, we randomly generated 10 test cases and MCMCDA is run 10 times per test case. Again, 50,000 samples are used for MCMCDA. The average F_1 measures for the three algorithms at different false alarm rates are given in Figure 4.8(c). The results show that MCMCDA performs well at high false alarm rates. The greedy algorithm suffers because it finds too many spurious tracks (poor precision) and MHT becomes hopelessly confused, finding no correct associations at $\lambda_{\mathrm{f}}V \geq 80$.

4.3.3.3 Detection Probability

The detection probability p_d is varied from 0.3 to 0.99 with an increment of 0.1, except the last increment which is 0.09, while keeping the other parameters as the previous experiments except $K = 10$, $\lambda_\mathrm{f} V = 1$, and $\bar{d} = 5$. Now the tracks are not observed at all times. For each value of p_d, we randomly generated 10 test cases and MCMCDA is run 10 times per test case. Again, 50,000 samples are used for MCMCDA. The average F_1 measures for three different algorithms at different detection probabilities are shown in Figure 4.8(d). The overall performance of MCMCDA is comparable to that of MHT. MHT performs better at high detection probability while MCMCDA performs better at low detection probability. The greedy algorithm performed very poorly.

Although, in theory, MHT gives an optimal solution in the sense of MAP, it performs poorly in practice when the detection probability is low or the false alarm rate is high. This is due to the heuristics, such as pruning and N-scan-back techniques, that are required to limit complexity. They work well when a few hypotheses carry most of the weight. When the detection probability is low or the false alarm rate is high, however, there are many hypotheses with appreciable weights and there is no small set of dominating hypotheses, so MHT cannot perform well. A major advantage of the MCMCDA algorithm is that its running time can be regulated by the number of samples and the number of observations but the running time of MHT depends on the complexity of the problem instance, which is not predictable in advance. In addition, the memory required by MCMCDA is significantly less than the memory required by MHT, since MCMCDA is only required to store one association event at a time.

4.3.3.4 Robustness Against the Uncertainty in the Number of Targets

Our probabilistic model assumes that the following parameters are known: detection probability p_d, new target birth rate λ_b, termination probability p_z, and false alarm rate λ_f. It is

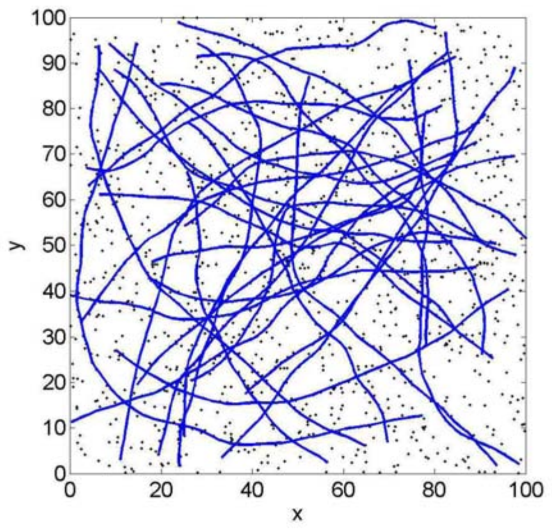


Figure 4.9: An example used in Section 4.3.3.4. The actual trajectories are shown in solid lines and measurements are shown in dots. The surveillance duration is $T = 100$ and there are 30 targets appearing and disappearing at random times.

reasonable to assume that the detection probability, termination probability and false alarm rate are estimated reliably from historic data or from sensor models. However, we cannot reliably estimate the new target birth rate λ_b since we do not know the number of targets ahead of time. Ideally, we want a multi-target tracking algorithm to be robust against the uncertainty in the number of targets, *i.e.*, when λ_b is not known. In this section, we compare online MCMCDA from Section 4.3.2 against MHT by varying λ_b.

An example of tracking multiple targets in a cluttered environment is used for demonstration (see Figure 4.9). For this example, the surveillance duration is $T = 100$ and there are 30 targets appearing and disappearing at random times over the surveillance region $\mathcal{R} = [0, 100] \times [0, 100]$. The size of the sliding window is $t_\mathrm{win} = 10$. The other parameters are: $p_\mathrm{d} = 0.7$, $\lambda_\mathrm{f} = 0.001$, $\bar{d} = 10$, and $\bar{v} = 3$ unit lengths per unit time. The linear model (4.19) is used, but with different covariance matrices $Q = \mathrm{diag}(0.04, 0.04)$ and $R = \mathrm{diag}(0.04, 0.04)$.

Both online MCMCDA and MHT are applied to estimate tracks from this example us-

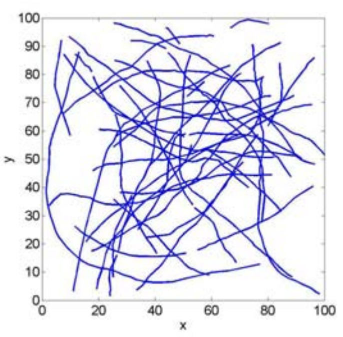
Table 4.1: Comparison between Online MCMCDA and MHT at Different New Target Birth Rates

Algorithm	λ_b	Run-time (per scan)	Estimated Number of Tracks	Unnormalized Log-posterior
Online MCMCDA	0.0005	0.41 sec	47	-10653.38
	0.005	0.42 sec	72	-11037.72
	0.05	0.46 sec	153	-11560.77
MHT	0.0005	0.62 sec	64	-10806.03
	0.005	1.78 sec	107	-11226.27
	0.05	6.72 sec	225	-12906.19

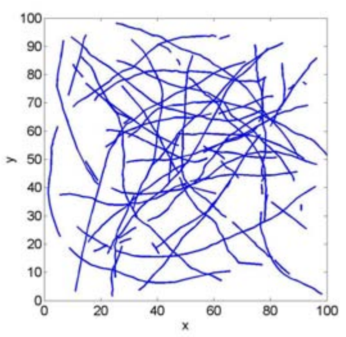
ing three different values of λ_b (0.0005, 0.005, and 0.05). The tracks estimated by online MCMCDA and MHT are shown in Figure 4.10. The algorithm run-times, estimated numbers of tracks, and unnormalized log-posterior of estimated tracks are listed in Table 4.1. For online MCMCDA, 5,000 MCMC samples are used at each scan. For different values of λ_b, online MCMCDA took about the same length of time. But MHT required a significantly more computation time for smaller values of λ_b, *i.e.*, when the algorithm expects more targets than there are. Also notice that more spurious tracks are detected by MHT and online MCMCDA achieved higher unnormalized log-posterior than MHT. This example illustrates that online MCMCDA is more robust than MHT against the uncertainty in the number of targets.

4.3.3.5 Convergence

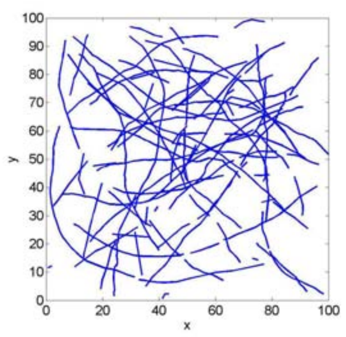
An example is used to demonstrate the convergence of MCMCDA (Theorem 7). In order to compute the exact estimates, we have chosen a simple example where there are four scans ($T = 4$) and three measurements per scan. Hence, there are a total of 12 measurements. But even for this simple case, there are over 45,000 partitions or association events, *i.e.*, the size of Ω is over 45,000. We again used the dynamics and measurement model given (4.19), where $Q = \text{diag}(4, 4)$ and $R = \text{diag}(4, 4)$. The surveillance region is $\mathcal{R} = [0, 100]\times$



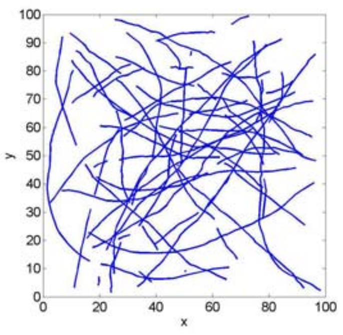
(a) Online MCMCDA, $\lambda_\mathsf{b} = 0.0005$



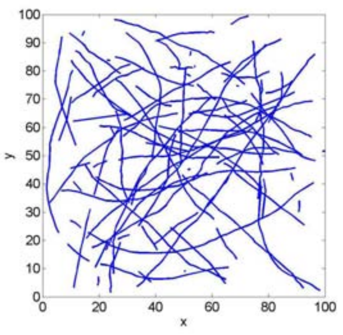
(b) Online MCMCDA, $\lambda_\mathsf{b} = 0.005$



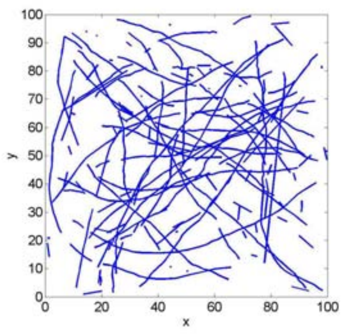
(c) Online MCMCDA, $\lambda_\mathsf{b} = 0.05$



(d) MHT, $\lambda_\mathsf{b} = 0.0005$



(e) MHT, $\lambda_\mathsf{b} = 0.005$



(f) MHT, $\lambda_\mathsf{b} = 0.05$

Figure 4.10: Tracks estimated by online MCMCDA (a)-(c) and by MHT (d)-(f) at different values of λ_b. MHT detected more spurious tracks and required longer running time than online MCMCDA. (See Table 4.1 for the comparison between online MCMCDA and MHT.)

$[0, 100]$. The other parameters are: $p_\mathrm{d} = 0.7$, $\lambda_\mathrm{f} = 0.0013$, $\lambda_\mathrm{b} = 9.38 \times 10^{-4}$, $\bar{d} = 4$, and $\bar{v} = 100$. All the measurements are shown in Figure 4.11(a) and they are

$$
\begin{aligned}
Y_1 &= \{[7.81, 44.58]^\mathrm{T}, [9.46, 51.03]^\mathrm{T}, [7.92, 56.38]^\mathrm{T}\} \\
Y_2 &= \{[28.93, 45.22]^\mathrm{T}, [29.88, 48.16]^\mathrm{T}, [27.31, 52.41]^\mathrm{T}\} \\
Y_3 &= \{[51.30, 52.29]^\mathrm{T}, [51.91, 48.79]^\mathrm{T}, [52.04, 42.47]^\mathrm{T}\} \\
Y_4 &= \{[70.58, 52.10]^\mathrm{T}, [75.70, 45.07]^\mathrm{T}, [66.71, 41.21]^\mathrm{T}\}.
\end{aligned}
$$

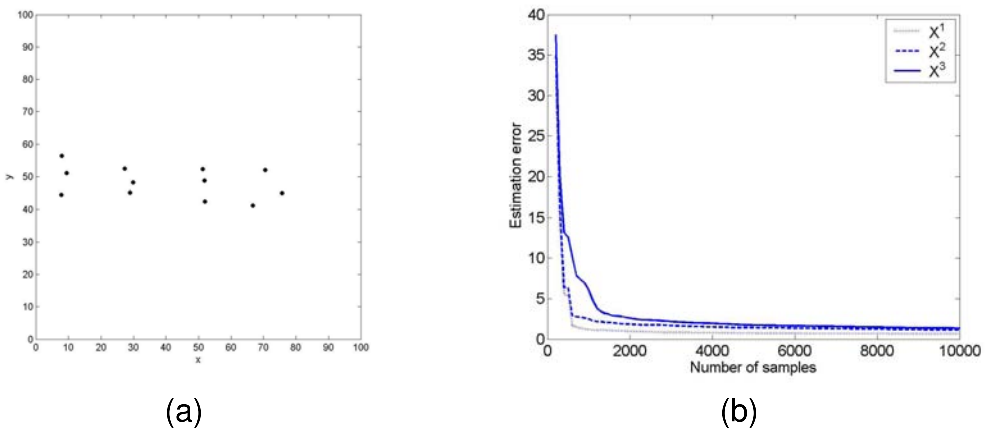


Figure 4.11: (a) All measurements from $t = 1$ to $t = 4$. (b) Average estimation error as a function of the number of MCMC samples. The dotted line represents the estimation error of $\hat{X}^1$ in estimating $\mathbb{E}(X^1|B)$. The dashed line is for estimating $\mathbb{E}(X^2|B)$ and the solid line is for estimating $\mathbb{E}(X^3|B)$.

Let A be an event such that there are 3 targets at time $t = T$ and X^k be the state of the k^{th} target at time $t = T$. Let B be an event such that $B \subset A$ and $\|X^1\| \leq \|X^2\| \leq \|X^3\|$. Our objective is the computation of $\mathbb{E}(X^k|B)$ for $k = 1, 2, 3$. Here, B provides a unique labeling described at the end of Section 3.2. $\mathbb{E}(X^k|B)$ are computed exactly by enumeration. Let $\hat{X}^k(n)$ be the estimate of $\mathbb{E}(X^k|B)$ made at the n^{th} MCMC step of Algorithm 7 after the first 100 burn-in samples. We ran 100 independent runs and computed the average estimation error for each k and n. The average estimation error is shown as a function of the number of samples in Figure 4.11(b). Although the size of the partition space Ω is over 45,000, this is a simple case and, as one might expect, Figure 4.11(b) shows rapid convergence of MCMCDA estimates toward the exact values. It is important to notice that MCMCDA only keeps a single association event (the current state) and does not enumerate all possible association events, hence, the memory requirement is significantly less than other approaches.

4.4 Summary

In this chapter, we have presented Markov chain Monte Carlo data association (MCMCDA) for solving data association problems arising in multi-target tracking in a cluttered environment. For the case of a fixed number of targets, we have shown that a single-scan MCMCDA algorithm provides a fully polynomial randomized approximation scheme for the single-scan Bayesian filter such as JPDA, which is known to be NP-hard. For a large problem, the exact calculation of JPDA by enumeration is not feasible but MCMCDA can efficiently find an approximate solution. In practice, an approach which combines both JPDA and MCMCDA takes advantages of both approaches. The combined approach can use JPDA for cases with a small number of edges in the measurement validation graph but use MCMCDA for cases with a large number of edges in the graph. The precise thresholding number of edges needs to be determined based on the specific application and available computing resource.

For the general multi-target tracking problem, in which an unknown number of targets appears and disappears at random times, we have presented a multi-scan MCMCDA algorithm that is capable of initiating and terminating an unknown number of tracks. The MCMCDA algorithm is flexible and can easily incorporate any domain specific knowledge to make it more efficient. Instead of enumerating the entire space of associations, MCMCDA randomly samples the region where the posterior is concentrated. Our simulation results show the remarkable performance of the MCMCDA algorithm under extreme conditions such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates. MCMCDA was also robust against the uncertainty in the number of targets. We have also shown that the algorithm can be formulated as an online, real-time algorithm with excellent performance. In Chapter 5 and 6, MCMCDA has been successfully extended hierarchically to improve its scalability.

Chapter 5

Distributed Multi-target Tracking and Identity Management

This chapter extends MCMCDA developed in Chapter 4 to distributed tracking for sensor networks. Our distributed tracking algorithm is inspired by the distributed tracking algorithm based on MHT by Chong *et al.* [Chong *et al.*, 1990]. In [Chong *et al.*, 1990], the tracking task is distributed and tracks are hierarchically merged. We follow the similar approach but our algorithm is based on a more computationally efficient multi-target tracking algorithm.

The tracks estimated by a multi-target tracking algorithm are usually used by other applications. These applications make decisions or reallocate resources based on the estimated tracks, *e.g.*, pursuer assignment and path planning in pursuit-evasion games [Schenato *et al.*, 2005]. But a decision based on a single set of tracks may be risky since tracks do not fully exhibit the uncertainty in the identities of targets accumulated from continuous interactions among crossing or nearby targets. For example, when two targets are moving close to each other, there can be multiple interpretations of the event. Figure 5.1(a) shows measurements about positions of two targets over time. Figure 5.1(b) and 5.1(c) shows two possible interpretations made from the measurements shown in Figure 5.1(a). Clearly, there can be more than two interpretations due to measurement noise and identity uncertainty but a multi-target tracking algorithm can only display a single interpretation (usually the one with the highest likelihood or an interpretation with expected positions). Since the number of possible interpretations grows exponentially as more measurements are collected, we can not display all possible interpretations. In this chapter, we assume that the communication medium is bandwidth-limited, hence, it is necessary to represent the uncertainty about the identities in the most compact representation.

This issue can be addressed by the identity management [Shin *et al.*, 2003; Hwang *et al.*, 2004b]. An identity is assigned to a target when it first appears; the *identity belief* associated with a target at any future point in time is represented as a probability distribution of the identity of the target over all existing identities. Thus, when two targets cross each other, the uncertainty in this crossing is represented by changes in the identity beliefs. However, the available identity management algorithms [Shin *et al.*, 2003; Hwang *et al.*, 2004b] work for the cases in which the number of targets in a sensor network is known and constant and their trajectories are available to local sensors. As a result, the existing algorithms are applicable to limited situations and difficult to scale them for a large sensor network. Hence, to handle general situations arising in a large-scale sensor network, a scalable and autonomous approach is required and it demands a new identity management system which can handle an unknown and time-varying number of targets.

Although target tracking and identity management are closely related, only the recent work by Hwang *et al.* [Hwang *et al.*, 2004a; Hwang *et al.*, 2006] describes an algorithmic framework for systematically tracking multiple maneuvering targets and maintaining their identities from noisy measurements. The algorithm has been shown to perform well for the case in which there is a single sensor (radar) and the number of targets are known and

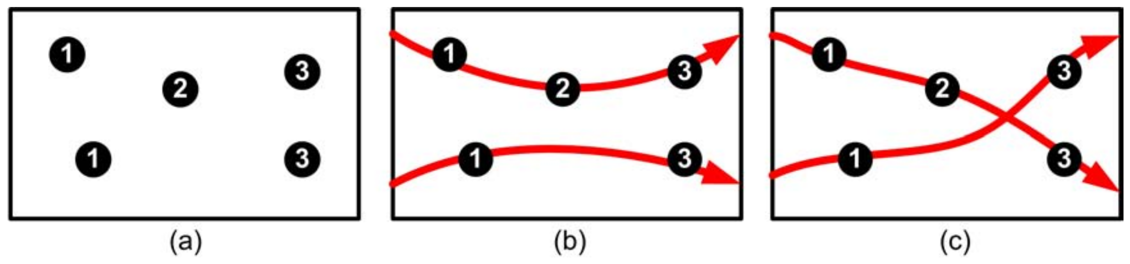


Figure 5.1: (a) Measurements about the positions of two targets (each circle represents a measurement and numbers represent measurement times); (b) One interpretation made from measurements shown in (a) (a solid line represents a track of a target); (c) Another interpretation.

constant. Thus, this algorithm cannot be used for sensor network applications in which there are many sensors and the number of targets is varying over time. We extend the results in [Hwang *et al.*, 2006] and propose a multi-target tracking and identity management system which is scalable, autonomous and distributed. The system is based on three new algorithms: multi-target tracking, identity management, and identity and track fusion algorithms. The multi-target tracking algorithm is a hierarchical implementation of MCMCDA described in Chapter 4 with a sliding window. MCMCDA is suitable for distributed sensor networks since it can autonomously initiate and terminate tracks. Also, it has been shown that MCMCDA is robust against packet losses and communication delays [Oh *et al.*, 2005].

For identity management and fusion algorithms, distributed multi-target identity management (DMIM) is proposed. DMIM is a scalable, event-driven, query-based algorithm for local maintenance of identity beliefs and the incorporation of identity information from nearby sensors. Global identity estimates are generated in DMIM using identity fusion, which is posed as an optimization problem such that the fused identity minimizes a cost function that represents a performance criterion. The DMIM algorithm combines local maintenance of identity beliefs with a query-based protocol for the transfer and fusion of identity information between sensors. The algorithm is appropriate for distributed sensor network scenarios, because it has the capability to reduce the uncertainty of the global identity estimates by fusing local estimates of the identity of a target collected by each sensor [Hwang *et al.*, 2004b]. The *identity-mass-flow* framework is used to maintain a local estimate of identity for a fixed set of maneuvering targets [Shin *et al.*, 2003; Hwang *et al.*, 2004b]. This framework prevents exponential growth in computation and storage of target-track association probabilities. We develop a distributed version of this centralized algorithm (DMIM) that allows an unknown, time-varying number of maneuvering targets in a sensor network. The DMIM algorithm is scalable with respect to number of targets, number of sensors, and can handle dynamic scenarios in which targets maneuver into and through the sensor network. A key component of DMIM is the fusion of tracks and identity beliefs between neighboring sensors. The tracks estimated by neighboring sensors are hierarchically merged using MCMCDA to maintain local consistency. Identity fusion in DMIM is based on principles of information theory. Information theoretic methods were initially developed to understand data communication and storage limits [Shannon, 1948] and have subsequently been applied to problems such as target localization [Wang *et al.*, 2004] and fault detection [Joshi *et al.*, 2005]. The identity fusion algorithm incorporates metrics from information theory as performance criteria when determining global belief estimates. Specifically, Shannon information, Chernoff information, and sum of Kullback-Leibler distances are used as cost functions. Minimizing these cost functions allow us to find locally consistent identity beliefs. Because local estimates can be combined under a query-based protocol, the identity fusion algorithm lends itself to a distributed sensor network in which targets maneuver in and out of the sensing range of individual sensors.

In this chapter, a distributed air traffic control system is used as an example of sensor networks. Although each air traffic controller is usually more capable than low-power sensor nodes described in [Estrin *et al.*, 2001; Culler *et al.*, 2004], the efficiency of the system described in this chapter makes the system applicable for a wide range of applications, including low-power or heterogeneous sensor networks.

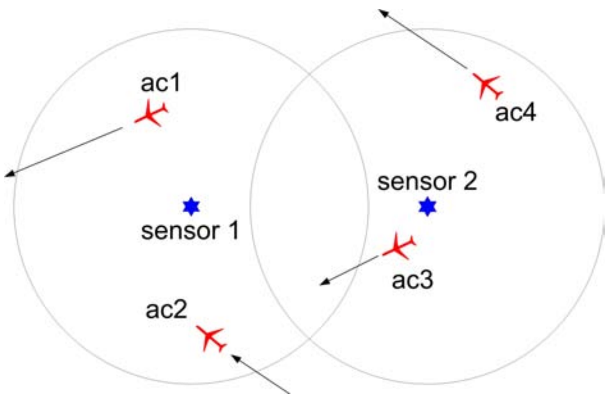


Figure 5.2: A distributed multi-target tracking and identity management scenario for a two-sensor network.

5.1 A System Architecture of Distributed Multi-target Tracking and Identity Management

The main focus of this chapter is the problem of tracking multiple targets and managing their identities in sensor networks. Each sensor is assumed to have its own surveillance region, and an ability to communicate with its neighboring sensors. A simple two-sensor example is shown in Figure 5.2 in which the circles represent the surveillance regions of the sensors. Each sensor is assumed to have the capability of tracking multiple targets and managing the identities of targets within its surveillance region. The problem gets complicated since the number of targets within the surveillance region of a sensor changes over time. For example, some targets may come from the surveillance regions of neighboring sensors, some targets may have not yet been registered into the identity management system, and some targets may leave the surveillance region of the current sensor. For a large network, a centralized approach to multi-target tracking and identity management is not feasible and a scalable distributed approach is required.

This chapter proposes a scalable distributed multi-target tracking and identity management (DMTIM) system that can track an unknown and time-varying number of maneuvering targets and manage their identities efficiently in a distributed sensor network. The key highlights of DMTIM include modularity, the compact representation of identity and tracks to reduce communication load, and event-driven mechanisms for identity and track management.

The identity of a target is maintained by a *belief vector*. When there are K known identities, the belief vector of the target at time t is $b(t) \in [0,1]^K$. The i-th element $b_i(t)$ of $b(t)$ represents the probability that the identity of the target is the i-th identity and $\sum_{i=1}^{K} b(t) = 1$. For multiple targets, we have a *belief matrix* $B(t)$ whose columns are belief vectors of the targets. Thus, entry $B_{ij}(t)$ represents the probability that target j can be identified as label (or name) i at time t.

The overall architecture of DMTIM is shown in Figure 5.3. DMTIM includes the *Data Association and Multi-target Tracking* (DAMTT) module and the *Distributed Multi-target Identity Management* (DMIM) module which contains the *Identity Management* (IM) and *Identity and Track Fusion* (ITF) sub-modules. At each sensor, the DAMTT module estimates the number of targets and tracks in its surveillance region using its local measurements and computes a mixing matrix and local information which are used by the IM module. A mixing matrix stores information about the interactions among targets while local information contains identity information about a target. A mixing matrix and local information are described in detail in Section 5.2. Upon receiving a mixing matrix and local information, the IM module updates its belief matrix. Then the sensor transmits its updated belief matrix and estimated tracks to its neighboring sensors using the *Communication* unit. In DMTIM, instead of sharing raw measurement data among neighboring sensors, the neighboring sensors share identity information in the form of a belief matrix and estimated tracks. As a result, we can reduce the overall communication load of the network. When the updated belief matrices and estimated tracks from neighboring sensors are available, the ITF module combines identity information and tracks, and maintains local consistency among sensors. For example, if the same target is seen by sensor 1 and

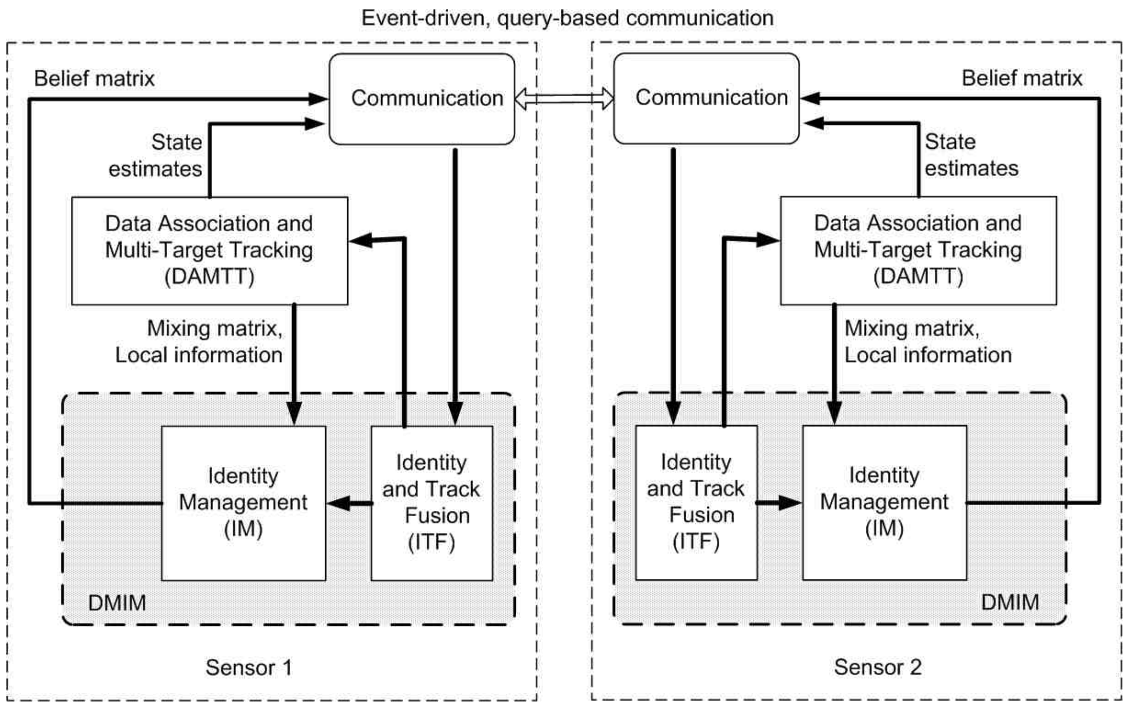

Event-driven, query-based communication
Belief matrix
Communication
State estimates
Data Association and Multi-Target Tracking (DAMTT)
Mixing matrix, Local information
Identity Management (IM)
Identity and Track Fusion (ITF)
DMIM
Sensor 1
Communication
Belief matrix
State estimates
Data Association and Multi-Target Tracking (DAMTT)
Mixing matrix, Local information
Identity and Track Fusion (ITF)
Identity Management (IM)
DMIM
Sensor 2

Figure 5.3: An architecture of a distributed multi-target tracking and identity management (DMTIM) system for a two-sensor example.

sensor 2, then the ITF module makes sure that sensor 1 and sensor 2 share the same information about the target. The ITF module also combines multiple tracks of a same target into a single track and add an entry for the identity of a new target into the belief matrix. In following sections, the components of DMTIM are described in detail.

5.2 Data Association and Multi-target Tracking (DAMTT)

The *Data Association and Multi-target Tracking* (DAMTT) module of DMTIM takes in sensor measurements and outputs a mixing matrix, state estimates, and local information. The DAMTT module needs to be able to track an unknown number of targets in order to distribute tracking tasks, since the number of targets in each sensor's surveillance region changes over time. This nonparametric estimation problem is efficiently solved by the online MCMCDA algorithm (Algorithm 8). In this section, we describe the computations of mixing matrices and local information.

5.2.1 Mixing Matrix

Suppose there are K targets and K identities in the surveillance region of the current sensor. Then, the problem of managing identities of multiple targets is to match each target to its identity over time. For this, we use the idea of the *identity-mass-flow* [Shin *et al.*, 2003]. The idea of the *identity-mass-flow* is that an identity is treated as a unit mass assigned to a target. These masses cannot be destroyed or created, and flow from a target into another through a *mixing matrix*, $M(t)$ at time t. A mixing matrix is a $K \times K$ matrix whose element $M_{ij}(t)$ represents the probability that target i at time $t-1$ has become target j at time t. Thus, a mixing matrix stores information about the interactions among targets for a single sampling period. It is a doubly stochastic matrix; that is, its column sums and row sums are equal to 1.

Let $K'(t)$ be the number of targets estimated by the online MCMCDA at time t. We are interested in computing a mixing matrix for the targets that are present both at time $t-1$ and t. Let K be the number of targets present at time $t-1$ and t, *i.e.*, $K = \min(K'(t-1), K'(t))$ and this excludes any disappeared targets or new targets. Without loss of generality, assume that the first K targets are present at time $t-1$ and t. Let $\hat{x}(t-1) = \{\hat{x}^k(t-1) : 1 \leq k \leq K\}$ be the state estimates of targets at time $t-1$ and $\hat{x}(t) = \{\hat{x}^k(t) : 1 \leq k \leq K\}$ be the current state estimates computed by online MCMCDA. The mixing matrix entry $M_{ij}(t)$ represents the probability that the target with state $\hat{x}^i(t-1)$ at time $t-1$ has become the target with state $\hat{x}^j(t)$ at time t. We use an algorithm similar to Algorithm 6 to estimate the mixing matrix to reduce computation time. We first encode this target-to-target association event in a bipartite graph. Let $G = (U, V, E)$ be a bipartite graph, where $U = \{1, \ldots, K\}$ is a set of target indices at time $t-1$, $V = \{1, \ldots, K\}$ is a set of target indices at time t, and

$E = \{(i,j) : i \in U, j \in V, P(\hat{x}^j(t)|\hat{x}^i(t-1)) > \epsilon_0\}$ for some small $\epsilon_0 > 0$. A feasible target-to-target association event ϕ is a *matching* in G, *i.e.*, a subset $E' \subset E$ such that no two edges in E' share a vertex. Let $\Phi = \{\phi\}$ be the set of all feasible target-to-target association events. The posterior of ϕ given $\hat{x}(t)$ and $\hat{x}(t-1)$ can be computed using the Bayes rule:

$$\begin{aligned} P\left(\phi|\hat{x}(t),\hat{x}(t-1)\right) &= \frac{1}{Z} P\left(\hat{x}(t)|\hat{x}(t-1),\phi\right) P\left(\hat{x}(t-1)|\phi\right) P(\phi) \\ &= \frac{1}{Z}\left(\prod_{(i,j)\in\phi} P\left(\hat{x}^j(t)|\hat{x}^i(t-1)\right)\right) P\left(\hat{x}(t-1)|\phi\right) P(\phi), \end{aligned} \tag{5.1}$$

where Z is a normalizing constant. We assume that matchings of the same size are equally likely, hence, $P(\phi) \propto \binom{|E|}{|\phi|}^{-1} \propto |\phi|!(|E|-|\phi|)!$, where $|E|$ is the number of edges in E and $|\phi|$ is the number of matches in ϕ. When computing a mixing matrix, we assume $\{\hat{x}^i(t-1)\}$ are fixed for the targets with known identities. So we simply set $P(\hat{x}(t-1)|\phi) = 1$.

Under these assumptions, the posterior (5.1) reduces to

$$P\left(\phi|\hat{x}(t),\hat{x}(t-1)\right) = \frac{1}{Z'}\left(\prod_{(i,j)\in\phi} P\left(\hat{x}^j(t)|\hat{x}^i(t-1)\right)\right) |\phi|!(|E|-|\phi|)!, \tag{5.2}$$

where Z' is another normalizing constant. Each $P(\hat{x}^j(t)|\hat{x}^i(t-1))$ can be computed using the target's dynamic model, hence, the posterior (5.2) can be computed up to a normalizing constant. Notice that $P(\hat{x}^j(t)|\hat{x}^i(t-1))$ can be computed exactly for linear-Gaussian dynamic models and approximately for the cases when the dynamic model is nonlinear and/or the noise processes are non-Gaussian using methods such as linearization, unscented filtering [Julier and Uhlmann, 2004], interacting multiple model [Blom and Bar-Shalom, 1988], and particle filters [de Freitas and Gordon, 2001]. Hence, our objective is to compute the mixing matrix based on our estimates $\{P(\hat{x}^j(t)|\hat{x}^i(t-1))\}$.

Let ϕ_{ij} be the event that the target with state $\hat{x}^i(t-1)$ becomes the target with state

$\hat{x}^j(t)$. Now the mixing probability $M_{ij}(t)$ can be computed as:

$$M_{ij}(t) = P\left(\phi_{ij}|\hat{x}(t), \hat{x}(t-1)\right) = \sum_{\phi \in \Phi \,:\, (i,j) \in \phi} P\left(\phi|\hat{x}(t), \hat{x}(t-1)\right). \tag{5.3}$$

The computation of $M_{ij}(t)$ requires a summation over the posteriors, hence the enumeration of all joint association events. The exact computation of a mixing matrix is NP-hard. More generally, the exact computation of association probabilities in JPDA is NP-hard [Collins and Uhlmann, 1992] since the related problem of finding the permanent of a 0-1 matrix is #P-complete [Valiant, 1979]. Hence, for a large problem, *i.e.*, when the number of targets is large, we need to seek for an efficient approximation algorithm. In the remainder of this section, we describe a polynomial-time approximation algorithm based on MCMC, namely the Metropolis-Hastings algorithm. The algorithm for computing a mixing matrix is shown in Algorithm 9. The inputs to Algorithm 9 are the graph G, state estimates $\hat{x}(t)$ and $\hat{x}(t-1)$, an initial Markov chain state ϕ_{init} which is chosen randomly from Φ, the number of MCMC samples n_{mc}, and the number of burn-in MCMC samples n_{bi}. The mixing matrix is estimated by the Monte Carlo integration of samples after the burn-in period. For more information about burn-in samples and general MCMC techniques, we refer readers to [Gilks *et al.*, 1996]. Notice that we only need to compute the ratio $P(\phi'|\hat{x}(t), \hat{x}(t-1))/P(\phi|\hat{x}(t), \hat{x}(t-1))$, avoiding the need to normalize $P(\phi|\hat{x}(t), \hat{x}(t-1))$.

While heuristic approaches do not guarantee asymptotic optimality and may fail in some situations, Algorithm 9 can approximate the mixing matrix $M = [M_{ij}]$ (the time index is suppressed) in polynomial time with guaranteed error bounds.

Theorem 8. For any $0 < \epsilon_1, \epsilon_2 < 1$ and $0 < \eta < .5$, with time complexity

$$O(\epsilon_0^{-4}\epsilon_1^{-2}\epsilon_2^{-1} \log \eta^{-1} K^8 (K \log \epsilon_0^{-1} + \log(\epsilon_1^{-1}\epsilon_2^{-1}))), \tag{5.4}$$

Algorithm 9 MCMC for Mixing Matrix Computation

```
Input: $n_{\text{bi}}, n_{\text{mc}}, G = (U, V, E), \hat{x}(t), \hat{x}(t-1), \phi_{\text{init}}$
Output: $\hat{M}(t)$
  $\phi = \phi_{\text{init}}$; $\hat{M}(t) = \mathbf{0}^{K\times K}$
  for $n = 1$ to $n_{\text{mc}}$ do
    choose $e = (u, v) \in E$ uniformly at random
    if $e \in \phi$ then
      $\phi' = \phi - e$
    else if both $u$ and $v$ are unmatched in $\phi$ then
      $\phi' = \phi + e$
    else if exactly one of $u$ and $v$ is matched in $\phi$ and $e'$ is the matching edge then
      $\phi' = \phi + e - e'$
    else
      $\phi' = \phi$
    end if
    $\phi = \phi'$ with probability $\min\left(1, \frac{P(\phi'|\hat{x}(t),\hat{x}(t-1))}{P(\phi|\hat{x}(t),\hat{x}(t-1))}\right)$
    if $n > n_{\text{bi}}$ then
      for each $(i, j) \in \phi$ do
        $\hat{M}_{ij}(t) = \hat{M}_{ij}(t) + 1$;
      end for
    end if
  end for
  for each $(i, j) \in \{(i', j') : 1 \le i', j' \le K\}$ do
    $\hat{M}_{ij}(t) = \hat{M}_{ij}(t)/(n_{\text{mc}} - n_{\text{bi}})$;
  end for
```

Algorithm 9 finds estimates $\hat{M}_{ij}$ for M_{ij} with probability at least $1 - \eta$, such that, for $M_{ij} \ge \epsilon_2$, $\hat{M}_{ij}$ estimates M_{ij} within ratio $1 + \epsilon_1$, *i.e.*, $(1 - \epsilon_1)M_{ij} \le \hat{M}_{ij} \le (1 + \epsilon_1)M_{ij}$, and, for $M_{ij} < \epsilon_2$, $|\hat{M}_{ij} - M_{ij}| \le (1 + \epsilon_1)\epsilon_2$.

Theorem 8 is a corollary of Theorem 5 and Theorem 6 after replacing the number of measurements N in Theorem 5 with the number of targets K and letting $R = |E|/\epsilon_0$ and $m_5(K, N) = O(K \log \epsilon_0^{-1})$.

5.2.2 Local Information

In some applications, identity information about a target (local information) could be obtained from sensors which can measure its physical attributes or from the target's dynamic characteristics. When local information is available, we use local information to decrease the uncertainty of the belief matrix measured by entropy. MCMCDA described in Section 4.3 allows an efficient way to compute local information from both latest and past measurements. Another benefit of MCMCDA is that local information can be computed simultaneously while the number of targets and tracks of all targets are estimated. For identity k, let N_{jk} be the number of times the j-th latest measurement is associated with the initial measurement identified by k after the initial n_{bi} samples while running Algorithm 7, where n_{bi} is the number of initial burn-in samples and $n_{\text{bi}} < n_{\text{mc}}$. When Algorithm 7 terminates, we compute $\gamma^k = (\gamma_1^k, \ldots, \gamma_{n(t)}^k)^T$ for identity k, where $\gamma_j^k = N_{jk}/(n_{\text{mc}} - n_{\text{bi}})$. Then we form local information from γ^k by resizing the vector according to the latest measurements assigned in state estimates and normalizing the resized vector.

5.3 Distributed Multi-target Identity Management (DMIM)

5.3.1 Identity Management (IM)

The *Identity Management* (IM) module consists of *Belief Matrix Update* and *Local Information Incorporation*. In the IM module, a mixing matrix and local information from the DAMTT module are used to update the belief matrix.

5.3.1.1 Belief Matrix Update

The *Belief Matrix Update* block maintains identity information stored in a belief matrix $B(t)$ whose columns are belief vectors of the targets over time. The evolution of this belief matrix is governed by a mixing matrix $M(t)$, which stores interaction information for a

Algorithm 10 Event-driven, query-based Belief Matrix Update

```
• For sensor $i$ and target $k$

if target $k$ leaves the surveillance region of sensor $i$. then
    delete the corresponding column in the belief matrix.
end if
if a target enters the surveillance region of sensor $i$. then
    send a query about the identity of target $k$.
    if there is an answer “yes” and receive the belief vector of target $k$, then
        augment the belief matrix with the belief vector received.
    else
        augment the belief matrix with a belief vector with a new identity assigned to the
        target.
    end if
end if
```

single time step. Then, the belief matrix is updated by [Shin *et al.*, 2003]:

$$B(t) = B(t-1)M(t) \tag{5.5}$$

We can show that (5.5) keeps row and column sums of the belief matrix constant when the numbers of targets and identities are the same. However, this is not the case for distributed identity management since the number of the targets within the surveillance region of individual sensors may change over time. There are two possible cases: a target leaves or enters the surveillance region of a sensor. When a target leaves, we delete the corresponding column in the belief matrix managed by the sensor. When a target enters the surveillance region of a sensor, there are two possible cases: (i) the target comes from the surveillance region of another sensor, which may be queried, or (ii) the target comes from the outside of the surveillance region of a sensor network. For these cases, we propose Algorithm 10, a scalable, event-driven, query-based belief matrix update algorithm.

For distributed identity management, a belief matrix managed by each sensor may not be a square matrix but might more likely be a skinny matrix which has more rows than columns. The belief matrix may not be a doubly stochastic matrix whose row and column sums are equal to one, but it should be a stochastic matrix with column sums equal to one. Its row sums remain constant because an identity mass cannot be destroyed or created. Since the evolution of the belief matrix is governed by (5.5), these characteristics of the belief matrix are preserved over time.

5.3.1.2 Local Information Incorporation

When local information is available, we use local information to decrease the uncertainty of the belief matrix measured by entropy. The entropy (Shannon information) of an $L \times K$ belief matrix is defined as

$$H(B(t)) \triangleq -\sum_{i=1}^{L}\sum_{j=1}^{K} B_{ij}(t) \log B_{ij}(t). \tag{5.6}$$

Then, the problem is how to incorporate this information to the belief matrix. From the idea of the *identity-mass-flow* and the characteristics of (5.5), we know that the belief matrix should have the following properties: its column sums are equal to one and its row sums remain constant. However, if we replace a column in the belief matrix with local information, it is not guaranteed that the new belief matrix has the above properties. For a nonnegative square matrix, the Sinkhorn algorithm [Sinkhorn, 1967] can be used to scale a matrix to achieve specified row and column sums [Rothblum and Schneider, 1989; Linial *et al.*, 2000]; that is, we scale a new belief matrix so that its row and column sums remain the same as those of the belief matrix before local information incorporation. However, since the belief matrix is, in general, a nonnegative rectangular matrix, such iteration may not converge [Rothblum and Schneider, 1989; Linial *et al.*, 2000; Balakrishnan *et al.*, 2004]. But the question whether a given matrix is *almost* scalable, *i.e.*, the matrix can be scaled within $\epsilon > 0$, can be decided in polynomial time [Balakrishnan *et al.*, 2004]. Thus, we can efficiently check whether the available local information

Algorithm 11 Local Information Incorporation

- **Given:** local information (belief vector) of a target and a belief matrix $B(t)$.
- Make a matrix $B'(t)$ by replacing the column corresponding to the target in $B(t)$ with the local information.
- Operator S represents the matrix scaling process in [Balakrishnan *et al.*, 2004].

```
if $B'(t)$ is almost scalable then
  $B_{new}(t) := \mathsf{S}(B'(t))$
  if $H(B_{new}(t)) \leq H(B(t))$ then
    $B(t) := B_{new}(t)$
  else
    $B(t) := B(t)$
  end if
else
  $B(t) := B(t)$
end if
```

can be incorporated. Thus, local information can be incorporated when it makes the new belief matrix almost scalable. Even though the new belief matrix is almost scalable, local information incorporated may not necessarily decrease the uncertainty (entropy) of the belief matrix. Therefore, local information is incorporated only when it reduces the uncertainty of the belief matrix. The local information incorporation algorithm is described in Algorithm 11.

5.3.2 Identity and Track Fusion (ITF)

For DMTIM, the identity and track fusion is crucial to compute the global information of the system from information provided by local sensors. In this section, we explain how the *Identity and Track Fusion* (ITF) module combines state estimates and belief vectors of the same target from neighboring sensors.

5.3.2.1 Identity Fusion

We now consider the problem of combining two belief vectors of the same target from two different sensors. Identity fusion can be formulated as an optimization problem such that the fused identity is the one that minimizes a cost function which represents a performance criterion. For optimization, we propose three different cost functions: Shannon information, Chernoff information, and the sum of Kullback-Leibler distances. We then derive a Bayesian identity fusion method and discuss the relationship between the Bayesian method and the optimization algorithms.

Shannon information

The Shannon information is defined as

$$H(b') = \sum_{i=1}^{n} -b'(i) \log b'(i) \tag{5.7}$$

where $b' \in [0,1]^n$ with $\sum_i b'(i) = 1$. The Shannon information (also known as *entropy*) is a measure of the uncertainty of a system. Thus, the minimization of the Shannon information selects a belief vector that is most informative in the sense of minimum entropy. Suppose b_1 and b_2 are belief vectors of target t computed by sensor 1 and sensor 2 respectively. Since the most common data fusion algorithms compute a *linear combination* of two data, we propose the following fusion strategy:

$$b' = \lambda b_1 + (1-\lambda) b_2 \tag{5.8}$$

where $\lambda \in [0,1]$, $b_i \in [0,1]^n$ with $\sum_{j=1}^{n} b_i(j) = 1$ for $i \in \{1,2\}$, and $\sum_{j=1}^{n} b'(j) = 1$. Then, the problem of computing the fused belief vector becomes a problem of finding a weight, λ, which minimizes the cost function in (5.7). If we use the fusion strategy in (5.8), the Shannon information of the new fused belief vector is

$$H(b') = H(\lambda b_1 + (1-\lambda)b_2) \geq \lambda H(b_1) + (1-\lambda)H(b_2) \tag{5.9}$$

From (5.9), we can see that the minimum is always achieved at either $\lambda = 0$ or $\lambda = 1$. This means that a fused belief vector that has the minimum Shannon information is either of the two given belief vectors, which is a *hard* choice. For some applications such as identity management, the hard choice may not be desirable since it ignores one possibility completely and thus might quickly lead to a wrong answer over time if not immediately. Thus, we propose a *soft* decision method which has $\lambda \in (0, 1)$ for almost all cases. Motivated by the fact that Shannon information minimization chooses a belief vector which has the minimum entropy, we propose to use the inverse of the Shannon information of a belief vector as a weight. Thus, we put large confidence on a belief vector which has small Shannon information. Then, a new belief vector $b' = [b'(i)]$ is

$$b'(i) = \frac{H(b_1)^{-1} b_1(i)}{H(b_1)^{-1} + H(b_2)^{-1}} + \frac{H(b_2)^{-1} b_2(i)}{H(b_1)^{-1} + H(b_2)^{-1}} \tag{5.10}$$

From (5.8) and (5.10), we get

$$\lambda = \frac{H(b_1)^{-1}}{H(b_1)^{-1} + H(b_2)^{-1}} = \frac{H(b_2)}{H(b_1) + H(b_2)} \tag{5.11}$$

When $H(b_1) = H(b_2) = 0$, we set $\lambda = \frac{1}{2}$. $\lambda = 0$ if $H(b_2) = 0$ (no uncertainty in b_2) and $\lambda = 1$ when $H(b_1) = 0$ (no uncertainty in b_1). In these cases, the fused belief vector computed by the proposed fusion algorithm is a belief vector which has no uncertainty. This fusion algorithm is a soft decision method since the fused data is a convex combination of the two given data with a larger weight on the data which has smaller entropy than the other. From (5.9) and (5.11), the Shannon information of the new belief $H(b')$ has the property that:

$$H(b') \geq \frac{2H(b_1)H(b_1)}{H(b_1)+H(b_2)} \qquad \text{or} \qquad 2H(b')^{-1} \leq H(b_1)^{-1} + H(b_2)^{-1} \tag{5.12}$$

Inequality (5.12) tells us that the achievable minimum uncertainty of the fused belief vector with the fusion strategy in (5.8) with (5.11) as a weight is under-bounded by uncertainties of the given information. In other words, the maximum achievable certainty (inverse of the Shannon information) is upper-bounded by the arithmetic mean of the inverse of the Shannon information of the given belief vectors. If we use the fusion strategy in (5.8), we can also derive the upper bound of the Shannon information of the new belief vector:

$$\begin{aligned} H(b') &\leq \lambda^2 H(b_1) + (1-\lambda)^2 H(b_2) \\ &+ \lambda(1-\lambda)(H(b_1) + H(b_2) + D(b_1 \parallel b_2) + D(b_2 \parallel b_1)) \end{aligned} \tag{5.13}$$

where $D(p \parallel q) \triangleq \sum_i p(i)\log(\frac{p(i)}{q(i)})$ is the Kullback-Leibler distance [Cover and Thomas, 1991]. If we use λ in (5.11), then

$$H(b') \leq \frac{2H(b_1)H(b_1)}{H(b_1)+H(b_2)} + \frac{H(b_1)H(b_2)[D(b_1 \parallel b_2) + D(b_2 \parallel b_1)]}{(H(b_1)+H(b_2))^2} \tag{5.14}$$

Thus, we can analytically compute the upper and lower bounds of the Shannon information of the new belief vector using the fusion strategy in (5.8) with (5.11). If we interpret the error covariance matrix as an uncertainty measure of an estimate of a continuous random variable, we find that (5.12) could be interpreted as an analogy of the *Cramér-Rao lower bound* of the mean-squared error (error covariance) of any unbiased estimator in the sense that the achievable minimum uncertainty has a lower bound. Thus, the Shannon information cost function would be useful when we have good knowledge about the performance and/or fidelity of each sensor, since we can get a solution which has lower entropy by weighing information that has smaller entropy more than the other. However, if we do not have such knowledge, we may get a biased solution by consistently putting more confidence on one piece of information (possibly the wrong one) than the other.

Chernoff information

The Chernoff information is defined as

$$C(b_1, b_2) = -\min_{0 \leq \lambda \leq 1} \log\left(\sum_{i=1}^{n} b_1(i)^{\lambda} b_2(i)^{1-\lambda}\right) \tag{5.15}$$

If λ^* minimizes (5.15), the new belief vector b' $(= [b'(i)]$ for $i = \{1, 2, \cdots, n\})$ is

$$b'(i) = \frac{b_1(i)^{\lambda^*} b_2(i)^{1-\lambda^*}}{\sum_{j=1}^{n} b_1(j)^{\lambda^*} b_2(j)^{1-\lambda^*}} \tag{5.16}$$

The new belief vector in (5.16) satisfies [Cover and Thomas, 1991; Kapur and Kesavan, 1992]

$$D(b' \parallel b_1) = D(b' \parallel b_2) \tag{5.17}$$

This fusion strategy is different from that in (5.8) which is a convex combination of the two data. From (5.17), the minimization of the Chernoff information is equivalent to finding a function that is in the *middle* of the two original functions, where the middle is defined in terms of the Kullback-Leibler distance. In other words, Chernoff information minimization could be interpreted as selecting a probability vector which is “equally close” in terms of the Kullback-Leibler distance to the original probability vectors. This fusion algorithm does not put more confidence on one than the other. Thus, this cost function could be useful when we do not know the quality of information obtained from individual sensors; by choosing the middle point of the two pieces of information, we could minimize the bias over time. However, the fused belief vector computed by the Chernoff information minimization algorithm may have larger entropy than that computed by the algorithm in (5.8) with (5.11).

Sum of the Kullback-Leibler distances

Since the Kullback-Leibler distance is not symmetric, we consider two possible optimization problems:

$$\begin{array}{ll} \text{minimize} & D(b' \parallel b_1) + D(b' \parallel b_2) \\ \text{subject to} & \sum_{j=1}^{n} b'(j) = 1 \\ & b'(j) \geq 0 \end{array} \tag{5.18}$$

$$\begin{array}{ll} \text{minimize} & D(b_1 \parallel b') + D(b_2 \parallel b') \\ \text{subject to} & \sum_{j=1}^{n} b'(j) = 1 \\ & b'(j) \geq 0 \end{array} \tag{5.19}$$

where $b'(j)$ is the j-th element of a vector b'. Let us first consider the optimization problem in (5.18). The Lagrangian is given by

$$L(b', \lambda) = D(b' \parallel b_1) + D(b' \parallel b_2) + \lambda(\sum_{j=1}^{n} b'(j) - 1) \tag{5.20}$$

To get an optimal solution, we set the derivatives of L with respect to $b'(i)$ and to λ to be equal to zero. Then, we get a new belief vector:

$$b'(i) = \frac{\sqrt{b_1(i)b_2(i)}}{\sum_{j=1}^{n} \sqrt{b_1(j)b_2(j)}} \tag{5.21}$$

From (5.21), we see that the fused data is a geometric mean of the given data. The fused data is the same as that in (5.16) for Chernoff information minimization when $\lambda^* = \frac{1}{2}$. Thus, this data fusion strategy can be interpreted as a special case of the Chernoff information minimization method.

Now, let us consider the optimization problem in (5.19). The Lagrangian is given by

$$L(b', \lambda) = D(b_1 \parallel b') + D(b_2 \parallel b') + \lambda(\sum_{j=1}^{n} b'(j) - 1) \tag{5.22}$$

Similarly, we get an optimal solution:

$$b'(i) = \frac{b_1(i) + b_2(i)}{\sum_{j=1}^{n}[b_1(j) + b_2(j)]} = \frac{b_1(i) + b_2(i)}{2} \tag{5.23}$$

In this case, the fused data is the arithmetic mean of the given data. This fusion strategy is the same as that in (5.8) when $\lambda = \frac{1}{2}$. Thus, from (5.9) and (5.13), we get the lower and upper bounds of Shannon information of the new belief vector:

$$\begin{aligned} H(b') &\geq \tfrac{H(b_1)+H(b_2)}{2} \\ H(b') &\leq \tfrac{H(b_1)+H(b_2)}{2} + \tfrac{D(b_1\|b_2)+D(b_2\|b_1)}{4} \end{aligned} \tag{5.24}$$

Therefore, the fusion algorithms obtained by solving the optimization problems in (5.18) or (5.19) are to average the given data either geometrically or arithmetically. This is similar to Chernoff information minimization and thus these fusion strategies would be useful when we want to get unbiased fused data in situations where we do not have good *a prior* information about a system. An example would be a case in which information from one sensor is wrong due to failure of the sensor or the malicious intent of the sensor that is unknown *a priori*. These information fusion strategies would be robust to this wrong information since they do not put more confidence on one (possibly incorrect information) than the other, but average them to compute a fused belief vector.

Bayesian approach

In this section, we derive a fused belief vector using a Bayesian approach. Suppose the target's identity $\theta \in \{1, 2, \cdots, N\}$ and, without loss of generality, suppose there are two sensors. Denote events X_1 and X_2 to be measurements at sensor 1 and sensor 2, respectively. We are assumed to be given information $b_1(\theta) \triangleq P(\theta|X_1)$ from sensor 1 and $b_2(\theta) \triangleq P(\theta|X_2)$ from sensor 2 where $P(\cdot|\cdot)$ is a conditional probability. Then, the problem of identity fusion is to find the *a posteriori* probability $P(\theta|X_1, X_2)$. We assume $P(X_1, X_2|\theta) = P(X_1|\theta)P(X_2|\theta)$ since given the identity of a target, the events that it is observed by sensor 1 or sensor 2 are independent in distributed identity management. Using the Bayes rule, we get

$$P(\theta|X_1, X_2) = \frac{P(X_1|\theta)P(X_2|\theta)P(\theta)}{P(X_1, X_2)} \tag{5.25}$$

Since $P(\theta|X_i) = \frac{P(X_i|\theta)P(\theta)}{P(X_i)}$ for $i \in \{1, 2\}$, we obtain

$$P(\theta|X_1, X_2) = \frac{b_1(\theta)b_2(\theta)}{P(\theta)} \frac{P(X_1)P(X_2)}{P(X_1, X_2)} \tag{5.26}$$

Therefore, a fused belief vector is

$$b'(\hat{\theta}) = \arg\max_{\theta} P(\theta|X_1, X_2) = \frac{b_1(\theta)b_2(\theta)}{P(\theta)} \cdot \frac{1}{c} \tag{5.27}$$

where c is a normalization constant. This is an interesting result because the fused data does depend only on the given data ($b_1(\theta)$, $b_2(\theta)$) and the *a priori* probability $P(\theta)$. Thus, if we knew *a priori* information, we could compute the *a posteriori* probability (*i.e.*, the fused data). However, since we may not know the *a priori* probability for some applications such as distributed identity management, we cannot compute the fused data from (5.27). In order to compute the fused data for this case, we have to assume $P(\theta)$ either from the characteristics of the systems or from that of applications. For example, if we assume the *a priori* probability as a geometric mean of the given data due to lack of information about a system $(P(\theta) = \sqrt{b_1(\theta)b_2(\theta)})$, then we can get exactly the same result as that in (5.21) which minimizes the sum of Kullback-Leibler distances to the original data in (5.18). Thus, the *a posteriori* probability is the same as the *a priori* probability; that is, we cannot extract any information from the given data. From Bayesian analysis, we can see that the data fusion strategies such as Chernoff information minimization and the minimization of the sum of Kullback-Leibler distances in (5.18) compute the solution in a similar form to the solution produced by the Bayesian approach.

We consider the case in which good knowledge about the performance/fidelity of all sensors is available; hence, the Shannon information method is used. For a comprehensive comparison of identity fusion methods, see [Hwang *et al.*, 2004b].

5.3.2.2 Track Fusion

Since each sensor maintains its own set of tracks, there can be multiple tracks from the same target maintained by different sensors. In order to resolve this inconsistency, we perform the multi-track fusion also described in Section 6.3.2.2 to combine tracks from different sensors. Let ω_i be the set of tracks maintained by sensor i and NB_i be a set of neighboring sensors around i, including i itself. Let $Y'_{\text{w}} = \{\tau_k(t) : \tau_k \in \omega_j, 1 \leq t \leq T, 1 \leq k \leq |\omega_j|, j \in \text{NB}_i\}$ be a set of measurements of all identified targets. We form a set of combined measurements Y_{w} from Y'_{w} by combining measurements made from overlapping surveillance regions and keeping the remaining measurements[1]. We then form a new set of tracks ω_{init} from $\{\tau \in \omega_j : j \in \text{NB}_i\}$ while making sure that constraints defined in Chapter 3 are satisfied. Then we run Algorithm 7 on the set of combined measurements Y_{w} with the initial state ω_{init} to find locally consistent tracks.

[1]In our current implementation, when multiple measurements from different sensors are in close proximity (measured by the Mahalanobis distance using the measurement covariance matrix), their mean value is used in Y_{w} instead. In future implementation, we plan to allow multiple measurements over the overlapping regions by relaxing constraints in Chapter 3.

5.4 Simulation Results

In this section, we present two sets of simulations to illustrate the features of the DMTIM algorithm. There are stationary sensors, *e.g.*, air traffic control radars, tracking multiple aircraft through two-dimensional space. The sensing range of each sensor is assumed to be circular with a radius of 10 km and a pair of sensors can communicate if they are within the communication radius of 20 km. We first present a two-sensor scenario with three aircraft and two sensors and describe the behavior of DMTIM in detail. Then we describe an example with five aircraft and seven sensors to demonstrate a complete DMTIM system. Moderate size examples are chosen for better illustration.

The nonlinear dynamics of maneuvering aircraft is modeled using interacting multiple model [Blom and Bar-Shalom, 1988] with two linear kinematic models based on the discrete-time linear dynamics (4.19):

(Model 1) Second-order kinematic model:

$$A(\nu^k(t)=1) = \begin{bmatrix} 1 & \delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G(\nu^k(t)=1) = \begin{bmatrix} \delta^2/2 & 0 \\ \delta & 0 \\ 0 & \delta^2/2 \\ 0 & \delta \end{bmatrix}$$

and $Q(\nu_t^k = 1) = \text{diag}(0.1, 0.1)$, where $\nu^k(t)$ indexes the kinematic model of target k operating at time t and δ is the sampling period. The state vector is $x = (x_1, \dot{x}_1, x_2, \dot{x}_2)^T$. This model assumes that the variation in a velocity component is a discrete time white noise acceleration [Lerro and Bar-Shalom, 1993].

(Model 2) Third-order kinematic model:

$$A(\nu^k(t)=2) = \begin{bmatrix} 1 & \delta & \delta^2/2 & 0 & 0 & 0 \\ 0 & 1 & \delta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \delta & \delta^2/2 \\ 0 & 0 & 0 & 0 & 1 & \delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad G(\nu^k(t)=2) = \begin{bmatrix} \delta^2/2 & 0 \\ \delta & 0 \\ 1 & 0 \\ 0 & \delta^2/2 \\ 0 & \delta \\ 0 & 1 \end{bmatrix}$$

and $Q(\nu_t^k=2)=\text{diag}(20,20)$. The state vector is $x=(x_1,\dot{x}_1,\ddot{x}_1,x_2,\dot{x}_2,\ddot{x}_2)^T$. This is a third-order kinematic model with accelerations modeled as a discrete time Wiener process [Lerro and Bar-Shalom, 1993].

The measurement model given in (4.19) is used for both models where $R=\text{diag}(200,200)$. The other parameters are: $p_\text{d}=0.98$, $\lambda_\text{f}=1.0\times 10^{-7}$, and $\delta=5$.

5.4.1 Two-sensor Scenario

A simple, yet illustrative, scenario with three aircraft is shown in Figure 5.4(a). The aircraft labeled A and B are previously registered and aircraft labeled X is unknown to the identity management system. The sensor on the left is denoted by sensor 1 and the sensor on the right is denoted by sensor 2.

The DAMTT module of each sensor estimates the number of targets (Figure 5.4(b)) and estimates tracks of targets as shown in Figure 5.4(c) and 5.4(d). In Figure 5.4(b), the events, in which the number of targets changes, are indicated by dotted vertical lines. The belief vector for each target, *i.e.*, a column of the belief matrix, computed by the *Identity Management* (IM) module is shown in Figure 5.5(a) and 5.5(b). At time 1, sensor 1 knows about target 1 and its belief vector is $(b_{A,1}^1, b_{B,1}^1)^T=(0.8,0.2)^T$, where $b_{j,k}^i$ is the probability that target k of sensor i can be identified as label j, while sensor 2 knows about its target

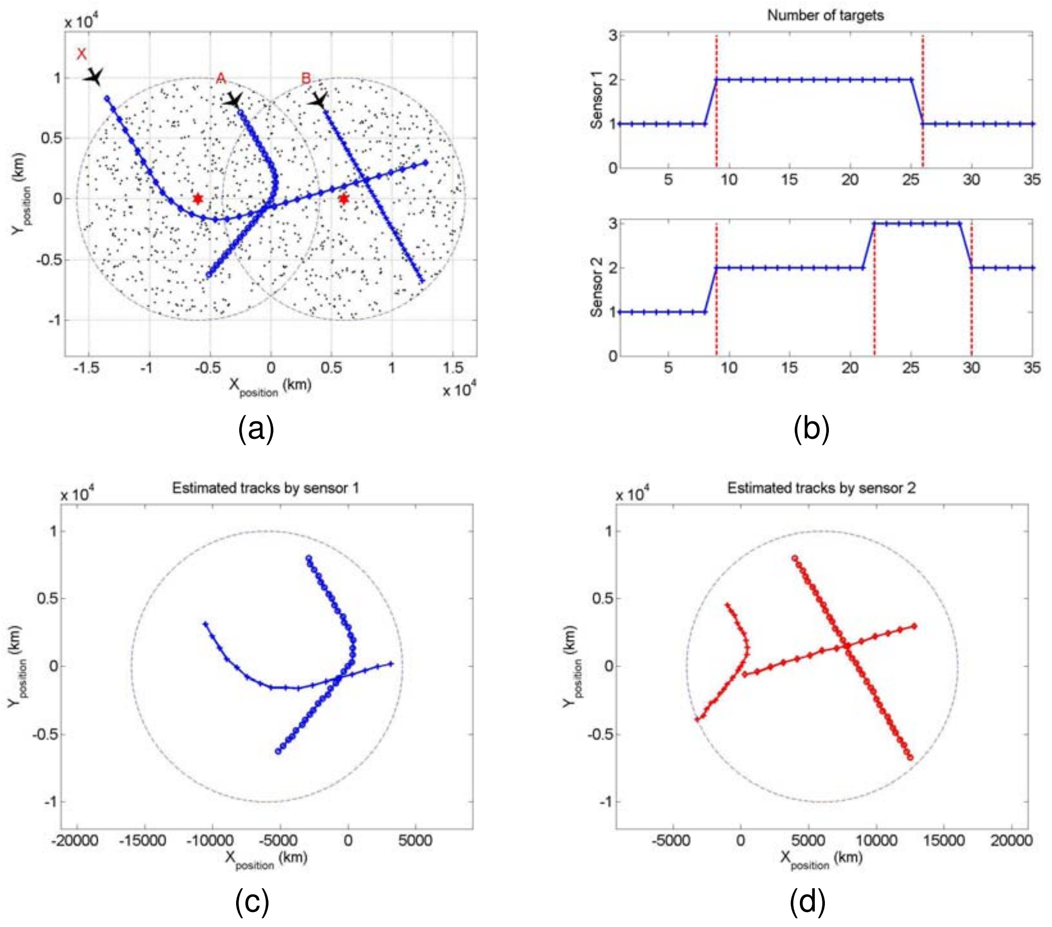
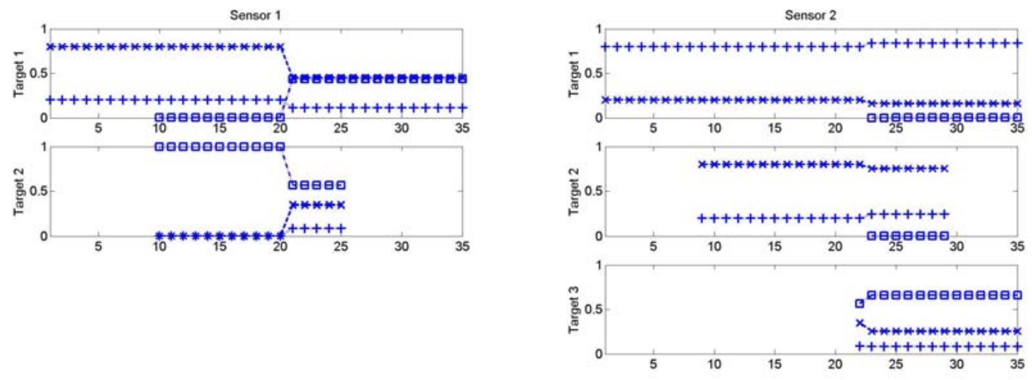
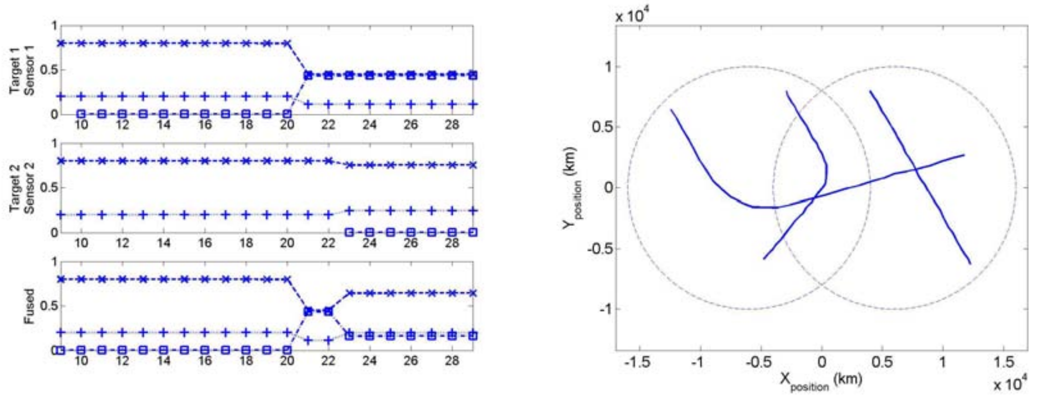


Figure 5.4: (a) Aircraft trajectories for three-aircraft , superimposed with accumulated measurements (dots). The sensor positions are marked by $\star$. (b) Estimated number of targets by each sensor. (c) Tracks estimated by sensor 1. (d) Tracks estimated by sensor 2. A track of a target is shown after its detection.

1 and its belief vector is $(b^2_{A,1}, b^2_{B,1})^T = (0.2, 0.8)^T$. At time 9, sensor 1 detects a new target (target 2 of sensor 1) and assigns a new identity (X) since the target is unknown to its neighboring sensors. The updated belief vectors are shown in Figure 5.5(a). At the same time, sensor 2 detects a new target (target 2 of sensor 2) and its identity and state estimate information is transferred from sensor 1, since its track is recognized as target 1 of sensor 1. The updated belief vectors are shown in Figure 5.5(b). At time 22, sensor 2



(a) (b)



(c) (d)

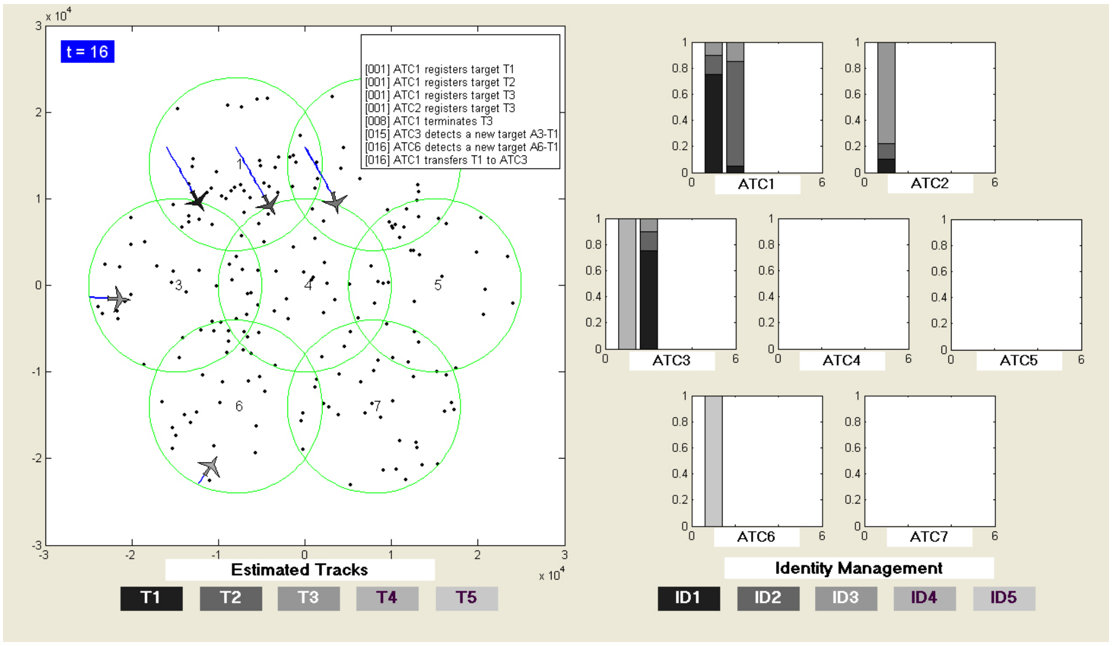
Figure 5.5: (a) Local belief vectors computed by sensor 1. (b) Local belief vectors computed by sensor 2. (c) Fused belief vector between target 1 of sensor 1 and target 2 of sensor 2 (d) Fused tracks. (The symbols $\times$, $+$, and $\Box$ denote aircraft A, aircraft B, and aircraft X, respectively).

detects a new target (target 3 of sensor 2) and its identity and state estimate information is transferred from sensor 1, since its track is recognized as target 2 of sensor 1. At time 26, target 2 of sensor 1 leaves the surveillance region of sensor 1 and information about target 2 is removed from sensor 1. At time 30, target 2 of sensor 2 leaves the surveillance region of sensor 2 and information about target 2 is removed from sensor 2.

For illustration purpose, Figure 5.5(a) and 5.5(b) are showing the local belief vectors at each sensor before *Identity and Track Fusion* (ITF). At time 21, aircraft A and aircraft X cross one another and the uncertainty about identity is increased as shown in Figure 5.5(a). For example, the belief that target 1 of sensor 1 can be identified with aircraft A is reduced from 0.8 to 0.45. However, ITF can reduce this uncertainty by fusing the belief vector of target 1 of sensor 1 and the belief vector of target 2 of sensor 2. We use Shannon information for ITF since we are considering a cooperative situation and it has been shown that Shannon information is superior against the other criteria in terms of cooperative efficiency [Hwang *et al.*, 2004b]. The fused belief vector is shown in Figure 5.5(c), in which the belief vectors from time 9 to 29 are shown. When target 3 of sensor 2 appears at time 23, *i.e.*, one time step after its detection, the identity uncertainty is reduced by ITF. For example, the (fused) belief that target 1 of sensor 1 can be identified with aircraft A is increased from 0.45 to 0.64 as shown in the bottom plot of Figure 5.5(c). Lastly, the tracks estimated by each sensor in a distributed manner are fused by ITF and the fused tracks are shown in Figure 5.5(d).

5.4.2 Seven-sensor Scenario

There are seven air traffic control radars (ATC1 through ATC7) and five aircraft (T1 through T5) as shown in Figure 5.6, which represents the state of the system at simulation time $t = 17$. The left frame of Figure 5.6 shows the position and heading of each aircraft along with sensing range (circle) of each sensor and measurements (dots). As the simulation time progresses, the left frame of Figure 5.6 will show the estimated tracks and event logs. The right frame of Figure 5.6 shows belief vectors of aircraft at each sensor. The order of belief vector is dictated by the order the target is registered to the corresponding sensor. Out of five targets, only three targets (T1, T2, and T3) are known to the system initially and the targets T4 and T5 are unknown to the system. At time $t = 1$, targets T1, T2 and T3 are registered to ATC1 and target T3 is registered to ATC2. The identities ID1, ID2 and ID3 are



```
[001] ATC1 registers target T1
[001] ATC1 registers target T2
[001] ATC1 registers target T3
[001] ATC2 registers target T3
[008] ATC1 terminates T3
[015] ATC3 detects a new target A3-T1
[016] ATC6 detects a new target A6-T1
[016] ATC1 transfers T1 to ATC3
```

Figure 5.6: (Simulation time $t = 16$) The left frame shows the estimated tracks and event logs. Circles and dots represent sensing ranges and measurements, respectively. The gray-scale of aircraft corresponds the true identity based on the gray-scale shown below *Estimated Tracks*. The right frame shows the belief vectors maintained by each sensor. The gray-scale of each segment of the bar graph represents an identity based on the gray-scale shown below *Identity Management*. At $t = 15$, ATC3 also detects a new target and the target is labeled as A3-T1 and identity ID4 is assigned to this target. At $t = 16$, ATC6 detects a new target and the target is labeled as A6-T1 and identity ID5 is assigned to this target. At the same time, ATC1 transfers T1 to ATC3.

associated to targets T1, T2 and T3, respectively. At $t = 8$, ATC1 terminates T3 since the target moves away from the sensing region of ATC1. At $t = 15$, ATC3 detects a new target and the target is labeled as A3-T1 and identity ID4 is assigned to this target. At $t = 16$, ATC6 detects a new target and the target is labeled as A6-T1 and identity ID5 is assigned to this target. At the same time, ATC1 transfers T1 to ATC3. See Figure 5.6. The snapshots at $t = 51$ and $t = 52$ are shown in Figure 5.7 and Figure 5.8, respectively. The uncertainty about the identities at $t = 52$ can be observed from the mixing of belief vectors.

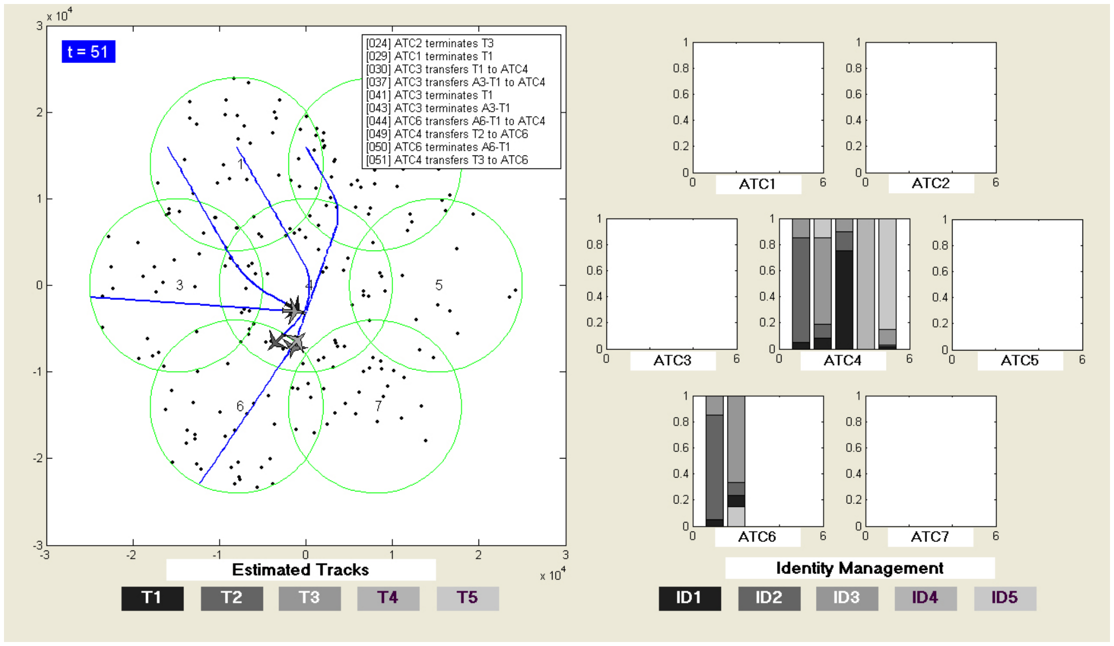


Figure 5.7: (Simulation time $t = 51$) At $t = 23$, ATC1 terminates T2 and ATC2 terminates T3 at $t = 24$. At $t = 29$, ATC1 terminates T1. At $t = 30$, ATC3 transfers T1 to ATC4. At $t = 37$, ATC3 transfers A3-T1 to ATC4. At $t = 41$, ATC3 terminates T1. At $t = 43$, ATC3 terminates A3-T1. At $t = 44$, ATC6 transfers A6-T1 to ATC4. At $t = 49$, ATC4 transfers T2 to ATC6. At $t = 50$, ATC6 terminates A6-T1. Targets T3 and A6-T1 move close to each other and the belief vectors of targets T3 and A6-T1 are mixed in ATC4. At $t = 51$, ATC4 transfers T3 to ATC6.

At $t = 80$, the belief matrix of ATC7 is

$$B_{\text{ATC7}}(80) = \begin{bmatrix} 0.2572 & 0.0514 & 0.0343 & 0.6571 & 0 \\ 0.4928 & 0.0986 & 0.0657 & 0.3429 & 0 \end{bmatrix}^T \tag{5.28}$$

where the rows 1 through 5 of the belief matrix $B_{\text{ATC7}}(80)$ correspond to identities ID1 through ID5 and the first and second columns of the belief matrix correspond to targets A3-T1 and T1, respectively.

At the same time, local information about target A3-T1 is obtained by ATC7 and target A3-T1 is now thought to have belief $[0.1\ 0\ 0\ 0.9\ 0]^T$. This local information is incorporated

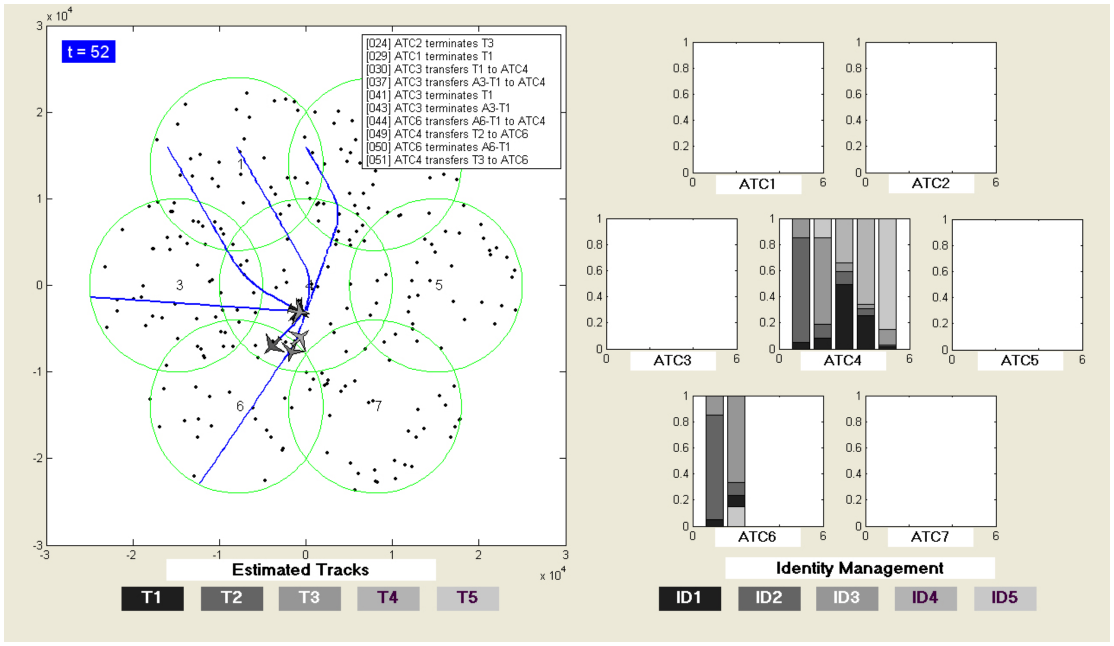

```
[024] ATC2 terminates T3
[029] ATC1 terminates T1
[030] ATC3 transfers T1 to ATC4
[037] ATC3 transfers A3-T1 to ATC4
[041] ATC3 terminates T1
[043] ATC3 terminates A3-T1
[044] ATC6 transfers A6-T1 to ATC4
[049] ATC4 transfers T2 to ATC6
[050] ATC6 terminates A6-T1
[051] ATC4 transfers T3 to ATC6
```

Figure 5.8: (Simulation time $t = 52$) At $t = 52$, targets T1 and A3-T1 move close to each other and the belief vectors of targets T1 and A3-T1 are mixed.

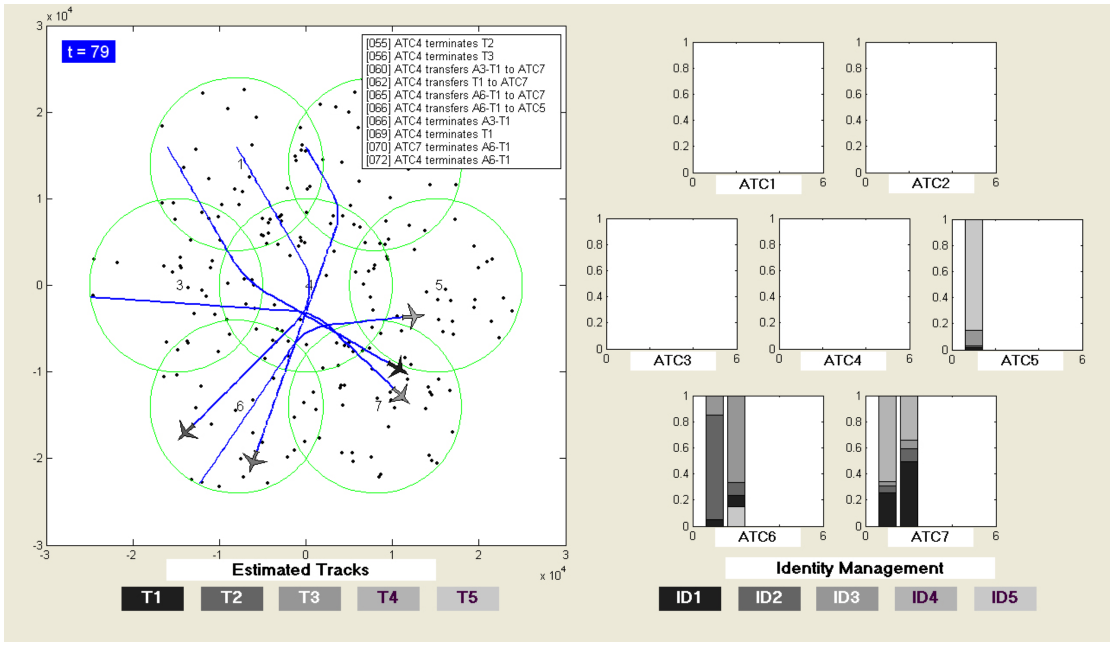
according to Algorithm 11. First, a new matrix $B'_{\text{ATC7}}(80)$ is formed by replacing the column for target A3-T1 with local information, where

$$B'_{\text{ATC7}}(80) = \begin{bmatrix} 0.1 & 0 & 0 & 0.9 & 0 \\ 0.4928 & 0.0986 & 0.0657 & 0.3429 & 0 \end{bmatrix}^T \tag{5.29}$$

Since $B'_{\text{ATC7}}(80)$ is almost scalable, Algorithm 11 computes the scaled belief matrix:

$$B^{\text{new}}_{\text{ATC7}}(80) = \begin{bmatrix} 0.1879 & 0 & 0 & 0.8121 & 0 \\ 0.5621 & 0.15 & 0.1 & 0.1879 & 0 \end{bmatrix}^T \tag{5.30}$$

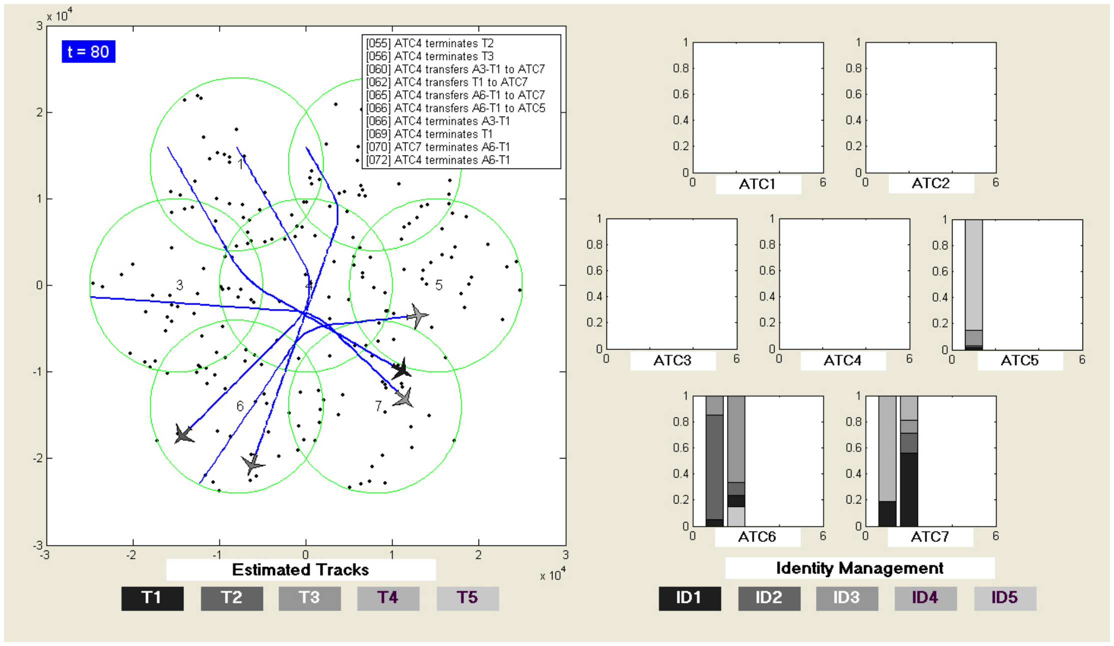
The scaled belief matrix $B^{\text{new}}_{\text{ATC7}}(80)$ decreases entropy from 2.9091 to 2.3602. Hence, we assign $B_{\text{ATC7}}(80) = B^{\text{new}}_{\text{ATC7}}(80)$. See the updated belief matrix in Figure 5.10 and compare it against Figure 5.9.



```
[055] ATC4 terminates T2
[056] ATC4 terminates T3
[060] ATC4 transfers A3-T1 to ATC7
[062] ATC4 transfers T1 to ATC7
[065] ATC4 transfers A6-T1 to ATC7
[066] ATC4 transfers A6-T1 to ATC5
[066] ATC4 terminates A3-T1
[069] ATC4 terminates T1
[070] ATC7 terminates A6-T1
[072] ATC4 terminates A6-T1
```

Figure 5.9: (Simulation time $t = 79$) Before the belief matrix update by ATC7.

In order to evaluate the performance of DMTIM, we compared DMTIM against the centralized version of the algorithm. In the centralized version, there is a single ATC at $[0, 0]$ with a larger sensing radius of 26 km. The same set of measurements are used except those on the overlapping sensing regions. When there are multiple measurements made by overlapping ATCs, a randomly chosen measurement is used by the centralized algorithm. The performance comparison between DMTIM and the centralized algorithm is made by comparing belief vectors and state estimates from both algorithms. The relative estimation errors in the position and belief vector estimations are shown in Figure 5.11. The relative position estimation error is high when there is confusion or when the same targets are detected by multiple sensors. The latter is due to the fact that a measurement from a single target can be different for a different ATC. Since the norm of standard deviations of position can be 20.77 when $\nu_t^k = 2$ and 81.54 when $\nu_t^k = 2$ (including both process and measurement noises), the position estimates of DMTIM are close to the estimates of the



```
[055] ATC4 terminates T2
[056] ATC4 terminates T3
[060] ATC4 transfers A3-T1 to ATC7
[062] ATC4 transfers T1 to ATC7
[065] ATC4 transfers A6-T1 to ATC7
[066] ATC4 transfers A6-T1 to ATC5
[066] ATC4 terminates A3-T1
[069] ATC4 terminates T1
[070] ATC7 terminates A6-T1
[072] ATC4 terminates A6-T1
```

Figure 5.10: (Simulation time $t = 80$) The belief matrix is updated due to the local information about target A3-T1 obtained by ATC7. See text for detail.

centralized algorithm. At $t = 52$ and $t = 53$, the relative position estimates are high for all targets and this is when the relative belief vector estimation error increases. As shown in Figure 5.11(b), the belief vector estimation error may increase over time, hence, it is necessary to perform the local information incorporation to reduce the error as described earlier. The centralized algorithm required more computation time than DMTIM since it processes more measurements per each time. Hence, the centralized algorithm is not a practical solution for a large-scale sensor network.

5.5 Summary

We have proposed a scalable distributed multi-target tracking and identity management (DMTIM) algorithm that can track an unknown number of targets and manage their identities efficiently in a distributed sensor network environment. A decision based on a single set

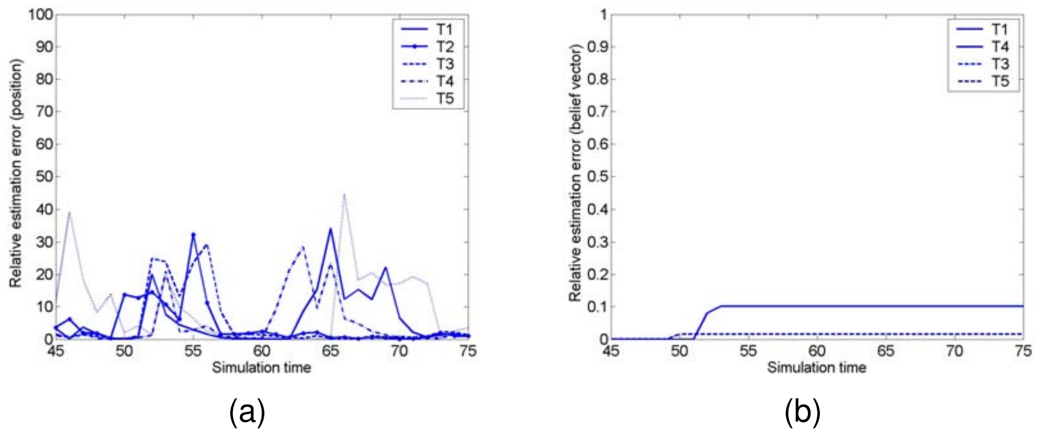


Figure 5.11: (a) Relative estimation error in the position of each target. (b) Relative estimation error in belief vectors of each target. (From simulation time $t = 45$ to $t = 75$.) For each target, the figures plot the maximum estimation error among all ATCs at each simulation time. The estimation error at each ATC is computed by calculating the vector norm of the difference between estimates made by the ATC and the centralized algorithm.

of tracks can be risky due to the uncertainty in the identities of targets accumulated from continuous interactions among crossing or nearby targets. DMTIM provides a scalable and distributed solution to the multi-target tracking and identity management problem by combining an efficient data association algorithm and identity management methods. The novelty of the system architecture of DMTIM includes modularity, the compact representation of identity and tracks to reduce communication load, and event-driven mechanisms for identity and track management.

DMTIM consists of data association, multi-target tracking, identity management, and identity and track fusion. The data association and multi-target tracking problems are efficiently solved by Markov chain Monte Carlo data association (MCMCDA). DMTIM efficiently incorporates local information about the identity of a target to reduce the uncertainty in the system and maintains local consistency among neighboring sensors via identity and track fusion.

In this chapter, the system is evaluated using simple measurement models. We are currently applying more sophisticated measurement models to evaluate the performance of the system. In particular, we are applying our method to the distributed camera network, in which the measurements can be, but not limited to, position, color, texture, and shape. Currently, the DAMTT module produces the mixing matrix for the IM module. But updated identity information from the IM and ITF modules are not used in the DAMTT module. We are currently developing methods to improve the performance of DAMTT by incorporating updated identity information in DAMTT.

CHAPTER 6

LOCHNESS: A REAL-TIME CONTROL SYSTEM FOR SENSOR NETWORKS

Wireless sensor networks are useful in applications that require locating and tracking moving targets and real-time dispatching of resources. Typical examples include search-and-rescue operations, civil surveillance systems, inventory systems for moving parts in a warehouse, and search-and-capture missions in military scenarios. The analysis and design of such applications are often reformulated within the framework of pursuit evasion games (PEGs), a mathematical abstraction which addresses the problem of controlling a swarm of autonomous agents in the pursuit of one or more evaders [Hespanha *et al.*, 1999; Vidal *et al.*, 2002]. The locations of moving targets (evaders) are unknown and their detection is typically accomplished by employing a network of cameras or by searching the area using mobile vehicles (pursuers) with on-board high resolution sensors. However, networks of cameras are rather expensive and require complex image processing to properly fuse their information. On the other hand, mobile pursuers with their on-board cameras or ultrasonic sensors with a relatively small detection range can provide only local observability over the area of interest. Therefore, a time-consuming exploratory phase is required

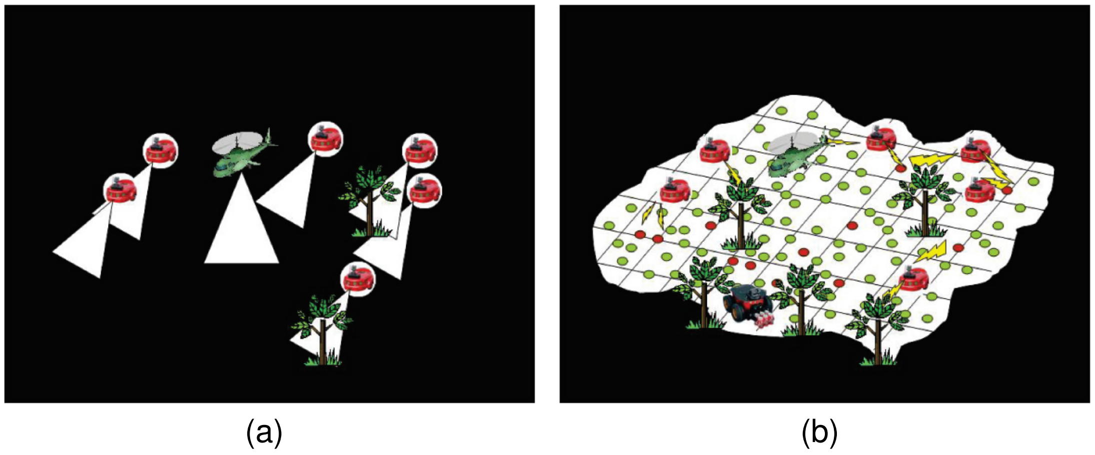


Figure 6.1: (a) Sensor visibility in PEGs without sensor network. (b) Sensor visibility in PEGs with sensor network. Dots correspond to sensor nodes, each provided with a vehicle detection sensor. Courtesy of [Sinopoli *et al.*, 2003].

[Thrun *et al.*, 1998; Guibas *et al.*, 1999]. This constraint makes the task of designing a cooperative pursuit algorithm harder because partial observability results in suboptimal pursuit policies (see Figure 6.1(a)). An inexpensive way to improve the overall performance of a PEG is to use wireless ad-hoc sensor networks [Sinopoli *et al.*, 2003]. With sensor networks, global observability of the field and long-distance communication are possible (see Figure 6.1(b)). Global pursuit policies can then be used to efficiently find the optimal solution regardless of the level of intelligence of the evaders. Also, with a sensor network, the number of pursuers needed is a function exclusively of the number of evaders and not the size of the field.

We consider the problem of pursuit evasion games (PEGs), where the objective of a group of pursuers is to chase and capture a group of evaders in minimum time with the aid of a sensor network. The evaders can either move randomly to model moving vehicles in search-and-rescue and traffic control applications, or can adopt evasive maneuvers to model search-and-capture missions in military scenarios.

While sensor networks provide global observability, they cannot provide high quality measurements in a timely manner due to packet loss, communication delay, and false detections. This has been the main challenge in developing a real-time control system using sensor networks. We address this challenge by developing a real-time hierarchical control system called *LochNess* (Large-scale "on-time" collaborative heterogeneous Networked embedded systems). *LochNess* decouples the estimation of evader states from the control of pursuers via multiple layers of data fusion. Although a sensor network generates noisy, inconsistent, and bursty measurements, the multiple layers of data fusion convert raw sensor measurements into fused measurements in a compact and consistent representation and forward the fused measurements to the pursuers' controllers in a timely manner.

In this chapter, we describe a real-time hierarchical control system *LochNess* for tracking and coordination using sensor networks and a demonstration of the system on a large-scale sensor network deployment. In the development of *LochNess*, three new algorithms developed:

- A multi-sensor fusion algorithm that combines noisy and inconsistent sensor measurements locally. The algorithm produces coherent evader position reports and reduces the communication load on the network.

- A multi-target tracking algorithm that tracks an unknown number of targets (or evaders). The algorithm is a hierarchical extension of the Markov chain Monte Carlo data association (MCMCDA) [Oh *et al.*, 2004] algorithm for sensor networks to add scalability. MCMCDA is a true approximation scheme for the *optimal Bayesian filter*; *i.e.*, when run with unlimited resources, it converges to the Bayesian solution [Oh *et al.*, 2006b]. MCMCDA is computationally efficient and robust against measurement noise and inconsistency (including packet loss and communication delay). In addition, MCMCDA operates with no or incomplete classification information, making it suitable for sensor networks. In fact, the performance of the algorithm can be improved given additional measurements to help identify the targets.

- A multi-agent coordination algorithm that assigns one pursuer to one evader such that the estimated time to capture the last evader is minimized based on the estimates computed by the multi-target tracking algorithm.

Our control system *LochNess* was successfully demonstrated using a large-scale sensor network. The system correctly found the number of evaders and their tracks and coordinated the pursuers to capture the evaders. Only a handful of the tracking algorithms in the literature that are designed for sensor networks have been demonstrated on a real sensor network deployment. Of these demonstrations, the algorithms are usually used to track a single target [Brooks *et al.*, 2004; Sharp *et al.*, 2005; Liu *et al.*, 2003a; Liu *et al.*, 2003b] or track multiple targets using classification [Arora *et al.*, 2004]. To our knowledge, we present the first demonstration of multi-target tracking using a sensor network without relying on classification.

6.1 Related Work in Target Tracking using Sensor Networks

One of the main applications of wireless ad-hoc sensor networks is surveillance. However, considering the resource constraints on each sensor node, the well known multi-target tracking algorithms such as joint probabilistic data association (JPDA) filter [Bar-Shalom and Fortmann, 1988] and multiple hypothesis tracker (MHT) [Reid, 1979; Chong *et al.*, 1990] are not feasible for sensor networks due to their exponential time and space complexities. As a result, many new tracking algorithms have been developed recently.

Most of the algorithms developed for sensor networks are designed for single-target tracking [Li *et al.*, 2002; Arora *et al.*, 2004; Brooks *et al.*, 2004; Sharp *et al.*, 2005; Liu *et al.*, 2003a; Zhao *et al.*, 2003; Liu *et al.*, 2003b; McErlean and Narayanan, 2002; Aslam *et al.*, 2003; Chen *et al.*, 2003; Coates, 2004; Oh and Sastry, 2005b] and some of these algorithms are applied to track multiple targets using classification [Li *et al.*, 2002; Arora *et al.*, 2004; Oh and Sastry, 2005b] or heuristics, such as the nearest-neighbor filter (NNF) [Brooks *et al.*, 2004]. A few algorithms are designed for multi-target tracking [Shin *et al.*, 2003; Chu *et al.*, 2004; Liu *et al.*, 2004] where the complexity of the data association problem inherent to multi-target tracking is avoided by classification [Shin *et al.*, 2003; Liu *et al.*, 2004] or heuristics [Chu *et al.*, 2004]. When tracking targets of a similar type or when reliable classification information is not available, the classification-based tracking algorithm behaves as the NNF. Considering the fact that the complexity of the data association problem is NP-hard [Collins and Uhlmann, 1992; Poore, 1995], a heuristic approach breaks down under difficult circumstances. Furthermore, the measurement inconsistencies common in sensor networks, such as false alarms and missing measurements (due to missing detection or packet loss), are not fully addressed in many algorithms. On the contrary, the multi-target tracking algorithm developed in this thesis is based on a rigorous probabilistic model and based on a true approximation scheme for the optimal Bayesian filter.

Tracking algorithms for sensor networks can be categorized according to their computational structure: centralized [Arora *et al.*, 2004; Sharp *et al.*, 2005; Aslam *et al.*, 2003], hierarchical [Chen *et al.*, 2003; Coates, 2004], or distributed [Li *et al.*, 2002; Brooks *et al.*, 2004; Liu *et al.*, 2003a; Zhao *et al.*, 2003; Liu *et al.*, 2003b; McErlean and Narayanan, 2002; Oh and Sastry, 2005b; Shin *et al.*, 2003; Chu *et al.*, 2004; Liu *et al.*, 2004]. However, since each sensor has only local sensing capability and its measurements are noisy and inconsistent, measurements from a single sensor and its neighboring sensors are not sufficient to initiate, maintain, disambiguate, and terminate tracks of multiple targets in the presence of clutter; it requires measurements from distant sensors. Considering the communication load and delay when exchanging measurements between distant sensors, a completely distributed approach to solve the multi-target tracking problem is not feasible for real-time applications. On the other hand, a completely centralized approach is not robust and scalable. In order to minimize the communication load and delay while being robust and scalable, a hierarchical architecture is considered in this chapter.

6.2 Problem Formulation and Control System Architecture

We consider the problem of pursuing multiple evaders over a region of interest (or the surveillance region). Evaders (or targets) arise at random in space and time, persist for a random length of time, and then cease to exist. When evaders appear, a group of pursuers is required to detect, chase and capture the group of evaders in minimum time with the aid of a sensor network. In order to solve this problem, we propose a hierarchical real-time control system *LochNess* which is shown in Figure 6.2. *LochNess* is composed of seven layers: the *sensor network*, the *multi-sensor fusion* (MSF) module, the *multi-target tracking* (MTT) modules, the *multi-track fusion* (MTF) module, the *multi-agent coordination* (MAC) module, the *path planner* module, and the *path follower* modules.

Sensors are spread over the surveillance region and form an ad-hoc network. The sensor network detects moving objects in the surveillance region and the MSF module converts the sensor measurements into target position estimates (or reports) using spatial correlation. We consider a hierarchical sensor network. In addition to regular sensor nodes ("Tier-1" nodes), we assume the availability of "Tier-2" nodes which have long-distance wireless links and more processing power. We assume that each Tier-2 node can communicate with its neighboring Tier-2 nodes. Examples of a Tier-2 node include high-bandwidth sensor nodes such as iMote and BTnode [Hill *et al.*, 2004], gateway nodes such as Stargate, Intrinsyc Cerfcube, and PC104 [Hill *et al.*, 2004], and the Tier-2 nodes designed for our experiment [Dutta *et al.*, 2006]. Each Tier-1 node is assigned to its nearest Tier-2 node and the Tier-1 nodes are grouped by Tier-2 nodes. We call the group of sensor nodes formed around a Tier-2 node a "tracking group". When a node detects a possible target, it listens

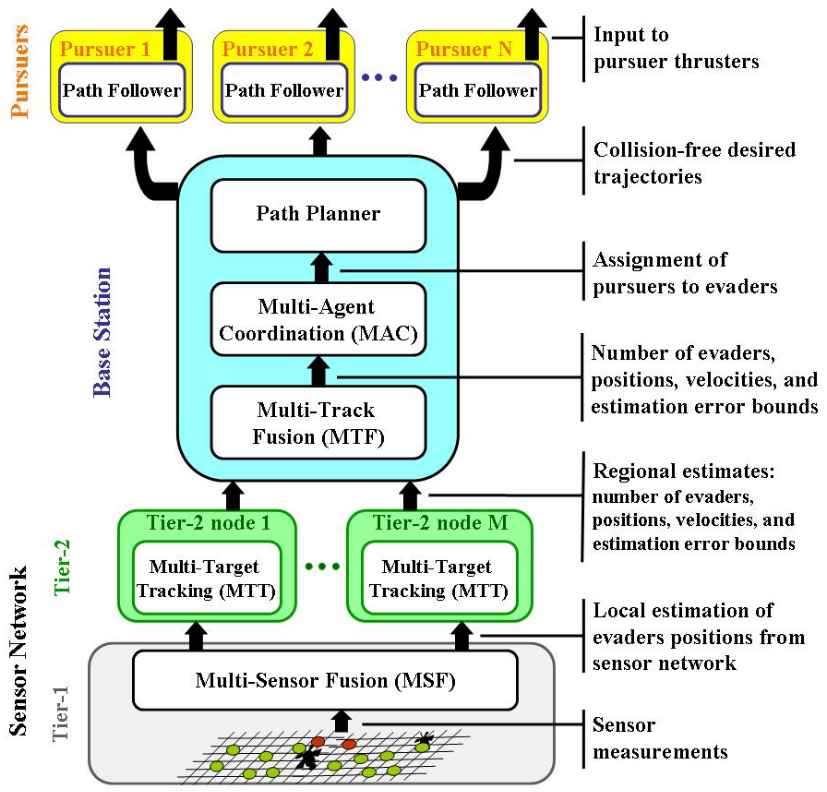

Pursuers
Pursuer 1
Pursuer 2
Pursuer N
Path Follower
Path Follower
Path Follower
Input to pursuer thrusters
Collision-free desired trajectories
Base Station
Path Planner
Assignment of pursuers to evaders
Multi-Agent Coordination (MAC)
Number of evaders, positions, velocities, and estimation error bounds
Multi-Track Fusion (MTF)
Regional estimates: number of evaders, positions, velocities, and estimation error bounds
Tier-2
Tier-2 node 1
Tier-2 node M
Multi-Target Tracking (MTT)
Multi-Target Tracking (MTT)
Local estimation of evaders positions from sensor network
Sensor Network
Tier-1
Multi-Sensor Fusion (MSF)
Sensor measurements

Figure 6.2: *LochNess*: a hierarchical real-time control system architecture using sensor networks for multi-target tracking and multi-agent coordination.

to its neighbors for their measurements and fuses the measurements to forward to its Tier-2 node. Each Tier-2 node receives the fused measurements from its tracking group and the MTT module in each Tier-2 node estimates the number of evaders, the positions and velocities of the evaders, and the estimation error bounds. Each Tier-2 node communicates with its neighboring Tier-2 nodes when a target moves away from the region monitored by its tracking group. Lastly, the tracks estimated by the Tier-2 nodes are combined hierarchically by the MTF module at the base station.

The estimates computed by the MTF module are then used by the MAC module to estimate the expected capture times of all pursuer-evader pairs. Based on these estimates, the MAC module assigns one pursuer to one evader by solving the bottleneck assignment problem [Burkard and Çela, 1998] such that the estimated time to capture the last evader is minimized. Once the assignments are determined, the path planner module computes a trajectory for each pursuer to capture its assigned evader in the least amount of time without colliding into other pursuers. Then, the base station transmits each trajectory to the path following controller of the corresponding pursuer. The path following controller modifies the pursuer's trajectory on the fly to avoid any obstacles sensed by the pursuer's on-board sensors. The path planning and path follower modules can be implemented using dynamic programming [Nilim and Ghaoui, 2004] or model predictive control [Shim *et al.*, 2003]. In the chapter, we focus on MSF, MTT, MTF, and MAC modules and they are described in Section 6.3. In the remainder of this section, we describe the sensor network model and the problem formulations of multi-target tracking and multi-agent coordination.

6.2.1 Sensor Network and Sensor Models

In this section, we describe the sensing models – the signal-strength and binary sensor models – and the sensor network model considered in this chapter. A signal-strength sensor reports the range to a nearby target while a binary sensor reports only a binary value indicating whether an object is detected near the reporting sensor. The signal-strength sensor model is used for the development and analysis of our system while the binary sensor model is used in our experiments. While the signal-strength sensors provide better accuracy, our evaluation of the sensors developed for the experiments showed that the variability in the signal strength of the sensor reading prohibited extraction of ranging information. However, we found that the sensors were still effective as binary sensors. We also found that binary sensors were much less sensitive to time synchronization errors than signal-strength sensors.

Let N_s be the number of sensor nodes, including both Tier-1 and Tier-2 nodes, deployed over the surveillance region $\mathcal{R} \subset \mathbb{R}^2$. Let $s_i \in \mathcal{R}$ be the location of the i-th sensor node and let $S = \{s_i : 1 \leq i \leq N_\text{s}\}$. Let $N_\text{ss} \ll N_\text{s}$ be the number of Tier-2 nodes and let $s^\text{s}_j \in S$ be the position of the j-th Tier-2 node, for $j = 1, \ldots, N_\text{ss}$.

6.2.1.1 Signal-Strength Sensor Model

Let $R_\text{s} \in \mathbb{R}$ be the sensing range. If there is an object at $x \in \mathcal{R}$, a sensor can detect the presence of the object. Each sensor records the sensor's signal strength,

$$z_i = \begin{cases} \frac{\beta}{1+\gamma\|s_i - x\|^\alpha} + w^\text{s}_i, & \text{if } \|s_i - x\| \leq R_\text{s} \\ w^\text{s}_i, & \text{if } \|s_i - x\| > R_\text{s}, \end{cases} \tag{6.1}$$

where α, β and γ are constants specific to the sensor type, and we assume that z_i are normalized such that w^s_i has the standard Gaussian distribution. This signal-strength based sensor model is a general model for many sensors available in sensor networks, such as acoustic and magnetic sensors, and has been used frequently [Brooks *et al.*, 2004; Liu *et al.*, 2003a; Liu *et al.*, 2003b; Liu *et al.*, 2004].

6.2.1.2 Binary Sensor Model

For each sensor i, let R_i be the sensing region of i. R_i can have an arbitrary shape but we assume that it is known to the system. Let $z_i \in \{0, 1\}$ be the detection made by sensor i, such that sensor i reports $z_i = 1$ if it detects a moving object in R_i, and $z_i = 0$ otherwise. Let p_i be the detection probability and q_i be the false detection probability of sensor i.

6.2.1.3 Sensor Network Model

Let $G = (S, E)$ be a communication graph such that $(s_i, s_j) \in E$ if and only if node i can communicate directly to node j. Let $g : \{1, \ldots, N_\text{s}\} \to \{1, \ldots, N_\text{ss}\}$ be the assignment of each sensor to its nearest Tier-2 node such that $g(i) = j$ if $\|s_i - s^\text{s}_j\| = \min_{k=1,\ldots,N_\text{ss}} \|s_i -$

$s_k^{\mathrm{s}}\|$. For a node i, if $g(i) = j$, the shortest path from s_i to s_j^{s} in G is denoted by $sp(i)$. We assume that the length of $sp(i)$, *i.e.*, the number of communication links from node i to its Tier-2 node, is smaller when the physical distance between node i and its Tier-2 node is shorter. But if this is not the case, we can assign a node to the Tier-2 node with the fewest communication links between them.

Local sensor measurements are fused by the MSF module described in Section 6.3.1. Let $\hat{z}_i$ be a fused measurement originated from node i. Node i transmits the fused measurement $\hat{z}_i$ to the Tier-2 node $g(i)$ via the shortest path $sp(i)$. A transmission along an edge (s_i, s_j) on the path fails independently with probability p_{te} and the message never reaches the Tier-2 node. Transmission failures along an edge (s_i, s_j) may include failures from retransmissions from node i to node j. We can consider transmission failure as another form of a missing observation. If k is the number of hops required to relay data from a sensor node to its Tier-2 node, the probability of successful transmission decays exponentially as k increases. To overcome this problem, we use k independent paths to relay data if the reporting sensor node is k hops away from its Tier-2 node. The probability of successful communication p_{cs} from the reporting node i to its Tier-2 node $g(i)$ can be computed as $p_{\text{cs}}(p_{\text{te}}, k) = 1 - \left(1 - (1 - p_{\text{te}})^k\right)^k$, where $k = |sp(i)|$ and $|sp(i)|$ denotes the cardinality of the set $sp(i)$.

We assume each node has the same probability p_{de} of delaying a message. If d_i is the number of (additional) delays on a message originating from the sensor i, then d_i is distributed as

$$p(d_i = d) = \binom{|sp(i)| + d - 1}{d} (1 - p_{\text{de}})^{|sp(i)|} (p_{\text{de}})^d. \tag{6.2}$$

We are modeling the number of (additional) delays by the negative binomial distribution. A negative binomial random variable represents the number of failures before reaching a fixed number of successes from Bernoulli trials. In our case, it is the number of delays before $|sp(i)|$ successful delay-free transmissions.

If the network is heavily loaded, the independence assumptions on transmission failure and communication delay may not hold. However, the model is realistic under moderate conditions and we have chosen it for its simplicity.

6.2.2 Agent Dynamics and Coordination Objective

In a situation where multiple pursuers and evaders are present, several assignments are possible and some criteria need to be chosen to optimize performance. In this work, we focus on minimizing the time to capture all evaders. However, other criteria might be possible, such as minimizing the pursuer's energy consumption while guaranteeing capture of all evaders or maximizing the number of captured evaders within a certain amount of time. Since the evaders' motions are not known, an exact time to capture a particular evader is also not known. Therefore, we need to define a metric to estimate the time to capture the evaders. Let us define the state vector of a vehicle as $x = [x_1, x_2, \dot{x}_1, \dot{x}_2]^T$, where (x_1, x_2) and $(\dot{x}_1, \dot{x}_2)$ are the position and the velocity components of the vehicle along the x and y axes, respectively. We denote by x^{p} and x^{e} the state of a pursuer and an evader, respectively. We will use the following definition of time-to-capture:

Definition 2 (Time-to-capture). *Let $x^e(t_0)$ be the position and velocity vector of an evader in a plane at time t_0, and $x^p(t)$ be the position and velocity vector of a pursuer at the current time $t \geq t_0$. We define the (constant speed) time-to-capture as the minimum time T_c necessary for the pursuer to reach the evader with the same velocity, assuming that the evader will keep moving at a constant velocity,* i.e.,

$$T_c := \min \Big[T \mid x^p(t+T) = x^e(t+T) \Big], \tag{6.3}$$

where $x^e_{1,2}(t+T) = x^e_{1,2}(t_0) + (t+T-t_0)\dot{x}^e_{1,2}(t_0)$, $\dot{x}^e_{1,2}(t+T) = \dot{x}^e_{1,2}(t_0)$, and the pursuer moves according to its dynamics.

This definition allows us to quantify the time-to-capture in an unambiguous way. Although an evader can change trajectories over time, it is a more accurate estimate than, for example, some metric based on the distance between an evader and a pursuer, since the time-to-capture incorporates the dynamics of the pursuer.

Given Definition 2 and the constraints on the dynamics of the pursuer, it is possible to calculate explicitly the time-to-capture T_c, as well as the optimal trajectory $x^{\mathrm{e}*}(t)$ for the pursuers as shown in Section 6.3.3.

We assume the following dynamics for both pursuers and evaders:

$$\begin{aligned} x(t+\delta) &= A_\delta x(t) + G_\delta u(t) && (6.4) \\ \eta(t) &= x(t) + v(t) && (6.5) \end{aligned}$$

where δ is the sampling interval, $u = [u_1, u_2]^T$ is the control input vector, $\eta(t)$ is the estimated vehicle state provided by the MTF module, $v(t)$ is the estimation error, and

$$A_\delta = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G_\delta = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix},$$

which correspond to the discretization of the dynamics of a decoupled planar double integrator. Although this model appears simplistic for modeling complex motions, it is widely used as a first approximation in path-planning [Lepetic *et al.*, 2003; Saccon, 2005; Velenis and Tsiotras, 2005]. Moreover, there exist methodologies to map such a simple dynamic model into a more realistic model via consistent abstraction as shown in [Belta *et al.*, 2005; Tabuada and Pappas, 2005]. Finally, any possible mismatch between this model and the true vehicle dynamics can be compensated for by the path-follower controller implemented on the pursuer [Shim *et al.*, 2003].

The observation vector $\eta = [\eta_1, \eta_2, \dot{\eta}_1, \dot{\eta}_2]^T$ is interpreted as a measurement, although in reality it is the output from the MTF module shown in Figure 6.2. The estimation error $v_t = [v_1, v_2, \dot{v}_1, \dot{v}_2]^T$ can be modeled as a Gaussian noise with zero mean and covariance Q or as an unknown but bounded error, *i.e.*, $|v_1| < V_1, |v_2| < V_2, |\dot{v}_1| < \dot{V}_1, |\dot{v}_2| < \dot{V}_2$, where V_1, V_2, $\dot{V}_1$ and $\dot{V}_2$ are positive scalars that are possibly time-varying. Both modeling approaches are useful for different reasons. Using a Gaussian noise approximation allows a closed-form optimal filter solution such as the well-known Kalman filter [Kailath *et al.*, 1999]. On the other hand using the unknown but bounded error model allows for the design of a robust controller such as the robust minimum-time control of pursuers proposed in Section 6.3.3.

We also assume that the control input to a pursuer is bounded, *i.e.*,

$$|u_1^{\mathrm{p}}| \leq U_{\mathrm{p}},\ |u_2^{\mathrm{p}}| \leq U_{\mathrm{p}} \tag{6.6}$$

where $U_{\mathrm{p}} > 0$. We consider two possible evader dynamics:

$$u_1^{\mathrm{e}} \sim \mathcal{N}(0, q_{\mathrm{e}}), u_2^{\mathrm{e}} \sim \mathcal{N}(0, q_{\mathrm{e}}) \qquad \text{(random motion)} \tag{6.7}$$

$$|u_1^{\mathrm{e}}| \leq U_{\mathrm{e}}, |u_2^{\mathrm{e}}| \leq U_{\mathrm{e}} \qquad \text{(evasive motion),} \tag{6.8}$$

where $\mathcal{N}(0, q_{\mathrm{e}})$ is a Gaussian distribution with zero mean and variance $q_{\mathrm{e}} \in \mathbb{R}^+$. Equation (6.7) is a standard model for the unknown motion of vehicles, where the variation in a velocity component is a discrete-time white noise acceleration [Lerro and Bar-Shalom, 1993]. Equation (6.8) allows for evasive maneuvers but places bounds on the maximum thrust. The multi-agent coordination scheme proposed in Section 6.3.3 is based on dynamics (6.8) as pursuers choose their control actions to counteract the best possible evasive maneuver of the evader being chased. However, in our simulations and experiments, we test our control architecture using the dynamics (6.7) for evaders where we set $q_{\mathrm{e}} = 2U_{\mathrm{e}}$.

Since the definition of the time-to-capture is related to relative distance and velocity between the pursuer and the evader, we consider the state space error $\xi = x^{\mathrm{p}} - x^{\mathrm{e}}$ which evolves according to the following error dynamics:

$$\begin{aligned} \xi(t+\delta) &= A_\delta \xi(t) + G_\delta u^{\mathrm{p}}(t) - G_\delta u^{\mathrm{e}}(t) \\ \eta^{\xi}(t) &= \xi(t) + v^{\xi}(t) \end{aligned} \tag{6.9}$$

where the pursuer thrust $u^{\mathrm{p}}(t)$ is the only controllable input, while the evader thrust $u^{\mathrm{e}}(t)$ acts as a random or unknown disturbance, and $v^{\xi}(t)$ is the measurement error which takes into account the uncertainties on the states of both the pursuer and the evader. According to the definition above, an evader is captured if and only if $\xi(t) = 0$, and the time-to-capture T_{c} corresponds to the time necessary to drive $\xi(t)$ to zero assuming $u^{\mathrm{e}}(t) = 0$ for $t \geq t_0$. However, this assumption is relaxed in Section 6.3.3.

According to the definition of time-to-capture above and the error dynamics (6.9), given the positions and velocities of all the pursuers and evaders, it is possible to compute the time-to-capture matrix $C = [c_{ij}] \in \mathbb{R}^{N_{\mathrm{p}} \times N_{\mathrm{e}}}$, where N_{p} and N_{e} are the total number of pursuers and evaders, respectively, and the entry c_{ij} of the matrix C corresponds to the expected time-to-capture between pursuer i and evader j. When coordinating multiple pursuers to chase multiple evaders, it is necessary to assign pursuers to evaders. Our objective is to select an assignment that minimizes the expected time-to-capture of *all* evaders, which correspond to the *global worst case time-to-capture*. We focus on a scenario with the same number of pursuers and evaders, *i.e.*, $N_{\mathrm{p}} = N_{\mathrm{e}}$. When there are more pursuers than evaders, then only a subset of all the pursuers can be dispatched and the others are kept on alert in case more evaders appear. Alternatively, more pursuers can be assigned to a single evader. When there are more evaders than pursuers, one approach is to minimize the time to capture the N_{p} closest evaders. Obviously, many different coordination objectives can be formulated as they are strongly application-dependent. We have chosen the definition of global worst case time-to-capture as it enforces strong global coordination to achieve high performance.

6.3 Control System Implementation

6.3.1 Multi-sensor Fusion Module

6.3.1.1 Signal-strength Sensor Model

Consider the signal-strength sensor model described in Section 6.2.1. Recall that z_i is the signal strength measured by node i. For each node i, if $z_i \geq \theta$, where θ is a threshold set for appropriate values of detection and false-positive probabilities, the node transmits z_i to its neighboring nodes, which are at most $2R_\mathrm{s}$ away from s_i, and listens to incoming messages from neighboring nodes within a $2R_\mathrm{s}$ radius. We assume that the communication range of each node is larger than $2R_\mathrm{s}$. For a node i, if z_i is larger than all incoming messages, $z_{i_1}, \ldots, z_{i_{k-1}}$, and $z_{i_k} = z_i$, then the position of an object is estimated by

$$\hat{z}_i = \frac{\sum_{j=1}^{k} z_{i_j} s_{i_j}}{\sum_{j=1}^{k} z_{i_j}}. \tag{6.10}$$

The estimate $\hat{z}_i$ corresponds to a center of mass of the node locations weighed by their measured signal strengths. Node i transmits $\hat{z}_i$ to the Tier-2 node $g(i)$. If z_i is not the largest compared to the incoming messages, node i simply continues sensing. Although each sensor cannot give an accurate estimate of the object's position, as more sensors collaborate, the accuracy of the estimates improves as shown in Figure 6.3.

6.3.1.2 Binary Sensor Model

In order to obtain finer position reports from binary detections, we use spatial correlation among detections from neighboring sensors. The idea behind the fusion algorithm is to

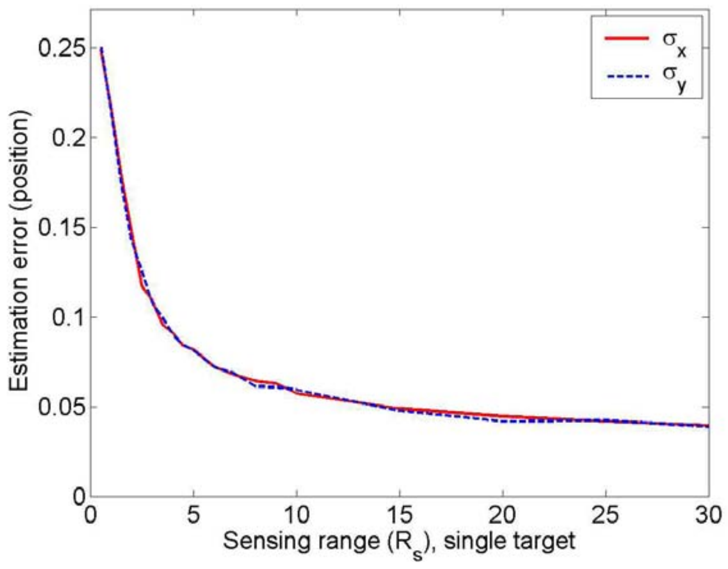


Figure 6.3: Single target position estimation error as a function of sensing range. See Section 6.4.5 for the sensor network setup used in simulations (Monte Carlo simulation of 1000 samples, unity corresponds to the separation between sensors).

compute the likelihood of detections assuming there is a single target. This is only an approximation since there can be more than one target. However, any inconsistencies caused by this approximation are fixed by the tracking algorithm described in Section 6.3.2 using spatio-temporal correlation.

Consider the binary sensor model described in Section 6.2.1. Let x be the position of an object. For the purpose of illustration, suppose that there are two sensors, sensor 1 and sensor 2, and $R_1 \cap R_2 \neq \emptyset$ (see Figure 6.4(a)). The overall sensing region $R_1 \cup R_2$ can be partitioned into a set of non-overlapping cells (or blocks) as shown in Figure 6.4(b). The likelihoods can be computed as follows:

$$\begin{aligned} P(z_1, z_2 | x \in S_1) &= p_1^{z_1}(1-p_1)^{1-z_1} q_2^{z_2}(1-q_2)^{1-z_2} \\ P(z_1, z_2 | x \in S_2) &= q_1^{z_1}(1-q_1)^{1-z_1} p_2^{z_2}(1-p_2)^{1-z_2} \\ P(z_1, z_2 | x \in S_3) &= p_1^{z_1}(1-p_1)^{1-z_1} p_2^{z_2}(1-p_2)^{1-z_2}, \end{aligned} \tag{6.11}$$

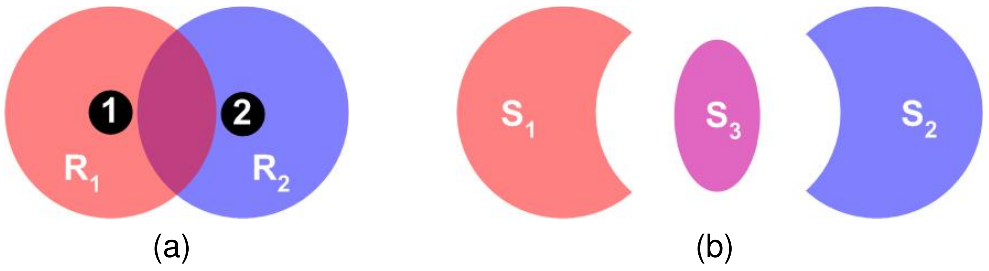


Figure 6.4: (a) Sensing regions of two sensors 1 and 2. R_i is the sensing region of sensor i. (b) A partition of the overall sensing region $R_1 \cup R_2$ into non-overlapping cells S_1, S_2 and S_3, where $S_1 = R_1 \setminus R_2$, $S_2 = R_2 \setminus R_1$ and $S_3 = R_1 \cap R_2$.

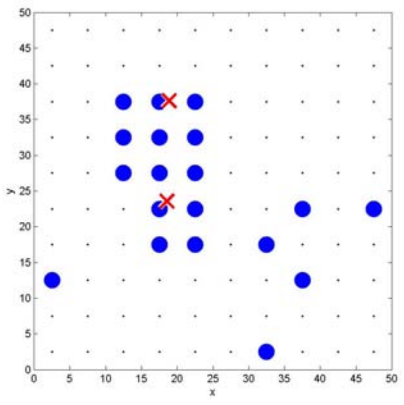
where $S_1 = R_1 \setminus R_2$, $S_2 = R_2 \setminus R_1$ and $S_3 = R_1 \cap R_2$ (see Figure 6.4(b)). Hence, for any deployment we can first partition the surveillance region into a set of non-overlapping cells. Then, given detection data, we can compute the likelihood of each cell as shown in the previous example.

An example of detections of two targets by a 10×10 sensor grid is shown in Figure 6.5(a). In this example, the sensing region is assumed to be a disk with radius of 7.62m (10 ft). We have assumed $p_i = 0.7$ and $q_i = 0.05$ for all i. These parameters are estimated from measurements made with the passive infrared (PIR) sensor of an actual sensor node described in Section 6.5. From the detections shown in Figure 6.5(a), the likelihood can be computed using equations similar to (6.11) for each non-overlapping cell (see Figure 6.5(b)). Notice that it is a time-consuming task to find all non-overlapping cells for arbitrary sensing region shapes and sensor deployments. Hence, we quantized the surveillance region and the likelihoods are computed for a finite number of points as shown in Figure 6.5(b).

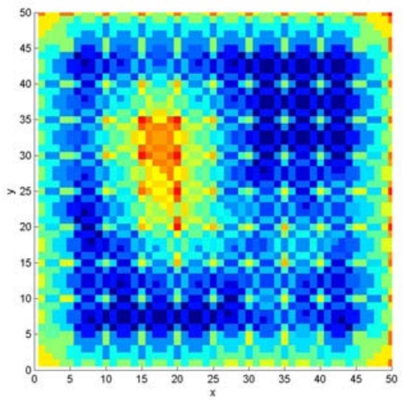
There are two parts in this likelihood computation: the detection part (terms involving p_i) and the false detection part (terms involving q_i). Hereafter, we call the detection part of the likelihood as the *detection-likelihood* and the false detection part of the likelihood as the *false-detection-likelihood*. Notice that the computation of the false-detection-likelihood requires measurements from all sensors. However, for a large wireless sensor network, it is not feasible to exchange detection data with all other sensors. Instead, we use a threshold test to avoid computing the false-detection-likelihood and distribute the likelihood computation. The detection-likelihood of a cell is computed if there are at least $\theta_{\rm d}$ detections, where $\theta_{\rm d}$ is a user-defined threshold. Using $\theta_{\rm d} = 3$, the detection-likelihood of the detections from Figure 6.5(a) can be computed as shown in Figure 6.5(c). The computation of the detection-likelihood can be done in a distributed manner. Assign a set of non-overlapping cells to each sensor such that no two sensors share the same cell and each cell is assigned to a sensor whose sensing region includes the cell. For each sensor i, let $\{S_{i_1}, \ldots, S_{i_{m(i)}}\}$ be a set of non-overlapping cells, where $m(i)$ is the number of cells assigned to sensor i. Then, if sensor i reports a detection, it computes the likelihoods of each cell in $\{S_{i_1}, \ldots, S_{i_{m(i)}}\}$ based on its own measurements and the measurements from neighboring sensors. A neighboring sensor is a sensor whose sensing region intersects the sensing region of sensor i. Notice that no measurement from a sensor means no detection.

Based on the detection-likelihoods, we compute target position reports by clustering. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a set of cells whose detection-likelihoods are computed, *i.e.*, the number of detections for each S_i is at least $\theta_{\rm d}$. First, randomly pick S_j from $\mathcal{S}$ and remove S_j from $\mathcal{S}$. Then cluster around S_j the remaining cells in $\mathcal{S}$ whose set-distance to S_j is less than the sensing radius. The cells clustered with S_j are then removed from $\mathcal{S}$. Now repeat the procedure until $\mathcal{S}$ is empty. Let $\{C_k : 1 \leq k \leq K_{\rm cl}\}$ be the clusters formed by this procedure, where $K_{\rm cl}$ is the total number of clusters. For each cluster C_k, its center of mass is computed to obtain a a fused position report, *i.e.*, an estimated position of a target. An example of position reports is shown in Figure 6.5(c).

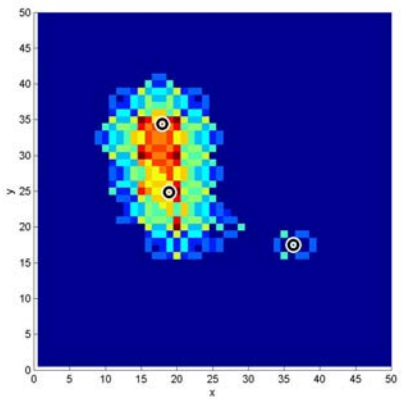
The multi-sensor fusion algorithm described above runs on two levels: Algorithm 12 on the Tier-1 nodes and Algorithm 13 on the Tier-2 node. Each Tier-1 node combines detection data from itself and neighboring nodes using Algorithm 12 and computes detection-likelihoods. The detection-likelihoods are forwarded to its Tier-2 node and the Tier-2 node



(a)



(b)



(c)

Figure 6.5: (a) Detections of two targets by a 10×10 sensor grid (targets in $\times$, detections in disks, and sensor positions in small dots). (b) Likelihood of detections from Figure 6.5(a). (c) Detection-likelihood of detections from Figure 6.5(a) with threshold $\theta_{\text{d}} = 3$. Estimated positions of targets are shown in circles.

generates position reports from the detection-likelihoods using Algorithm 13. The position reports are then used by the MTT module described in Section 6.3.2 to track multiple targets.

Algorithm 12 Multi-sensor Fusion: Sensor i

```
Input:  detections from sensor $i$ and its neighbors
Output:  detection-likelihoods
1: for each $S_{i_j}$, $j = 1, \dots, m(i)$ do
2:     if number of detections for $S_{i_j} \geq \theta_{\text{d}}$ then
3:         compute detection-likelihood $\hat{z}_i(j)$ of $S_{i_j}$
4:         forward $\hat{z}_i(j)$ to Tier-2 node $g(i)$
5:     end if
6: end for
```

Algorithm 13 Multi-sensor Fusion: Tier-2 Node

```
Input:  detection-likelihoods $\mathcal{Z} = \{\hat{z}_i(j)\}$ received from its tracking group
Output:  position reports $y$
1: $\mathcal{S} = \{S_{i_j} : \hat{z}_i(j) \in \mathcal{Z}\}$
2: $y = \emptyset$
3: find clusters $\{C_k : 1 \leq k \leq K_{\text{cl}}\}$ from $\mathcal{S}$ as described in the text
4: for each $C_k$, $k = 1, \dots, K_{\text{cl}}$ do
5:     compute the center of mass $y_k$ of $C_k$
6:     $y = y \cup y_k$
7: end for
```

6.3.2 Multi-target Tracking and Multi-track Fusion Modules

6.3.2.1 Multi-target Tracking Module

At each Tier-2 node, we implement the online MCMCDA algorithm (Algorithm 8) with a sliding window of size w_{s}. At each time step, we use the previous estimate to initialize MCMCDA and run MCMCDA on the observations belonging to the current window. Each Tier-2 node maintains a set of observations $Y = \{y^j(t) : 1 \leq j \leq n(t), t_{\text{curr}} - w_{\mathrm{s}} + 1 \leq t \leq t_{\text{curr}}\}$, where t_{curr} is the current time. Each $y^j(t)$ is either a fused measurement $\hat{z}_i$ from some signal-strength sensor i or an element of the fused position reports y from some binary sensors. At time $t_{\text{curr}} + 1$, the observations at time $t_{\text{curr}} - w_{\mathrm{s}} + 1$ are removed from Y and a new set of observations is appended to Y. Any delayed observations are inserted into the appropriate slots. Then, each Tier-2 node initializes the Markov chain with the previously estimated tracks and executes Algorithm 8 on Y. Once a target is found, the next state of the target is predicted. If the predicted next state belongs to the surveillance area of another Tier-2 node, the target's track information is passed to the corresponding Tier-2 node. These newly received tracks are then incorporated into the initial state of MCMCDA for the next time step. Lastly, each Tier-2 node forwards its track information to the base station.

6.3.2.2 Multi-track Fusion Module

Since each Tier-2 node maintains its own set of tracks, there can be multiple tracks from a single target maintained by different Tier-2 nodes. To make the algorithm fully hierarchical and scalable, the MTF module performs the track-level data association at the base station to combine tracks from different Tier-2 nodes. Let ω_j be the set of tracks maintained by Tier-2 node $j \in \{1, \ldots, N_{\text{ss}}\}$. Let $Y_{\mathrm{c}} = \{\tau_i(t) \in \omega_j : 1 \leq t \leq t_{\text{curr}}, 1 \leq i \leq |\omega_j|, 1 \leq j \leq N_{\text{ss}}\}$ be the combined observations only from the established tracks. We form a new set of tracks ω_{init} from $\{\tau_i \in \omega_j : 1 \leq i \leq |\omega_j|, 1 \leq j \leq N_{\text{ss}}\}$ while making sure that the constraints defined in Chapter 3 are satisfied. Then, we run Algorithm 7 on this combined observation set Y_c with the initial state ω_{init}. An example in which the multi-track fusion corrects mistakes made by Tier-2 nodes due to missing observations at the tracking group boundaries is shown in Section 6.4.5. The algorithm is autonomous and shown to be robust against packet loss, communication delay and sensor localization error (see Section 6.4).

6.3.3 Multi-agent Coordination Module

The time-to-capture is estimated using the abstract model of pursuer and evader dynamics given in Section 6.2.2. Let us consider the error between the pursuer and the evader $\xi = [\xi_1, \xi_2, \dot{\xi}_1, \dot{\xi}_2]^T$ whose dynamics is given in (6.9). The time-to-capture problem is equivalent to the following optimization problem:

$$\begin{array}{ll} \min_{u_1^{\text{p}}(t), u_2^{\text{p}}(t)} & T \\ \text{subject to} & \left\{ \begin{array}{l} \xi(t+\delta) = A_\delta \xi(t) + G_\delta u^{\text{p}}(t) \\ |u_1^{\text{p}}(t)| \leq U_{\text{p}}, \;\; |u_2^{\text{p}}(t)| \leq U_{\text{p}} \\ \xi(t+T) = 0. \end{array} \right. \end{array} \tag{6.12}$$

Recently, Gao *et al.* [Gao, 2004] solved the previous problem as an application of minimum-time control for the discretized double integrator. An extension to minimum-time control for the discretized triple integrator is also available [Zanasi and Morselli, 2003]. Despite its simplicity and apparent efficacy, minimum-time control is rarely used in practice, since it is highly sensitive to small measurement errors and external disturbances. Although, in principle, minimum-time control gives the best performance, it needs to be modified to cope with practical issues such as the quantization of inputs, measurement and process noise, and modeling errors. We propose an approach that adds robustness while preserving the optimality of minimum-time control.

Since the state error dynamics is decoupled along the x and y-axes, the solution of the optimization problem (6.12) can be obtained by solving two independent minimum-time problems along each axis. When $\delta \to 0$ in (6.9), the minimum-time control problem restricted to one axis reduces to the well known minimum-time control problem of a double integrator in continuous time, which can be found in many standard textbooks on optimal control such as [Lee and Markus, 1967; Ryan, 1982]. The solution is given by a bang-bang control law and can be written in state feedback form as follows:

$$u_1^{\mathrm{p}} = \begin{cases} -U_{\mathrm{p}} & \text{If } \; 2U_{\mathrm{p}}\dot{\xi}_1 > -\xi_1|\xi_1| \\ +U_{\mathrm{p}} & \text{If } \; 2U_{\mathrm{p}}\dot{\xi}_1 < -\xi_1|\xi_1| \\ -U_{\mathrm{p}}\,\mathrm{sign}(\xi_1) & \text{If } \; 2U_{\mathrm{p}}\dot{\xi}_1 = -\xi_1|\xi_1| \\ 0 & \text{If } \; \dot{\xi}_1 = \xi_1 = 0. \end{cases} \tag{6.13}$$

The minimum time required to drive ξ_1 to zero in the x-axis can be also written in terms of the position and velocity error as follows:

$$T_{\mathrm{c},1}(\xi_1, \dot{\xi}_1) = \begin{cases} \frac{-\dot{\xi}_1 + \sqrt{2\dot{\xi}_1^2 - 4U_{\mathrm{p}}\xi_1}}{U_{\mathrm{p}}} & \text{if } \; 2U_{\mathrm{p}}\dot{\xi}_1 \geq -\xi_1|\xi_1| \\ \frac{\dot{\xi}_1 + \sqrt{2\dot{\xi}_1^2 + 4U_{\mathrm{p}}\xi_1}}{U_{\mathrm{p}}} & \text{otherwise.} \end{cases} \tag{6.14}$$

Figure 6.6 shows the switching curve $2U_{\mathrm{p}}\dot{\xi}_1 = -\xi_1|\xi_1|$ and the level curves of the time-to-capture T_{c} for different values.

Similar equations can be written for the control u_2^{p} along the y-axis. Therefore the minimum time-to-capture is given by:

$$T_{\mathrm{c}} = \max(T_{\mathrm{c},1}, T_{\mathrm{c},2}) \tag{6.15}$$

According to the previous analysis, given the state error $\xi(t)$ at current time t, we can compute the corresponding constant velocity time-to-capture T_{c}, the optimal input sequence $u^{\mathrm{p}*}(t')$ and the optimal trajectory $\xi^*(t')$ for $t' \in [t, t+T_{\mathrm{c}}]$.

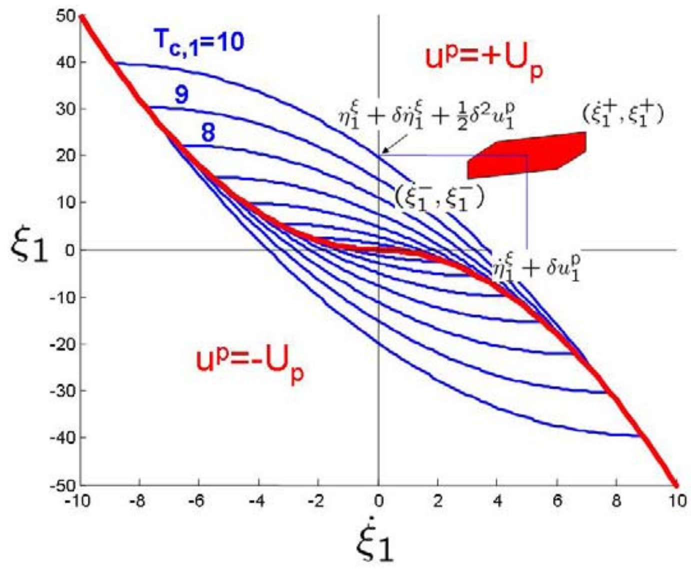


Figure 6.6: Optimal switching curve for the continuous minimum-time control of the double integrator (*thick solid line*) and curves of constant time-to-capture (*thin solid lines*) in the phase space $(\xi_1, \dot{\xi}_1)$. The hexagon represents the set of all possible locations of the true error state $(\xi_1(t+\delta), \dot{\xi}_1(t+\delta))$ at the next time step $t+\delta$ given measurement $(\eta_1, \dot{\eta}_1)$ and pursuer control input u_1^p at time t.

However, the optimal input (6.13) is the solution when $\delta \to 0$ in (6.9) with no measurement errors and no change in the evader's trajectory. In order to add robustness to take into account the quantization in the digital implementation, the measurement errors, and the evasive maneuvers of the evader, we analyze how the time-to-capture can be affected by these terms. Let us first rewrite the error dynamics given by (6.9) explicitly for the x-axis:

$$\begin{aligned}
\xi_1(t+\delta) &= \xi_1(t) + \delta\,\dot{\xi}_1(t) + \tfrac{1}{2}\delta^2 u_1^{\mathrm{p}}(t) + \tfrac{1}{2}\delta^2 u_1^{\mathrm{e}}(t) \\
\dot{\xi}_1(t+\delta) &= \dot{\xi}_1(t) + \delta\, u_1^{\mathrm{p}}(t) + \delta u_1^{\mathrm{e}}(t) \\
\eta_1^\xi(t) &= \xi_1(t) + v_1^\xi(t) \\
\dot{\eta}_1^\xi(t) &= \dot{\xi}_1(t) + \dot{v}_1^\xi(t)
\end{aligned}$$

If we substitute the last two equations into the first two we get:

$$\xi_1(t+\delta) = \eta_1^\xi(t) + \delta\dot{\eta}_1^\xi(t) + \frac{1}{2}\delta^2 u_1^{\mathrm{p}}(t) - K - v_1^\xi(t) - \delta\dot{v}_1^\xi(t) + \frac{1}{2}\delta^2 u_1^{\mathrm{e}}(t) \quad (6.16)$$

$$\dot{\xi}_1(t+\delta) = \dot{\eta}_1^\xi(t) + \delta\, u_1^{\mathrm{p}}(t) - \dot{v}_1^\xi(t) + \delta u_1^{\mathrm{e}}(t) \quad (6.17)$$

where $(\eta_1^\xi, \dot{\eta}_1^\xi)$ are output estimates from the MTF module, u_1^{p} is the controllable input, and $(u_1^{\mathrm{e}}, v_1^\xi, \dot{v}_1^\xi)$ play the role of external disturbances. Our goal now is to choose u_1^{p}, *i.e.*, the thrust of the pursuer, in such a way as to minimize the time-to-capture under the worst possible choice of $(u_1^{\mathrm{e}}, v_1^\xi, \dot{v}_1^\xi)$, which are not known in advance but are bounded. Figure 6.6 illustrates this approach graphically: the hexagon in the figure represents the possible position of the true state error $(\xi_1, \dot{\xi}_1)$ at the next time step $t+\delta$ which accounts for all possible evasive maneuvers of the evader, *i.e.*, $|u_1^{\mathrm{e}}| < U_e$, and accounts for the estimation errors on the position and velocity of the pursuer and the evader, *i.e.*, $|v_1^\xi| < V_1, |\dot{v}_1^\xi| < \dot{V}_1$, for a given choice of u_1^{p}. Since the center of the hexagon $(\eta_1^\xi + \delta\dot{\eta}_1^\xi + \frac{1}{2}\delta^2 u_1^{\mathrm{p}}, \dot{\eta}_1^\xi + \delta u_1^{\mathrm{p}})$ depends on the pursuer control u_1^{p}, one could try to choose u_1^{p} in such a way that the largest time-to-capture $T_{\mathrm{c},1}$ of the hexagon is minimized. This approach is common in the literature for non-cooperative games [Basar and Olsder, 1995]. More formally, the feedback control input will be chosen based on the following min-max optimization problem

$$u_1^{\mathrm{p}*}(t) = \arg \min_{|u_1^{\mathrm{p}}|\le U_{\mathrm{p}}} \left(\max_{\substack{|v_1^\xi|\le V_1, |\dot{v}_1^\xi|\le \dot{V}_1, \\ |u_1^{\mathrm{e}}|\le U_{\mathrm{e}}}} T_{\mathrm{c},1}\big(\xi_1(t+\delta), \dot{\xi}_1(t+\delta)\big) \right) \quad (6.18)$$

This is, in general, a nonlinear optimization problem. However, thanks to the specific structure of the time-to-capture function $T_{\mathrm{c},1}$, it is possible to show that (6.18) is equivalent to:

$$\begin{aligned} u_1^{\mathrm{p}*} &= \arg\min_{|u_1^{\mathrm{p}}|\le U_{\mathrm{p}}} \max\left(T_{\mathrm{c},1}(\xi_1^+,\dot{\xi}_1^+), T_{\mathrm{c},1}(\xi_1^-,\dot{\xi}_1^-)\right) \\ \xi_1^\pm &:= \eta_1^\xi + \delta\dot{\eta}_1^\xi \pm V_1 \pm \delta\dot{V}_1 \pm \frac{1}{2}\delta^2 U_{\mathrm{e}} + \frac{1}{2}\delta^2 u_1^{\mathrm{p}} \\ \dot{\xi}_1^\pm &:= \dot{\eta}_1^\xi \pm \dot{V}_1 \pm \delta U_{\mathrm{e}} + \delta u_1^{\mathrm{p}}, \end{aligned} \tag{6.19}$$

i.e., it is necessary to compute only the time-to-capture of the top right and the bottom left corner of the hexagon in Figure 6.6 since all points inside the set always have smaller values of $T_{\mathrm{c},1}$. Once the expected minimum time-to-capture control input $u^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ is computed, then the corresponding optimal trajectory for the pursuer $x^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ can be easily obtained by substituting $u^{\mathrm{p}*}(t')$ into the pursuer dynamics (6.4). The robust minimum-time path planning algorithm is summarized in Algorithm 14.

Algorithm 14 Robust Minimum-Time Path Planning

```
Input: $x^{\mathrm{p}}(t), x^{\mathrm{e}}(t)$, and bounds $V_1, V_2, \dot{V}_1, \dot{V}_2, U_{\mathrm{e}}, U_{\mathrm{p}}$
Output: optimal trajectory $x^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$
1: compute $u^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ using (6.19)
2: compute $x^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ given $u^{\mathrm{p}*}(t')$ using (6.4)
```

Figure 6.7 shows the performance of the proposed robust minimum time-to-capture control feedback for a scenario where the evader moves with random motion and the evader's position and velocity estimates are noisy. It is compared with the discrete-time minimum-time controller proposed in [Zanasi and Morselli, 2003] and [Gao, 2004]. Our controller feedback design outperforms the discrete-time minimum-time controller since the latter one does not take into account process and measurement noises. Note how both controllers do not direct pursuers toward the actual position of evader, but to the estimated future location of the evader to minimize the time-to-capture.

As introduced in Section 6.2.2, given the positions and velocities of all pursuers and evaders and bounds on the measurement error and evader input, it is possible to compute

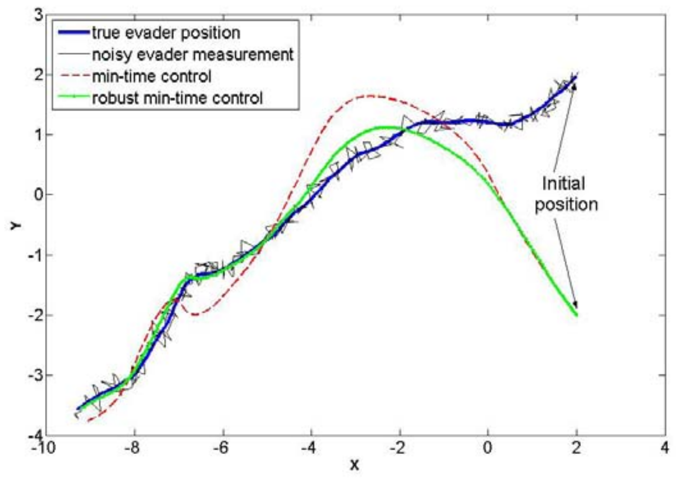


Figure 6.7: Trajectories of pursuers and evaders on the x-y plane. The feedback control is based on noisy measurements (*thin solid line*) of the true evader positions (*thick solid line*). The robust minimum time-to-capture feedback proposed in this chapter (*dot-solid line*) is compared with the discrete-time minimum time-to-capture feedback (*dashed line*) proposed in [Zanasi and Morselli, 2003].

the expected time-to-capture matrix $C = [c_{ij}] \in \mathbb{R}^{N_\mathrm{p} \times N_\mathrm{e}}$ using the solution to the optimal minimum-time control problem. The entry c_{ij} of the matrix C corresponds to the expected time for pursuer i to capture evader j, $T_\mathrm{c}(i,j)$, that can be computed as described in (6.14) and (6.15). As motivated in Section 6.2.2, we assume the same number of pursuers as the number of evaders, *i.e.*, $N_\mathrm{p} = N_\mathrm{e} = N$.

An assignment can be represented as a matrix $\Phi = [\phi_{ij}] \in \mathbb{R}^{N \times N}$, where the entry ϕ_{ij} of the matrix Φ is equal to 1 if pursuer i is assigned to evader j, and equal to 0 otherwise. The assignment problem can therefore be written formally as follows:

$$\begin{array}{ll} \min_{\phi_{ij} \in \{0,1\}} & \max_{i,j=1,\ldots,N} (c_{ij} \cdot \phi_{ij}) \\ \text{subject to} & \sum_{i=1}^{N} \phi_{ij} = 1, \ \forall i \\ & \sum_{j=1}^{N} \phi_{ij} = 1, \ \forall j. \end{array} \tag{6.20}$$

As formulated in (6.20), the assignment problem is a combinatorial optimization problem.

The optimization problem given in (6.20) can be reformulated as a *linear bottleneck assignment* problem and can be solved by any of the polynomial-time algorithms based on network flow theory. Here we give a brief description of one algorithm and we direct the interested reader to the survey [Burkard and Çela, 1998] for a detailed review of these algorithms. For our implementation, we use a randomized threshold algorithm that alternates between two phases. In the first phase, we list the cost elements c_{ij} in increasing order and we choose a cost element c^*, *i.e.*, a threshold. Then we construct the matrices $\bar{C}(c^*) = [\bar{c}_{ij}] \in \mathbb{R}^{N\times N}$ and $C_{\text{Tutte}}(c^*) \in \mathbb{R}^{2N\times 2N}$ as follows:

$$\bar{c}_{ij} = \begin{cases} a_{ij} & \text{if} \quad c_{ij} > c^* \\ 0 & \text{if} \quad c_{ij} \le c^* \end{cases}, \quad C_{\text{Tutte}} = \begin{bmatrix} 0 & \bar{C} \\ -\bar{C} & 0 \end{bmatrix} \tag{6.21}$$

where a_{ij}'s are independent random numbers sampled from a uniform distribution in the interval $[0,1]$, *i.e.*, $a_{ij} \sim \mathcal{U}([0,1])$. Using Tutte's Theorem [Burkard and Çela, 1998], it is possible to show that if $\det(C_{\text{Tutte}}(c^*)) \neq 0$, then there exists an assignment that achieves c^*[1]. Therefore, we search for the smallest c^*_{min} in the ordered list of costs c_{ij} which guarantees an assignment. Once we find c^*_{min}, we find the pursuer-evader pair corresponding to that cost. Then, we remove its row and column from the cost matrix C and repeat the procedure until all pursuers are assigned. The assignment algorithm is summarized in Algorithm 15.

It is important to note that an assignment based on the solution to the global optimization problem described above is necessary for good performance. For example, let us consider the *greedy assignment* algorithm. This algorithm looks for the smallest time-to-capture entry in the matrix C, assigns the corresponding pursuer-evader pair, and removes the corresponding row and column from the matrix C. The dimensions of the resulting

[1] In reality, since the algorithm is randomized, there is a small probability equal to $(1/N)^r$ that there exists a feasible assignment if $\det(C_{\text{Tutte}}) = 0$ for r random Tutte's matrices C_{Tutte}. In the rare cases when this event happens, the algorithm simply gives a feasible assignment with a higher cost to capture.

Algorithm 15 Pursuers-to-evaders Assignment

```
Input: $x_i^{\text{p}}, x_j^{\text{e}},\ i,j = 1,\dots,N$
Output: assignment $(i \to j)$ for $i = 1,\dots,N$
1: compute matrix $C = [c_{ij}]$, $c_{ij} = T_{\text{c}}(i,j)$
2: for $n = 1$ to $N$ do
3:     $[i^*, j^*] = \text{argmin}_{ij}\{c_{ij} \mid \det(C_{\text{Tutte}}(c_{ij})) \neq 0\}$, using (6.21)
4:     assign pursuer $i^*$ to evader $j^*$, i.e., $(i^* \to j^*)$
5:     $C \leftarrow \{C \,|\, \text{remove row } i^* \text{ and column } j^*\}$
6: end for
```

matrix C become $(N-1) \times (N-1)$ and the algorithm repeats the same process until each pursuer is assigned to an evader. This algorithm is very simple and can be implemented in a fully distributed fashion. However, it is a suboptimal algorithm since there are cases where the greedy assignment finds the worst solution. Consider the time-to-capture matrix $C = \begin{bmatrix} 1 & 2 \\ 3 & 100 \end{bmatrix}$. The optimal assignment that minimizes the time-to-capture of all evaders for this matrix is $(1 \to 2)$ and $(2 \to 1)$, which gives $T_{\text{c,max}} = 3$, where $T_{\text{c,max}}$ is the time-to-capture of all evaders. The greedy assignment would instead assign pursuer 1 to evader 1 and pursuer 2 to evader 2, with the time-to-capture of all evaders equal to $T_{\text{c,max}} = 100$.

6.4 Simulation Results

For simulations below, we consider the surveillance over a rectangular region on a plane, $\mathcal{R} = [0, 100]^2$. The state vector is $x = [x_1, x_2, \dot{x_1}, \dot{x_2}]^T$ where (x_1, x_2) is a position in $\mathcal{R}$ along the usual x and y axes and $(\dot{x_1}, \dot{x_2})$ is a velocity vector. The linear dynamics and measurement models (4.19) are used, where $Q = \text{diag}(.15^2, .15^2)$ and R is set according to Figure 6.3). We assume a 100×100 sensor grid, in which the separation between sensors is normalized to 1. So the unit length in simulation is the length of the sensor separation. In all simulations, $n_{\text{mc}} = 1000$, and $w_{\text{s}} = 10$. For the sensor model, we use $\alpha = 2$, $\gamma = 1$,

$\eta = 2$, and $\beta = 3(1 + \gamma R_s^\alpha)$.

Since the number of targets K is not fixed, it is difficult to measure the performance of an algorithm using a standard criterion such as the mean square error. Hence, we use two separate metrics to measure performance: the estimation error in the number of targets ϵ_K and the estimation error in position ϵ_X. Let $K^*(t)$ be the number of targets at time t and $K(t)$ be the estimated number of targets at time t. We define ϵ_K as

$$\epsilon_K = \frac{1}{\sum K^*(t)} \sum_{t=1}^{T} |K(t) - K^*(t)|. \tag{6.22}$$

The computation of ϵ_X is done when it makes sense. At any t, there can be at most $M(t) = \min(K(t), K^*(t))$ common tracks. We find $M(t)$ matches between true tracks and estimated tracks based on positions at $t-1, t, t+1$. For each match i, let $x_i^*(t)$ and $x_i(t)$ be the position of the true track and the estimated track at time t, respectively. We define ϵ_X as

$$\epsilon_X^2 = \frac{1}{\sum M(t)} \sum_{t=1}^{T} \sum_{i=1}^{M(t)} \|x_i(t) - x_i^*(t)\|^2. \tag{6.23}$$

Both ϵ_K and ϵ_X are normalized with respect to the number of targets for easier comparison.

We first evaluate the effect of the sensing range and empirically find that there is an optimal value at which the estimation error is minimized. Then we illustrate the robustness of our algorithm against sensor localization error, transmission failures and communication delays. We then give an example of surveillance with sensor networks and demonstrate how the hierarchical MCMCDA algorithm works.

6.4.1 Sensing Range

When localizing a single target, we can minimize the localization error by allowing more sensors to collaborate, which is equivalent to increasing R_s as shown in Figure 6.3. But when there is more than one target, this is no longer true, since observations from dif-

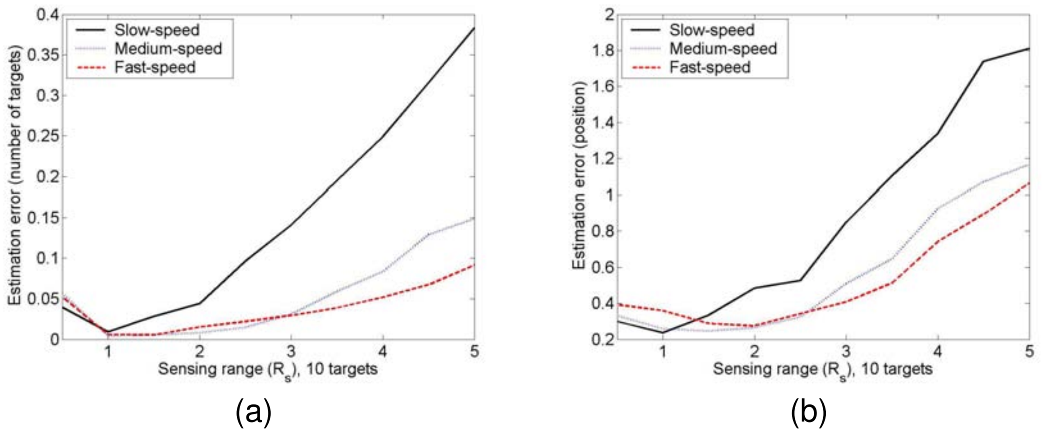


Figure 6.8: (a) Estimation error in K, ϵ_K, as a function of sensing range R_s. (b) Estimation error in X, ϵ_X, as a function of sensing range R_s. (Unity corresponds to the separation between sensors.)

ferent targets can collide, giving missing observations and observations away from target positions. Figure 6.8 shows the estimation errors ϵ_K and ϵ_X when 10 targets appear and disappear at random times and $T = 50$. For slow-speed vehicles, $|\dot{x_1}|, |\dot{x_2}| \in [0, 1]$; $|\dot{x_1}|, |\dot{x_2}| \in [1, 2]$ for medium-speed vehicles; and $|\dot{x_1}|, |\dot{x_2}| \in [2, 5]$ for high-speed vehicles. For each vehicle type, five different scenarios are used. When $R_\text{s} = .5$, the sensors do not completely cover the surveillance region $\mathcal{R}$ and do not detect targets at all times, hence the estimation error is higher. But as we increase R_s beyond 1.0, estimation errors increase, since there are more collisions among observations of different targets. The estimation errors are low for high-speed vehicles since it is easier to disambiguate crossing targets. From Figure 6.8, we find that $R_\text{s} = 1.5$ is a good range for all types of vehicles and it is used in simulations below. Notice that when $R_\text{s} = 1.5$, for each sensor s_i, there are 28 neighboring sensors which are at most $2R_\text{s}$ away from s_i. We can also interpret this result in terms of sensor density for a fixed value of R_s. Hence, once the surveillance region is fully covered by sensors, a further increase in density does not improve the estimation error.

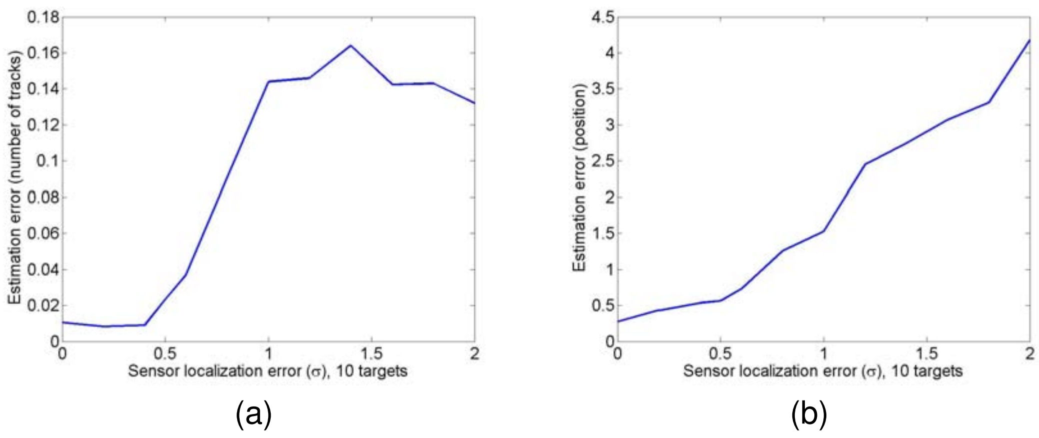


Figure 6.9: (a) Estimation error in K, ϵ_K, as a function of the localization error σ. (b) Estimation error in X, ϵ_X, as a function of the localization error σ.

6.4.2 Sensor Localization Error

The localization of sensor nodes in an ad-hoc wireless sensor network, without expensive hardware such as the global positioning system (GPS), is a challenging problem [Whitehouse *et al.*, 2004]. Hence, an algorithm which utilizes sensor positions needs to be robust against the sensor localization error. Suppose that the true position of sensor node i is s_i^* and $s_i = s_i^* + w_i^1$, where w_i^1 are Gaussian noises with zero mean and covariance $\Sigma = \text{diag}(\sigma^2, \sigma^2)$. Figure 6.9 shows the estimation errors from tracking 10 targets as functions of the sensor localization error σ. It shows that the algorithm is robust against the sensor localization error and, for $\sigma \leq .5$, the algorithm performs as if there is no sensor localization error. This is remarkable since $\sigma \leq .5$ corresponds to the case in which the average localization error is .707 times the separation between sensors. Notice that ϵ_K is always under .18, so the algorithm finds most tracks for all σ. But ϵ_X gets larger at high σ, since the target position estimation was based on incorrect node positions. Considering the fact that ϵ_X is computed from the norm of a vector in $\mathbb{R}^2$, ϵ_X is mostly due to the sensor localization error.

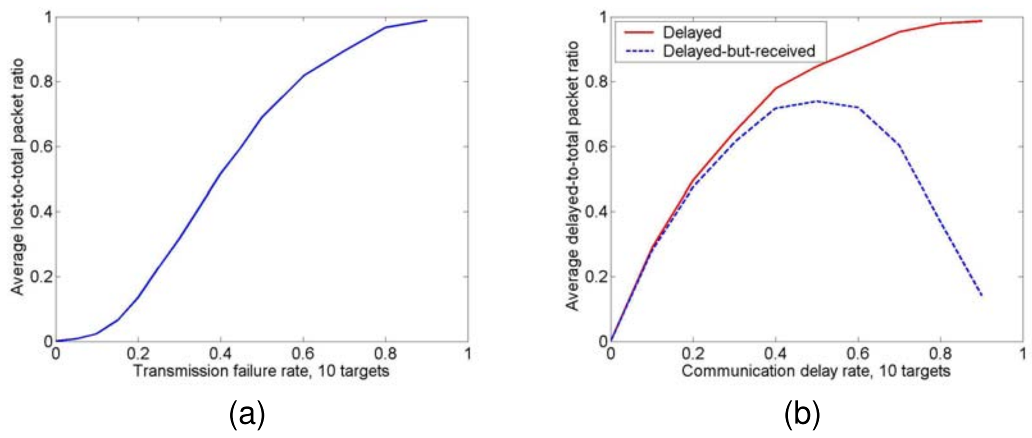
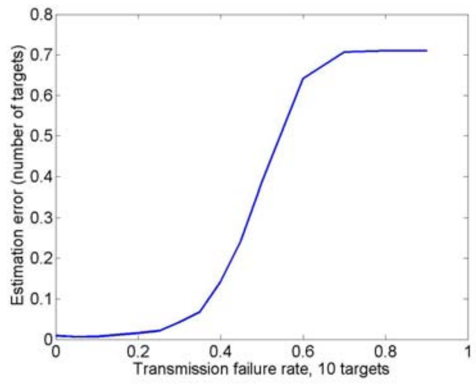


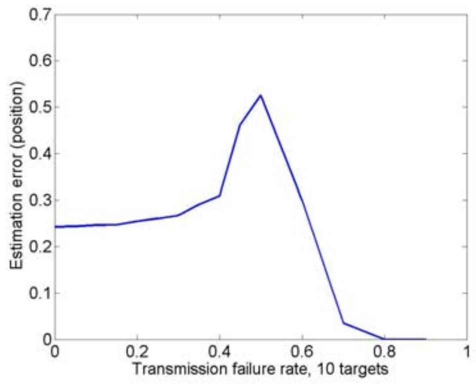
Figure 6.10: (a) Ratio between the number of lost packets and the number of total packets. (b) Ratio between the number of delayed packets and the number of total packets.

6.4.3 Transmission Failures

To assess the effects of transmission failures alone, we assume that there are no delayed observations, no false alarms, and no missing detections. A single supernode is placed at the center. As mentioned earlier, transmission failures are missing observations and Figure 6.10 (left) shows the ratio between the number of lost packets and the number of total packets as a function of the transmission failure rate p_{te}. As p_{te} increases, we lose more packets and, at $p_{\text{te}} \approx .9$, we lose all packets. Figure 6.11 shows the estimation errors and the algorithm performs well for $p_{\text{te}} \leq .4$. The estimation error ϵ_X is low at high p_{te} since most of packets are lost at high p_{te}, making the data association problem easier. Notice that when $p_{\text{te}} = .4$ more than 50% of packets are lost. It shows that our algorithm is very robust against transmission failures. Hence, for given p_{te}, we can find the maximum radius k_{max} of a tracking group, measured by the number of hops from a supernode, such that $p_{\text{ts}}(p_{\text{te}}, k_{\text{max}}) \geq .5$, sustaining the performance of the algorithm. For example, if $p_{\text{te}} = .1$, $k_{\text{max}} = 38$, *i.e.*, the most distant node can be 38 hops away from a supernode. But one must consider the communication delays which increases with the number of hops.



(a)



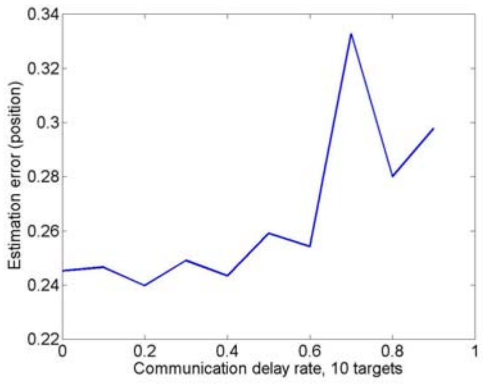
(b)

Figure 6.11: (a) Estimation error in K, ϵ_K, as a function of the transmission error p_{te}. (b) Estimation error in X, ϵ_X, as a function of the transmission error p_{te}.

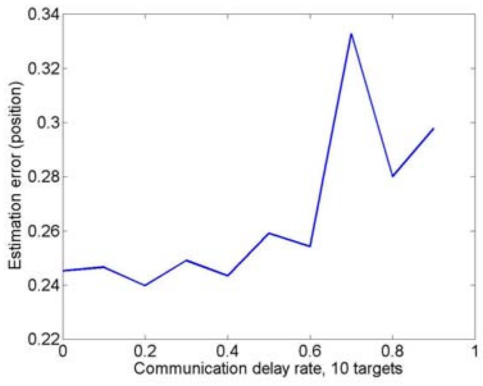
6.4.4 Communication Delays

As in the previous section, we assume that there are no transmission failures, no false alarms, and no missing observations. Figure 6.10 (right) shows the ratio between the number of delayed packets and the number of total packets as a function of the communication delay rate p_{de}. As p_{de} increases to 1, all packets are delayed. Since $w_s = 10$, we do not receive all the delayed packets and the ratio between the number of delayed packets that are eventually received and the number of packets is shown in a dotted line in Figure 6.10 (right). The estimations errors are shown in Figure 6.12. It shows a good performance for $p_{\text{de}} \leq .6$. At $p_{\text{de}} = .6$, 90% of packets are delayed and 72% of packets have delays less than w_s.

In summary, there is no performance loss up to an average localization error of 0.7 times the separation between sensors, and the algorithm tolerates up to 50% lost-to-total packet ratio and 90% delayed-to-total packet ratio.



(a)



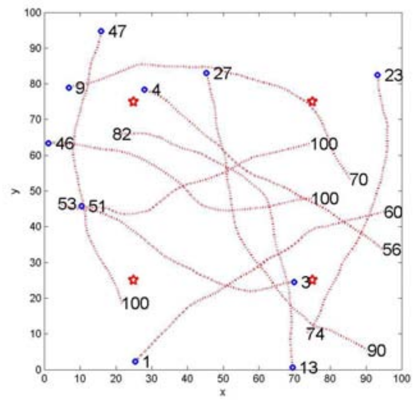
(b)

Figure 6.12: (a) Estimation error in K, ϵ_K, as a function of the communication delay p_{de}. (b) Estimation error in X, ϵ_X, as a function of the communication delay p_{de}.

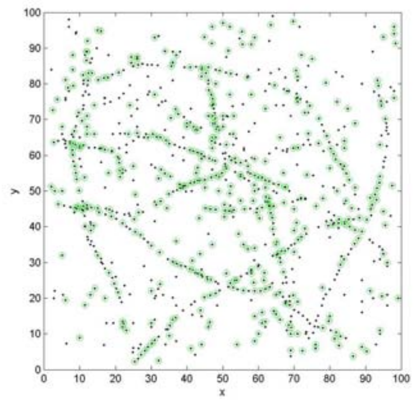
6.4.5 An Example of Surveillance with Sensor Networks

Here, we give a simulation example of surveillance using sensor networks. The surveillance region $\mathcal{R} = [0, 100]^2$ was divided into four quadrants and sensors in each quadrant formed a tracking group, where a Tier-2 node was placed at the center of each quadrant. The scenario is shown in Figure 6.13(a). We assumed a 100×100 sensor grid, in which the separation between sensors was normalized to 1. Thus, the unit length in simulation was the length of the sensor separation. For MCMCDA, $n_{\text{mc}} = 1000$ and $w_{\text{s}} = 10$. The signal-strength sensor model was used with parameters $\alpha = 2$, $\gamma = 1$, $\theta = 2$, and $\beta = 3(1 + \gamma R_{\text{s}}^{\alpha})$. In addition, $p_{\text{te}} = .3$ and $p_{\text{de}} = .3$. The surveillance duration was $T_{\text{s}} = 100$.

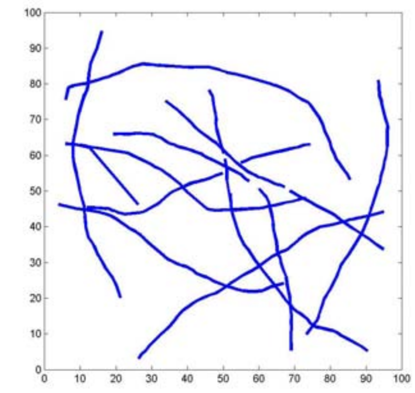
The state vector of a target is $x = [x_1, x_2, \dot{x_1}, \dot{x_2}]^T$ as described in Section 6.2.2. The simulation used the dynamic model in (6.4) and the evader control inputs were modeled by the random motion (6.7) with $q_{\text{e}} = .15^2$ and Q set according to Figure 6.3. Since the full



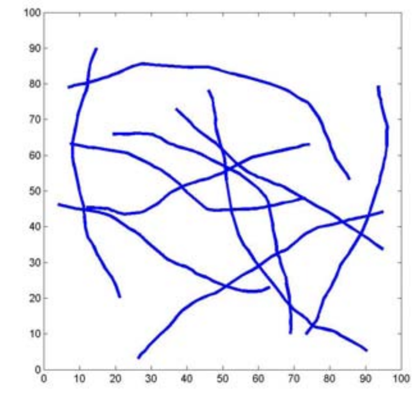
(a)



(b)



(c)



(d)

Figure 6.13: (a) Tracking scenario, where the numbers are target appearance and disappearance times, the initial positions are marked by circles, and the stars are the positions of Tier-2 nodes. (b) Accumulated observations received by Tier-2 nodes with delayed observations circled. (c) Tracks estimated locally by the MTT modules at Tier-2 nodes, superimposed. (d) Tracks estimated by the MTF module.

state is not observable, the measurement model (6.5) was modified as follows:

$$y(t) = Dx(t) + v(t), \quad \text{where } D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.24}$$

and y is a fused measurement computed by the MSF module in Section 6.3.1.

Figure 6.13(b) shows the observations received by the Tier-2 nodes. There were a total of 1174 observations and 603 of these observations were false alarms. A total of 319 packets out of 1174 packets were lost due to transmission failures and 449 packets out of 855 received packets were delayed. Figure 6.13(c) shows the tracks estimated locally by the MTT modules on the Tier-2 nodes while Figure 6.13(d) shows the tracks estimated by the MTF module using track-level data association. Figure 6.13(d) shows that the MTF module corrected mistakes made by Tier-2 nodes due to missing observations at the tracking group boundaries. The algorithm is written in C++ and MATLAB and run on PC with a 2.6-GHz Intel Pentium 4 processor. It takes less than 0.06 seconds per Tier-2 node, per simulation time step.

6.5 Experiments

Multi-target tracking and a pursuit evasion game using the control system *LochNess* were demonstrated at the Defense Advanced Research Projects Agency (DARPA) Network Embedded Systems Technology (NEST) final experiment on August 30, 2005. The experiment was performed under warm sunny conditions on a large-scale, long-term, outdoor sensor network testbed deployed on a short grass field at U.C. Berkeley's Richmond Field Station (see Figure 6.14). A total of 557 sensor nodes were deployed and 144 of these nodes were allotted for the tracking and PEG experiments. However, six out of the 144 nodes used in the experiment were not functioning on the day of the demo, reflecting the difficulties of deploying large-scale, outdoor systems.

The 144 nodes used for the tracking and PEG experiments were deployed at approximately 5 meter spacing in a 12×12 grid (see Figure 6.15). Each node was elevated using a camera tripod to prevent the passive infrared (PIR) sensors from being obstructed by grass and uneven terrain (see Figure 6.14(a)). The locations of the nodes were measured during

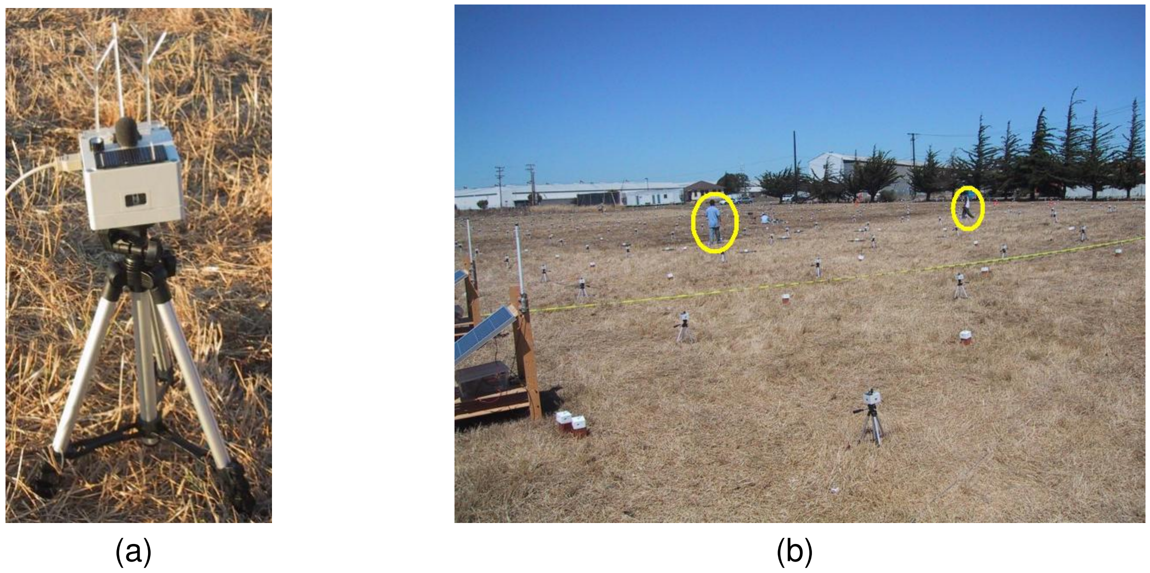


Figure 6.14: Hardware for the sensor nodes. (a) Trio sensor node on a tripod. On top is the microphone, buzzer, solar panel, and user and reset buttons. On the sides are the windows for the passive infrared sensors. (b) A live picture from the 2 target PEG experiment. The targets are circled.

deployment using differential GPS and stored in a table at the base station for reference and for generating Figure 6.15. However, in the experiments the system assumed the nodes were placed exactly on a 5 meter spacing grid to highlight the robustness of the system with respect to localization error.

The deployment of *LochNess* contained some modifications to the architecture described in Section 6.2. Due to the time constraint, the Tier-2 nodes were not fully functional on the day of the demo. Instead, we used a mote connected to a personal computer as the Tier-2 node. Only one such Tier-2 node was necessary to maintain connectivity to all 144 nodes used for the tracking experiment. In the experiment, simulated pursuers were used since it was difficult to navigate a ground robot in the field of tripods.

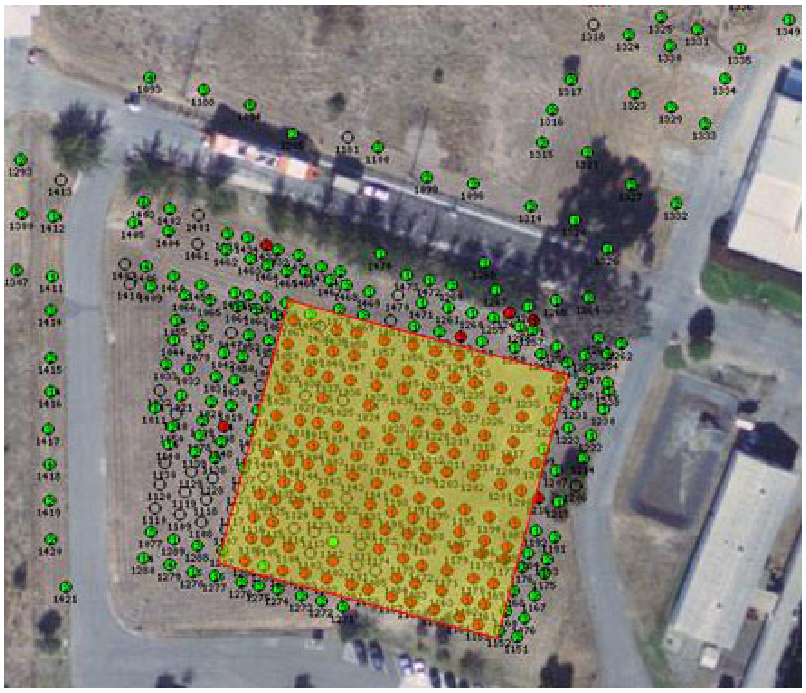
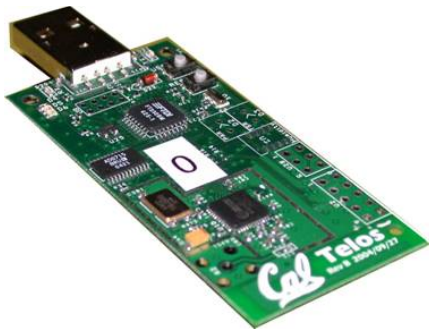


Figure 6.15: Sensor network deployment (not all deployed sensor nodes are shown). The disks and circles represent the positions of the sensor nodes. The network of 144 nodes used in the multi-target tracking and PEG experiments is highlighted.

6.5.1 Platform

A new sensor network hardware platform called the *Trio* mote was designed by cite-dutta:2006 for the outdoor testbed. The Trio mote is a combination of the designs of the *Telos B* mote, eXtreme Scaling Mote (XSM) sensor board [Dutta *et al.*, 2005], and Prometheus solar charging board [Jiang *et al.*, 2005], with improvements. Figure 6.16 shows the Trio node components and Figure 6.14(a) shows the assembled Trio node in a waterproof enclosure sitting on a tripod.

The Telos B mote [Polastre *et al.*, 2005] is the latest in a line of wireless sensor network platforms developed by U.C. Berkeley for the NEST project. It features an 8MHz Texas Instruments MSP430 microcontroller with 10kB of RAM and 48kB of program flash and a


(a)

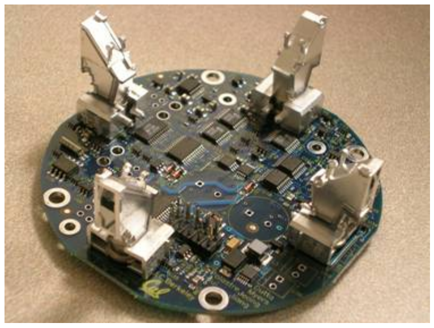

(b)

Figure 6.16: (a) Telos B. (b) Trio sensor board, based off the XSM sensor board and Prometheus solar power circuitry. See [Dutta *et al.*, 2006] for details.

250kbps, 2.4GHz, IEEE 802.15.4 standard compliant, Chipcon CC2420 radio. The Telos B mote provides lower power operation than previous motes (5.1 μA sleep, 19 mA on) and a radio range of up to 125 meters, making it the ideal platform for large-scale, long-term deployments.

The Trio sensor board includes a microphone, a piezoelectric buzzer, x-y axis magnetometers, and four passive infrared (PIR) motion sensors. For the multi-target tracking application, we found that the PIR sensors were the most effective at sensing human subjects moving through the sensor field. The magnetometer sensor had limited range even detecting targets with rare earth magnets and the acoustic sensor required complex signal processing to pick out the various acoustic signatures of a moving target from background noise. The PIR sensors provided an effective range of approximately 8 meters, with sensitivity varying depending on weather conditions and time of day. The variability in the signal strength of the PIR sensor reading prohibited extraction of ranging information from the sensor, so the PIR sensors were used as binary detectors.

The software running on the sensor nodes are written in NesC [Gay *et al.*, 2003] and run on TinyOS [TinyOS, 2006], an event-driven operating system developed for wireless embedded sensor platforms. The core sensor node application is the *DetectionEvent*

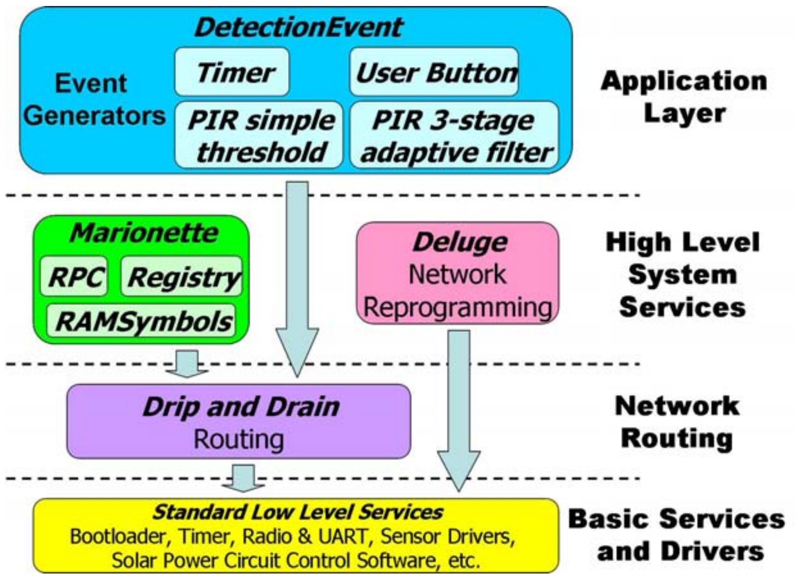

DetectionEvent
Event Generators
Timer
User Button
PIR simple threshold
PIR 3-stage adaptive filter
Application Layer
Marionette
RPC
Registry
RAMSymbols
Deluge Network Reprogramming
High Level System Services
Drip and Drain Routing
Network Routing
Standard Low Level Services
Bootloader, Timer, Radio & UART, Sensor Drivers, Solar Power Circuit Control Software, etc.
Basic Services and Drivers

Figure 6.17: Software services on the sensor network platform.

module, a multi-mode event generator for target detection and testing node availability. The sensor node application relies on a composition of various TinyOS subsystems and services that facilitate management and interaction with the network (see Figure 6.17). The core network management services are *Deluge* for network reprogramming [Hui and Culler, 2004] and *Marionette* for fast reconfiguration of parameters on the nodes [Whitehouse *et al.*, 2006]. The *DetectionEvent* application relies on the *Drip* and *Drain* routing layer for dissemination of commands and collection of data [Tolle, 2005]. For more details on the software architecture used in the outdoor testbed, see [Dutta *et al.*, 2006; Whitehouse *et al.*, 2006].

The *DetectionEvent* module provides four modes of event generation from the node – events generated periodically by a timer; events generated by pressing a button on the mote; events generated by the raw PIR sensor value crossing a threshold; and events generated by a three-stage filtering, adaptive threshold, and windowing detection algorithm for the PIR sensor signal developed by the University of Virginia [Gu *et al.*, 2005]. The timer generated events were parsed and displayed at the base station to help visualize which nodes in the network were alive. The three-stage PIR detection filter code was used during the development cycle. While it had potential to be more robust to different environmental conditions, during the day of the demo we reverted to the simple threshold PIR detector because the simple threshold detector was easy to tune and performed well.

The algorithms for the MSF, MTT, MTF, and MAC modules are all written in MATLAB and C++ and run on the base station in real-time. The same implementation of the tracking algorithm and the robust minimum time controller used in the simulations shown in Figure 6.13 and Figure 6.7 are used in the experiments. The data was timestamped at the base station.

6.5.2 Live Demonstration

The multi-target tracking algorithm was demonstrated on one, two, and three human targets, with targets entering the field at different times. In all three experiments, the tracking algorithm correctly estimated the number of targets and produced correct tracks. Furthermore, the algorithm correctly disambiguated crossing targets in the two and three target experiments without classification labels on the targets, using the dynamic models and target trajectories before crossing to compute the tracks.

Figure 6.18 shows the multi-target tracking results with three targets walking through the field. The three targets entered and exited the field around time 10 and 80, respectively. During the experiment, the algorithm correctly rejected false alarms and compensated for missing detections. There were many false alarms during the span of the experiments, as can be seen from the false alarms before time 10 and after time 80 in Figure 6.19. Also, though not shown in the figures, the algorithm dynamically corrected previous track hypotheses as it received more sensor readings. Figure 6.19 also gives a sense of the irregularity of network traffic. The spike in traffic shortly after time 50 was approximately when

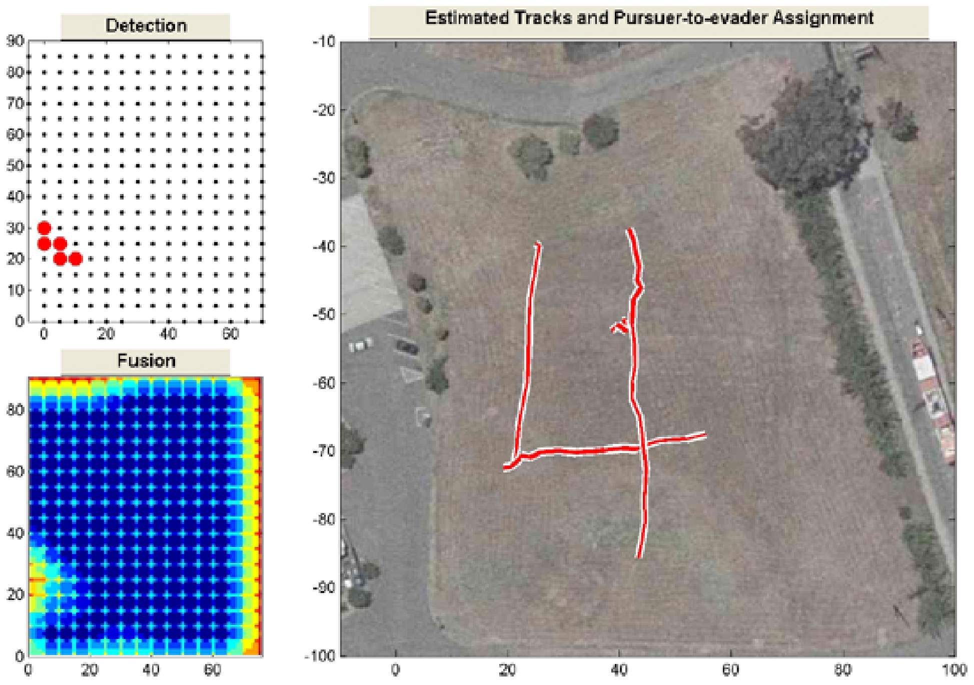


Figure 6.18: Estimated tracks of targets at time 70 from the experiment with three people walking in the field. (upper left) Detection panel. Sensors are marked by small dots and detections are shown in large disks. (lower left) Fusion panel shows the fused likelihood. (right) Estimated Tracks and Pursuer-to-evader Assignment panel shows the tracks estimated by the MTT module, estimated evader positions (stars) and pursuer positions (squares).

two of the targets crossed. It shows that the multi-target tracking algorithm is robust against missing measurements, false measurements, and the irregularity of network traffic.

In the last demonstration, two simulated pursuers were dispatched to chase two crossing human targets. The pursuer-to-target assignment and the robust minimum time-to-capture control law were computed in real-time, in tandem with the real-time tracking of the targets. The simulated pursuers captured the human targets, as shown in Figure 6.20. In particular, note that the MTT module is able to correctly disambiguate the presence of two targets (right panel of Figure 6.20(a)) using past measurements, despite the fact that the MSF

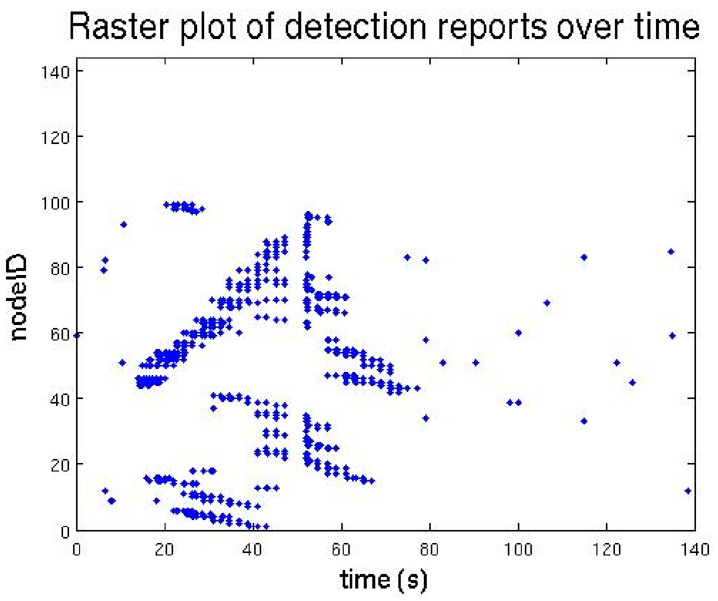


Figure 6.19: Raster plot of the binary detection reports from the three target tracking demo. Dots represent detections from nodes that were successfully transmitted to the base station.

module reports the detection of a single target (upper left panel of Figure 6.20(a)). A live picture of this experiment is shown on the right of Figure 6.14.

6.6 Summary

This chapter described *LochNess*, a hierarchical real-time control system for sensor networks. *LochNess* is applied to pursuit evasion games, in which a group of evaders are tracked using a sensor network and a group of pursuers are coordinated to capture the evaders. Although sensor networks provide global observability, they cannot provide high quality measurements in a timely manner due to packet loss, communication delay, and false detections. These factors have been the main challenge to developing a real-time control system using sensor networks.

We have proposed a possible solution for closing the loop around wireless ad-hoc sensor networks. The hierarchical real-time control system *LochNess* decouples the estimation of evader states from the control of pursuers by using multiple layers of data fusion, including the multi-sensor fusion (MSF) module, the multi-target tracking (MTT) module, and the multi-track fusion (MTF) module. While a sensor network generates noisy, inconsistent, and bursty measurements, the three layers of data fusion convert raw sensor measurements into fused measurements in a compact and consistent representation and forward the fused measurements to the pursuers' controllers in a timely manner.

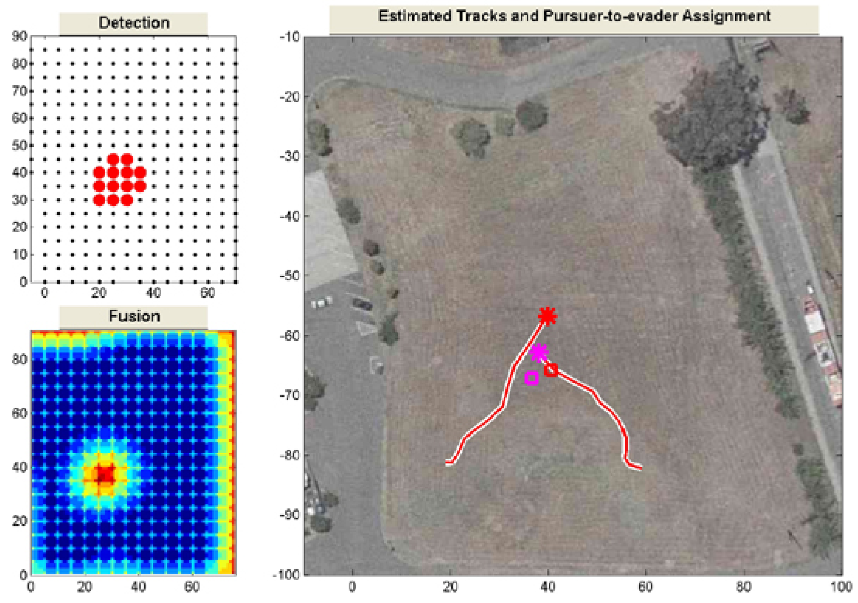
In order to coordinate multiple pursuers, the multi-agent coordination (MAC) module is developed. The assignments of pursuers to evaders are chosen such that the time to capture all evaders is minimized. The controllers for the pursuers are based on minimum-time control but were designed to account for the worst-case evader motions and to add robustness to the quantization of inputs, measurement and process noises, and modeling errors.

Simulation and experimental results have shown that *LochNess* is well suited for solving real-time control problems using sensor networks and that a sensor network is an attractive solution for the surveillance of a large area.

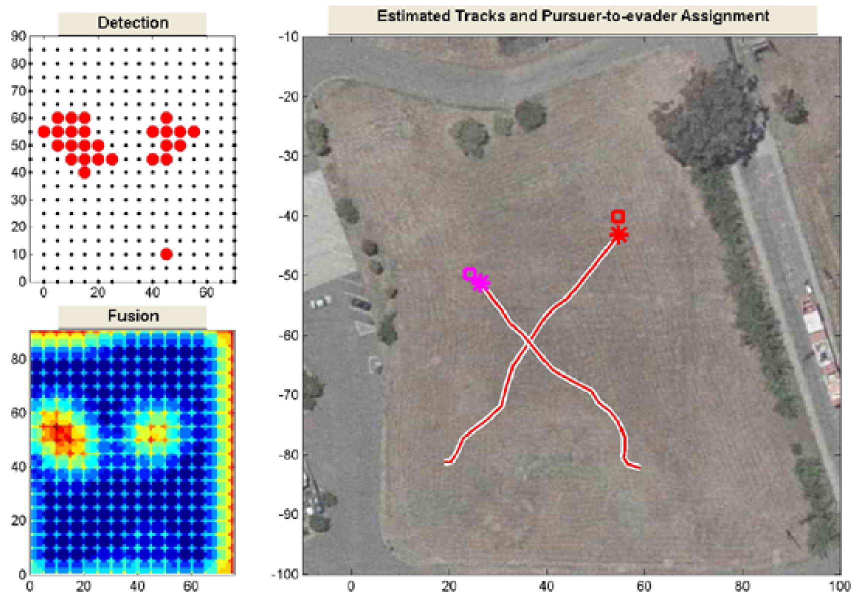
In this work, we assumed a stationary hierarchy, *i.e.*, the Tier-2 nodes and base station are fixed. However, a stationary hierarchy is not robust against malicious attacks. In our future work, we will address this issue by introducing redundancy, distributing the coordination tasks among Tier-2 nodes, and dynamically managing the hierarchy of the system. Our immediate goal is to quantify the robustness of the system against false measurements and packet loss and to identify the sensor network parameters such as maximum delay rate, maximum packet loss rate, and maximum false detection rate, necessary for seamless operation of the control system.

Acknowledgement

The experiments were made possible by contributions from David Culler, Prabal Dutta, Eric Fraser, Mike Howard, Jonathan Hui, Jaein Jeong, August Joki, Sukun Kim, Philip Levis, Michael Manzo, Joseph Polastre, Travis Pynn, Peter Ray, Tanya Roosta, Shawn Schaffert, Cory Sharp, Bruno Sinopoli, Jay Taneja, Gilman Tolle, David Shim, Robert Szewczyk, Kamin Whitehouse, and Bonnie Zhu.



(a)



(b)

Figure 6.20: Estimated tracks of evaders and pursuer positions from the pursuit evasion game experiment. (a) Before crossing. (b) After crossing.

CHAPTER 7

DISTRIBUTED NETWORKED CONTROL SYSTEMS

With the recent developments in communication, computing, and control systems, a networked control system (NCS) has received a fair amount of attention recently. In a general sense, an NCS consists of spatially distributed multiple systems or agents equipped with sensors, actuators, and computing and communication devices. The operation of each agent is coordinated over a communication network. The examples of an NCS includes sensor networks [Culler *et al.*, 2004; Oh *et al.*, 2007], networked autonomous mobile agents [Grocholsky *et al.*, 2004], *e.g.*, a team of UAVs, and arrays of micro or micro-electromechanical sensors (MEMS) devices.

Recently, different aspects of NCSs have been studied extensively. Sinopoli *et al.* [Sinopoli *et al.*, 2004] showed the phase transition behavior of the Kalman filter when the measurement packet loss is modeled by a Bernoulli random process and established the relationship between the speed of dynamics and the packet loss rate for the stable estimation of the system. Similar estimation problems are discussed in [Liu and Goldsmith, 2004; Xu and Hespanha, 2005; Shi *et al.*, 2005]. The control problems over an unreliable communication channel have been studied by many authors, including [Nilsson and Bernhardsson, 1997; Imer *et al.*, 2004; Sinopoli *et al.*, 2005]. The stability of NCSs has been also studied in [Zhang *et al.*, 2001; Liberzon, 2003].

There is a growing interest in consensus and coordination of networked systems inspired by the model by Vicsek *et al.* [Vicsek *et al.*, 1995], in which a large number of particles (or autonomous agents) move at a constant speed but with different headings. At each discrete time, each particle updates its heading based on the average heading of its neighboring particles. The analysis of the Vicsek model in different forms are reported in [Jadbabaie *et al.*, 2003; Blondel *et al.*, 2005; Olfati-Saber and Murray, 2004]. This chapter extends NCSs to model a distributed multi-agent system such as the Vicsek model.

In general, a single plant is assumed in an NCS and the links between the plant and the estimator or controller is closed by a common (unreliable) communication channel. This chapter extends this notion of NCSs by introducing a distributed networked control system (DNCS) consisting of multiple agents communicating over a lossy communication channel. The best examples of such system include ad-hoc wireless sensor networks and a network of mobile agents. We first consider the estimation problem appears in DNCSs and develop optimal linear filtering algorithms based on the Kalman filter. However, the time complexity of the exact method can be exponential in the number of possible communication link configurations. We address this issue by developing two approximate filtering algorithms for estimating states of a DNCS. The approximate filtering algorithms bound the state estimation error of the exact filtering algorithm and the time complexity of approximate methods is not dependent on the number of possible communication link configurations.

The stability of estimators under a lossy communication channel is studied in [Sinopoli *et al.*, 2004; Liu and Goldsmith, 2004]. However, the extension of the result to the general case with an arbitrary number of lossy communication links is unknown. Although we do not provide a definite answer for the general case, we derive a condition under which the state error covariance can be unbounded for the general case.

This chapter also considers the problem of finding a communication control which stabilizes a DNCS. This problem is called the *stabilizing communication control* problem and its goal is to find the acceptable ranges of packet loss rates at which the overall system is stable. We use the stability results for jump linear systems by Costa and Fragoso [Costa and Fragoso, 1993] to derive a sufficient condition for the stability of a DNCS. We then develop an efficient algorithm for checking the existence of stabilizing communication control using linear programming and discuss a method for solving the stabilizing communication control problem using geometric programming, a convex optimization method [Boyd and Vandenberghe, 2004; Boyd *et al.*, 2005].

7.1 Distributed Networked Control Systems with Lossy Links

Let us first consider a distributed control system consisting of N agents, in which there is no communication loss. The discrete-time linear dynamic model of the agent j can be described as following:

$$x_j(k+1) = \sum_{i=1}^{N} A_{ij} x_i(k) + G_j w_j(k) \tag{7.1}$$

where $k \in \mathbb{Z}^+$, $x_j(k) \in \mathbb{R}^{n_x}$ is the state of the agent j at time k, $w_j(k) \in \mathbb{R}^{n_w}$ is a white noise process, $A_{ij} \in \mathbb{R}^{n_x \times n_x}$, and $G_j \in \mathbb{R}^{n_x \times n_w}$. Hence, the state of the agent j is governed by the previous states of all N agents. We can also consider $A_{ij}x_i(k)$ as a control input from the agent i to the agent j for $i \neq j$.

Now consider a distributed networked control system (DNCS), in which agents communicate with each other over a lossy communication channel, *e.g.*, wireless channel. We assume an erasure channel between a pair of agents. At each time k, a packet sent by the agent i is correctly received by the agent j with probability p_{ij}. We form a communication matrix $P_{\text{com}} = [p_{ij}]$. Let $Z_{ij}(k) \in \{0, 1\}$ be a Bernoulli random variable, such that $Z_{ij}(k) = 1$ if a packet sent by the agent i is correctly received by the agent j at time k, otherwise, $Z_{ij}(k) = 0$. Since there is no communication loss within an agent, $p_{ii} = 1$ and $Z_{ii}(k) = 1$ for all i and k. For each (i, j) pair, $\{Z_{ij}(k)\}$ are i.i.d. (independent identically distributed) random variables such that $P(Z_{ij}(k) = 1) = p_{ij}$ for all k; and $Z_{ij}(k)$ are independent from $Z_{lm}(k)$ for $l \neq i$ or $m \neq j$. Then we can write the dynamic model of the agent j under lossy links as following:

$$x_j(k+1) = \sum_{i=1}^{N} Z_{ij}(k) A_{ij} x_i(k) + G_j w_j(k). \tag{7.2}$$

Let $x(k) = [x_1(k)^T, \ldots, x_N(k)^T]^T$ and $w(k) = [w_1(k)^T, \ldots, w_N(k)^T]^T$, where y^T is a transpose of y. Let $\bar{A}_{ij}$ be a $Nn_x \times Nn_x$ block matrix. The entries of $\bar{A}_{ij}$ are all zeroes except the (i, j)-th block is A_{ij}. For example, when $N = 2$

$$\bar{A}_{12} = \begin{bmatrix} \mathbf{0}_{n_x} & A_{12} \\ \mathbf{0}_{n_x} & \mathbf{0}_{n_x} \end{bmatrix}, \tag{7.3}$$

where $\mathbf{0}_{n_x}$ is a $n_x \times n_x$ zero matrix. Then the discrete-time linear dynamic model of the DNCS with lossy links can be represented as following:

$$x(k+1) = \left(\sum_{i=1}^{N} \sum_{j=1}^{N} Z_{ij}(k) \bar{A}_{ij} \right) x(k) + Gw(k), \tag{7.4}$$

where G is a block diagonal matrix of $G_1, \ldots, G_N$.

For notational convenience, we introduce a new index $n \in \{1, \ldots, N^2\}$ such that ij is indexed by $n = N(i-1) + j$. With this new index n, the dynamic model (7.4) can be rewritten as

$$x(k+1) = \left(\sum_{n=1}^{N^2} Z_n(k)\bar{A}_n\right) x(k) + Gw(k). \tag{7.5}$$

By letting $A(k) = \left(\sum_{n=1}^{N^2} Z_n(k)\bar{A}_n\right)$, we see that (7.5) is a time-varying linear dynamic model:

$$x(k+1) = A(k)x(k) + Gw(k). \tag{7.6}$$

Until now we have assumed that $\bar{A}_n$ is fixed for each n. We relax this assumption by letting $A(k) = A(Z(k))$, where $Z(k) = [Z_1(k), \ldots, Z_{N^2}(k)]^T$. This relaxed dynamical system is

$$x(k+1) = A(Z(k))x(k) + Gw(k). \tag{7.7}$$

The dynamic model (7.7) or (7.5) is a special case of the linear hybrid model or a jump linear system [Costa and Fragoso, 1993] since $A(k)$ takes an element from a set of a finite number of matrices. We will call the dynamic model (7.5) as the "simple" DNCS dynamic model and (7.7) as the "general" DNCS dynamic model.

7.2 Exact Kalman Filtering for DNCSs

In this section, we describe recursive filtering algorithms for the dynamic models (7.5) and (7.7) using the Kalman filter (KF). Since $Z(k)$ is independent from $Z(t)$ for $t \neq k$, we derive optimal linear filters for both cases. Notice that we denote $Z(k)$ by Z when there is no confusion.

7.2.1 KF for Simple DNCS

Consider the simple DNCS dynamic model (7.5), where $w(k)$ is a Gaussian noise with zero mean and covariance Q, and the following measurement model:

$$y(k) = Cx(k) + v(k), \tag{7.8}$$

where $y(k) \in \mathbb{R}^{n_y}$ is a measurement at time k, $C \in \mathbb{R}^{n_y \times Nn_x}$, and $v(k)$ is a Gaussian noise with zero mean and covariance R. Hence, we are assuming that the measurements are collected by a remote sensor or by a sensor in one of the agents. Notice that $Z(k)$ is not observed.

The following terms are defined to describe the modified Kalman filter.

$$\begin{aligned}
\hat{x}(k|k) &:= \mathbb{E}\left[x(k)|\mathbf{y_k}\right] && (7.9)\\
P(k|k) &:= \mathbb{E}\left[e(k)e(k)^T|\mathbf{y_k}\right] && (7.10)\\
\hat{x}(k+1|k) &:= \mathbb{E}\left[x(k+1)|\mathbf{y_k}\right] && (7.11)\\
P(k+1|k) &:= \mathbb{E}\left[e(k+1|k)e(k+1|k)^T|\mathbf{y_k}\right], && (7.12)
\end{aligned}$$

where $\mathbf{y_k} = \{y(t) : 0 \leq t \leq k\}$, $e(k|k) = x(k) - \hat{x}(k|k)$, and $e(k+1|k) = x(k+1) - \hat{x}(k+1|k)$.

Suppose that we have estimates $\hat{x}(k|k)$ and $P(k|k)$ from time k. At time $k+1$, a new measurement $y(k+1)$ is received and our goal is to estimate $\hat{x}(k+1|k+1)$ and $P(k+1|k+1)$ from $\hat{x}(k|k)$, $P(k|k)$, and $y(k+1)$. First, we compute $\hat{x}(k+1|k)$ and

$P(k+1|k)$.

$$\begin{aligned}\hat{x}(k+1|k) &= \mathbb{E}\left[x(k+1)|\mathbf{y_k}\right]\\ &= \mathbb{E}\left[\left(\sum_{n=1}^{N^2} Z_n(k)\bar{A}_n\right)x(k)+w(k)|\mathbf{y_k}\right]\\ &= \left(\sum_{n=1}^{N^2} p_n\bar{A}_n\right)\hat{x}(k|k)\\ &= \hat{A}\hat{x}(k|k), \qquad (7.13)\end{aligned}$$

where $p_n = P(Z_n(k)=1)$ and $\hat{A} = \sum_{n=1}^{N^2} p_n\bar{A}_n$. Let $A(k) = \sum_{n=1}^{N^2} Z_n(k)\bar{A}_n$. Then

$$\begin{aligned}P(k+1|k) &= \mathbb{E}\left[e(k+1|k)e(k+1|k)^T|\mathbf{y_k}\right]\\ &= \mathbb{E}[A(k)x(k)(A(k)x(k))^T|\mathbf{y_k}]\\ &- \hat{A}\hat{x}(k|k)(\hat{A}\hat{x}(k|k))^T + GQG^T \qquad (7.14)\end{aligned}$$

Since $\mathbb{E}[Z_n(k)Z_n(k)] = p_n$ and $\mathbb{E}[Z_n(k)Z_m(k)] = p_np_m$ for $m \neq n$,

$$\begin{aligned}\mathbb{E}[A(k)x(k)(A(k)x(k))^T|\mathbf{y_k}] &= \mathbb{E}[\sum_{n=1}^{N^2}\sum_{m=1}^{N^2} Z_n(k)Z_m(k)\bar{A}_n x(k)x(k)^T\bar{A}_m^T|\mathbf{y_k}]\\ &= \sum_{n=1}^{N^2}\sum_{m=1}^{N^2}\mathbb{E}[Z_n(k)Z_m(k)]\bar{A}_n\mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_m^T\\ &= \sum_{n=1}^{N^2} p_n\bar{A}_n\mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_n^T\\ &+ \sum_{n=1}^{N^2}\sum_{m=1,m\neq n}^{N^2} p_np_m\bar{A}_n\mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_m^T. \qquad (7.15)\end{aligned}$$

On the other hand,

$$\hat{A}\hat{x}(k|k)(\hat{A}\hat{x}(k|k))^T = \sum_{n=1}^{N^2}\sum_{m=1}^{N^2} p_n p_m \bar{A}_n \hat{x}(k|k)\hat{x}(k|k)^T \bar{A}_m^T. \tag{7.16}$$

Combining previous two results into the equation for $P(k+1|k)$, we get

$$\begin{aligned}
P(k+1|k) &= GQG^T + \sum_{n=1}^{N^2} p_n \bar{A}_n \mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_n^T \\
&+ \sum_{n=1}^{N^2} \sum_{m=1,m\neq n}^{N^2} p_n p_m \bar{A}_n \mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_m^T \\
&- \sum_{n=1}^{N^2}\sum_{m=1}^{N^2} p_n p_m \bar{A}_n \hat{x}(k|k)\hat{x}(k|k)^T \bar{A}_m^T \\
&= GQG^T + \sum_{n=1}^{N^2} p_n \bar{A}_n \mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_n^T \\
&- \sum_{n=1}^{N^2} p_n^2 \bar{A}_n \mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_n^T + \sum_{n=1}^{N^2}\sum_{m=1}^{N^2} p_n p_m \bar{A}_n P(k|k)\bar{A}_m^T \\
&= GQG^T + \sum_{n=1}^{N^2} p_n(1-p_n)\bar{A}_n \mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]\bar{A}_n^T \\
&+ \sum_{n=1}^{N^2}\sum_{m=1}^{N^2} p_n p_m \bar{A}_n P(k|k)\bar{A}_m^T.
\end{aligned} \tag{7.17}$$

We can also write it as

$$\begin{aligned}
P(k+1|k) &= GQG^T + \hat{A}P(k|k)\hat{A}^T \\
&+ \sum_{n=1}^{N^2} p_n(1-p_n)\bar{A}_n(P(k|k) + \hat{x}(k|k)\hat{x}(k|k)^T)\bar{A}_n^T.
\end{aligned} \tag{7.18}$$

Given $\hat{x}(k+1|k)$ and $P(k+1|k)$, $\hat{x}(k+1|k+1)$ and $P(k+1|k+1)$ are computed as in the regular Kalman filter.

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)(y(k+1) - C\hat{x}(k+1|k)) \quad (7.19)$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)CP(k+1|k), \quad (7.20)$$

where $K(k+1) = P(k+1|k)C^T(CP(k+1|k)C^T + R)^{-1}$.

7.2.2 KF for General DNCS

Now let us consider the general DNCS dynamic model (7.7) with the measurement model described in (7.8). We have

$$\begin{aligned} \hat{x}(k+1|k) &= \mathbb{E}\left[x(k+1)|\mathbf{y_k}\right] \\ &= \mathbb{E}\left[A(Z)x(k) + Gw(k)|\mathbf{y_k}\right] \\ &= \hat{A}\hat{x}(k|k), \end{aligned} \quad (7.21)$$

where

$$\hat{A} = \sum_{z \in \mathcal{Z}} p_z A(z) \quad (7.22)$$

is the expected value of $A(Z)$. Here, $p_z = P(Z = z)$, and $\mathcal{Z}$ is a set of all possible outcome vectors for Z, *i.e.*, $\mathcal{Z}$ is a set of all possible communication link configurations.

The prediction covariance can be computed as following.

$$\begin{aligned}
P(k+1|k) &= \mathbb{E}[e(k+1|k)e(k+1|k)^T|\mathbf{y_k}] \\
&= \mathbb{E}[A(Z)x(k)x(k)^T A(Z)^T|\mathbf{y_k}] - \hat{A}\hat{x}(k|k)\hat{x}(k|k)^T\hat{A}^T + GQG^T \\
&= \sum_{z\in\mathcal{Z}} p_z A(z)\mathbb{E}[x(k)x(k)^T|\mathbf{y_k}]A(z)^T - \hat{A}\hat{x}(k|k)\hat{x}(k|k)^T\hat{A}^T + GQG^T \\
&= GQG^T + \sum_{z\in\mathcal{Z}} p_z A(z)P(k|k)A(z)^T \\
&+ \sum_{z\in\mathcal{Z}} p_z A(z)\hat{x}(k|k)\hat{x}(k|k)^T(A(z)-\hat{A})^T. \qquad (7.23)
\end{aligned}$$

Lastly, $\hat{x}(k+1|k+1)$ and $P(k+1|k+1)$ are computed as shown in (7.19) and (7.20).

7.3 Approximate Kalman Filtering for DNCSs

The exact KF proposed in Section 7.2.2 for the general DNCS is an optimal linear filter but the time complexity of the algorithm can be exponential in N since the size of $\mathcal{Z}$ is $O(2^{N^2})$ in the worst case, *i.e.*, when all agents can communicate with each other. In this section, we describe two approximate Kalman filtering methods for the general DNCS dynamic model (7.7) which is computationally efficient than the exact KF by avoiding the enumeration over $\mathcal{Z}$. Since the computation of $P(k+1|k)$ is the only time-consuming process, we propose two filtering method which can bound $P(k+1|k)$. We use the notation $A \succ 0$ if A is a positive definite matrix and $A \succeq 0$ if A is a positive semidefinite matrix.

7.3.1 Lower-bound KF for General DNCS

The *lower-bound KF* (lb-KF) is the same as the exact KF described in Section 7.2.2, except we approximate $P(k+1|k)$ by $\underline{P}(k+1|k)$ and $P(k|k)$ by $\underline{P}(k|k)$. The covariances are updated as following:

$$\underline{P}(k+1|k) = \hat{A}\underline{P}(k|k)\hat{A}^T + GQG^T \tag{7.24}$$

$$\underline{P}(k+1|k+1) = \underline{P}(k+1|k) - \underline{K}(k+1)C\underline{P}(k+1|k), \tag{7.25}$$

where $\hat{A}$ is the expected value of $A(Z)$ and $\underline{K}(k+1) = \underline{P}(k+1|k)C^T(C\underline{P}(k+1|k)C^T + R)^{-1}$. Notice that $\hat{A}$ can be computed in advance and the lb-KF avoids the enumeration over $\mathcal{Z}$. The following theorem shows that the lb-KF maintains the state error covariance which is upper-bounded by the state error covariance of the exact KF.

Theorem 9. *If the lb-KF starts with an initial covariance* $\underline{P}(0|0)$*, such that* $\underline{P}(0|0) \preceq P(0|0)$*, then* $\underline{P}(k|k) \preceq P(k|k)$ *for all* $k \geq 0$.

Proof: See Appendix C.1. ■

7.3.2 Upper-bound KF for General DNCS

Similar to the lb-KF, the *upper-bound KF* (ub-KF) approximates $P(k+1|k)$ by $\bar{P}(k+1|k)$ and $P(k|k)$ by $\bar{P}(k|k)$. Let $\lambda_{\text{max}} = \lambda_{\text{max}}(\bar{P}(k|k)) + \lambda_{\text{max}}(\hat{x}(k|k)\hat{x}(k|k)^T)$, where $\lambda_{\text{max}}(S)$ denotes the maximum eigenvalue of S. The covariances are updated as following:

$$\bar{P}(k+1|k) = \lambda_{\text{max}}\mathbb{E}[A(Z)A(Z)^T] - \hat{A}\bar{x}(k|k)\bar{x}(k|k)^T\hat{A}^T + GQG^T \tag{7.26}$$

$$\bar{P}(k+1|k+1) = \bar{P}(k+1|k) - \bar{K}(k+1)C\bar{P}(k+1|k), \tag{7.27}$$

where $\hat{A}$ is the expected value of $A(Z)$ and $\bar{K}(k+1) = \bar{P}(k+1|k)C^T(C\bar{P}(k+1|k)C^T + R)^{-1}$. In the ub-KF, $\mathbb{E}[A(Z)A(Z)^T]$ can be computed in advance but we need to compute λ_{max} at each step of the algorithm. But if the size of $\mathcal{Z}$ is large, it is more efficient than the exact KF. (Notice that the computation of λ_{max} requires a polynomial number of operations in N while the size of $\mathcal{Z}$ can be exponential in N.) The following theorem shows that the ub-KF maintains the state error covariance which is lower-bounded by the state error covariance of the exact KF.

Theorem 10. *If the ub-KF starts with an initial covarnace $\bar{P}(0|0)$, such that $\bar{P}(0|0) \succeq P(0|0)$, then $\bar{P}(k|k) \succeq P(k|k)$ for all $k \geq 0$.*

Proof: See Appendix C.2. ■

7.4 Convergence

Using the approximate Kalman filtering method, we can derive convergence conditions for the state error covariance of the exact KF. Such condition is studied in [Sinopoli *et al.*, 2004], in which there is a lossy communication channel between the plant and the estimator, and the result is extend to the case with two communication links in [Liu and Goldsmith, 2004]. However, the extension of the result to the general case with N communication links is unknown. Although we do not provide a definite answer for the general case, it provides a condition under which the state error covariance can be unbounded for the general case with N communication links.

Theorem 11. *If $(\mathbb{E}[A(Z)]^T, \mathbb{E}[A(Z)]^T C^T)$ is not stabilizable, or equivalently, $(\mathbb{E}[A(Z)], C\mathbb{E}[A(Z)])$ is not detectable, then there exists an initial covariance $P(0|0)$ such that $P(k|k)$ diverges as $k \to \infty$.*

Proof: See Appendix C.3. ■

7.5 Simulation Results

In simulation, we study the performance of the modified Kalman filtering algorithm shown in Section 7.2.2 against the standard Kalman filter which assumes no communication errors. Then we provide motivating examples showing the effectiveness of the lb-KF and ub-KF.

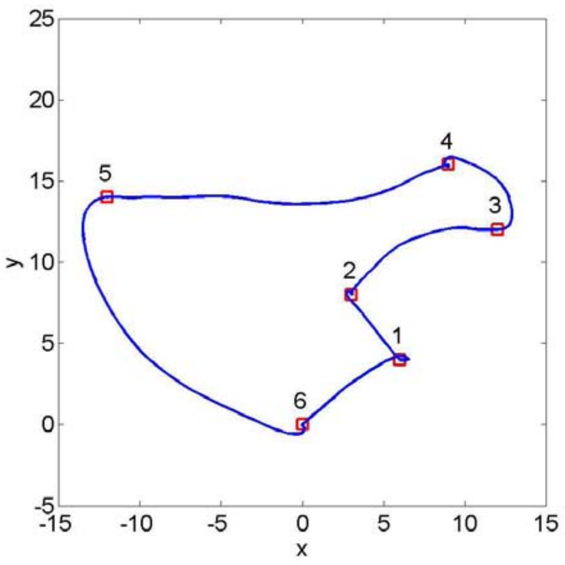


Figure 7.1: The mission of the multi-agent system is to visit sites of interests (shown in squares) sequentially from site 1 to site 6 (starting from site 6). The trajectory of the leader agent is shown in solid line. The control inputs to the leader are computed using the robust minimum-time control described in Section 6.3.3.

Our simulation is based on a scenario inspired by the model by Vicsek *et al.* [Vicsek *et al.*, 1995]. Consider a general DNCS system (7.7) consisting of $N = 5$ agents. The state vector of each agent is $x = [x, y, \dot{x}, \dot{y}]^T$, where (x, y) and $(\dot{x}, \dot{y})$ are the position and the velocity components of the vehicle along the x and y axes, respectively.

The agent 1 is a leader and its dynamics is modeled as

$$x_1(k+1) = A_{11}x_1(k) + B_1u_1(k) + G_1w_1(k), \tag{7.28}$$

where $u_1(k) \in \mathbb{R}^{n_u}$ is a control input to the leader agent and $B_1 \in \mathbb{R}^{n_x \times n_u}$.

The dynamics of an agent $i > 0$ is

$$x_i(k+1) = \sum_{j=i-1}^{i+1} A_{\kappa(j)i}(Z)x_{\kappa(j)}(k) + G_iw_i(k), \tag{7.29}$$

where $\kappa(j) = (j - 1 \mod N) + 1$. For $\kappa(j) = i$,

$$A_{ii}(Z) = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & \frac{1}{S(i)} & 0 \\ 0 & 0 & 0 & \frac{1}{S(i)} \end{bmatrix} \qquad G_i = \begin{bmatrix} \frac{\delta}{2} & 0 \\ 0 & \frac{\delta}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix},$$

where δ is the sampling interval. For $\kappa(j) \neq i$,

$$A_{\kappa(j)i}(Z) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\kappa(j)i}/S(i) & 0 \\ 0 & 0 & 0 & Z_{\kappa(j)i}/S(i) \end{bmatrix}$$

with $S(i) = \sum_{j=i-1}^{i+1} Z_{\kappa(j)i}$. Hence, when the agent i communicates with its neighboring agents $\kappa(i-1)$ and $\kappa(i+1)$, its new velocity is the average of its velocity and velocities received from its neighboring agents. In addition, $\delta = 1$ and $Q_i = \text{diag}(0.01^2, 0.01^2)$

The mission of this multi-agent system is to visit sites of interests in minimum time with a bounded control input. The mission scenario is shown in Figure 7.1 along with the trajectory of the leader agent. The control inputs to the leader are computed using the robust minimum-time control described in Section 6.3.3. The trajectories of all agents at different times are shown in Figure 7.3.

We first study the performance gap between the modified KF against the standard KF which does not assume communication losses. Let the communication matrix be

$$P_{\text{com}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \lambda & 1 & \lambda & 0 & 0 \\ 0 & \lambda & 1 & \lambda & 0 \\ 0 & 0 & \lambda & 1 & \lambda \\ \lambda & 0 & 0 & \lambda & 1 \end{bmatrix}. \tag{7.30}$$

The measurement model (7.8) is used where C is a 10×20 matrix such that $y(k)$ consists of noisy position measurements of all agents and $R = \text{diag}(0.1^2, \ldots, 0.1^2)$. We varied λ from 0.1 to 1.0 with a 0.1 increment. For each value of λ, 100 test cases are generated. For each test case, we ran the modified KF and the standard KF and computed the mean square error (MSE) of state estimates. The result is shown in Figure 7.2. The figure shows a clear benefit of the modified KF when the communication loss uncertainty is higher. In addition, the modified KF shows an excellent performance for all values of λ.

We now consider two cases: *Case A* and *Case B*. *Case A* is the model described above with $\lambda = 0.7$. *Case B* is the same as *Case A* except C is a 6×20 matrix such that $y(k)$ consists of noisy position measurements of agent 1, 3, and 4. The positions of agents 2 and 5 are not observed in *Case B*. The results are shown in Table 7.1. The modified KF performs well compared to the standard KF but it requires more computation time. The approximate KFs perform better than the standard KF without much overhead in run-time. Since a less number of states are observed in *Case B*, the state uncertainty is higher in *Case B* and the ub-KF performs better than the lb-KF for *Case B*.

7.6 Stabilizing Communication Control

In this section, we consider the problem of finding a communication control which stabilizes the general DNCS (7.7) for given $\{A(z) : z \in \mathcal{Z}\}$, *i.e.*, finding a communication

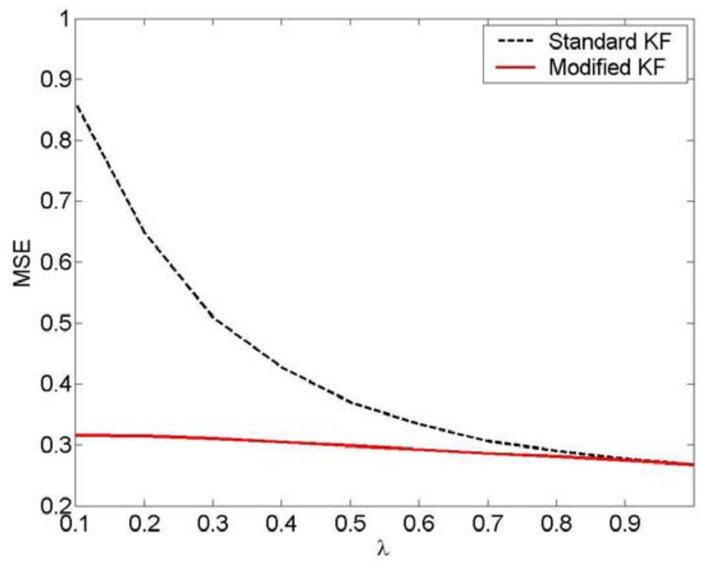


Figure 7.2: The average MSE as a function of λ in (7.30). For each value of λ, 100 test cases are used to compute the average MSE. As the value of λ decreases, the performance gap between the modified KF and the standard KF increases.

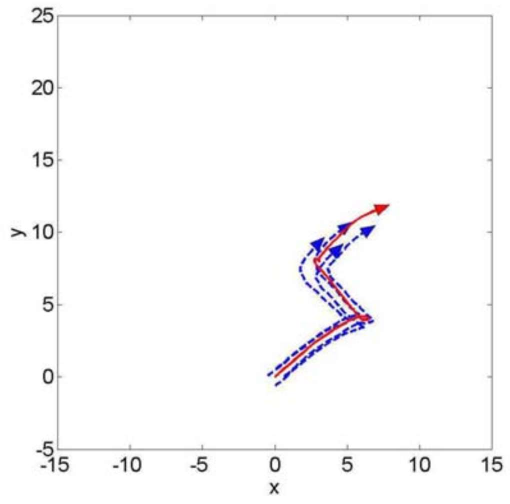
matrix P_{com} such that the general DNCS (7.7) is stable. For example, in wireless communication, one can control the transmission power to increase or decrease entries of P_{com}. We use the stability results for jump linear systems by Costa and Fragoso [Costa and Fragoso, 1993]. We use the notation $A \succ 0$ if A is a positive definite matrix and $A \succeq 0$ if A is a positive semidefinite matrix. The spectral radius of A is denoted by $\rho(A)$.

Definition 3. *The DNCS model (7.7) is mean square stable (MSS) if, for any initial condition x_0 and second-order independent wide sense stationary random process $\{w(k)\}$, there exist x^* and P^* independent of x_0 such that:*

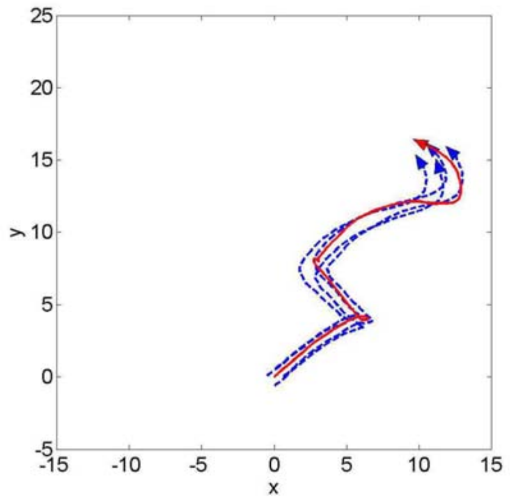
(a) $\| \mathbb{E}[x(k)] - x^* \| \to 0$ *as* $k \to \infty$

(b) $\| \mathbb{E}[x(k)x(k)^T] - P^* \| \to 0$ *as* $k \to \infty$.

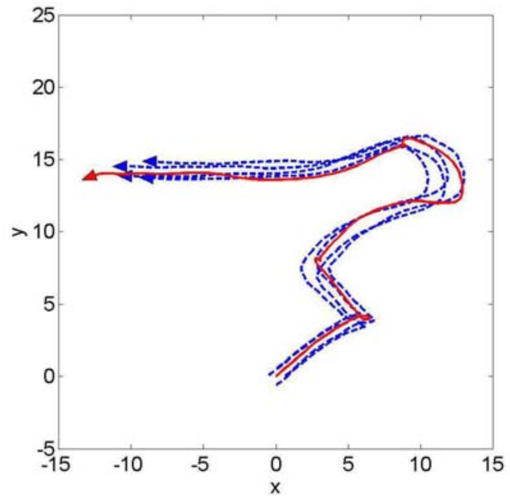
Theorem 12 (Corollary 1 of [Costa and Fragoso, 1993])**.** *The DNCS model (7.7) is MSS if*



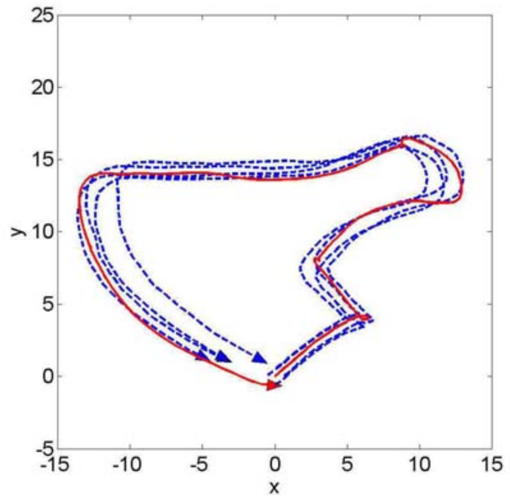
(a) $k = 100$



(b) $k = 140$



(c) $k = 220$



(d) $k = 270$

Figure 7.3: Trajectories of all agents at different times (leader in a solid line, followers in dashed lines).

and only if there exists $G \succ 0$ *such that*

$$G - \sum_{z=\mathcal{Z}} p_z A(z)^T G A(z) \succ 0. \tag{7.31}$$

Table 7.1: Comparison of different Kalman filters: standard KF, modified KF (Mod-KF), lower-bound KF, and upper-bound KF

		KF	mod-KF	lb-KF	ub-KF
Case A	MSE	0.303	0.283	0.292	0.352
	Run-time	0.56s	11.69s	0.64s	0.81s
Case B	MSE	0.696	0.542	0.748	0.512
	Run-time	0.52s	11.69s	0.60s	0.87s

Theorem 13. *The DNCS model (7.7) is MSS if*

$$\sum_{z\in\mathcal{Z}} p_z \rho(A(z)^T A(z)) < 1. \tag{7.32}$$

Proof: Fix $\alpha > 0$ and let $G = \alpha I_n$ where $n = Nn_x$ and I_n is a $n \times n$ identity matrix. Clearly, $G \succ 0$.

$$\begin{aligned} G - \sum_{z=\mathcal{Z}} p_z A(z)^T G A(z) &= \alpha I_n - \alpha \sum_{z=\mathcal{Z}} p_z A(z)^T A(z) \\ &\succeq \alpha \left(1 - \sum_{z=\mathcal{Z}} p_z \rho(A(z)^T A(z))\right) I_n \\ &\succ 0, \end{aligned}$$

since $\sum_{z=\mathcal{Z}} p_z \rho(A(z)^T A(z)) < 1$. Hence, by Theorem 12, (7.7) is MSS. ■

Using Theorem 13, one can easily check if there exists a stabilizing communication control. Let $\rho(z) = \rho(A(z)^T A(z))$ and consider the following linear programming (LP) problem.

$$\begin{aligned} \text{minimize} \quad & c = \textstyle\sum_{z\in\mathcal{Z}} p_z \rho(z) \\ \text{subject to} \quad & \textstyle\sum_{z\in\mathcal{Z}} p_z = 1 \\ & 0 \le p_z \le 1, \qquad z \in \mathcal{Z}. \end{aligned} \tag{7.33}$$

Notice that we can add restrictions on p_z to reflect physical constraints in the DNCS. If there exists a feasible solution with $c < 1$, we know for sure that there exists a stabilizing communication control based on Theorem 13. However, it is important to note that Theorem 13 is only a sufficient condition. Hence, when the LP (7.33) does not have a feasible solution with $c < 1$, we can not say there is no communication control which stabilizes the DNCS model (7.7).

Example 2. *Consider a 2-agent DNCS system where*

$$A_{11} = \begin{bmatrix} -.138 & .414 \\ .598 & .219 \end{bmatrix} \quad A_{12} = \begin{bmatrix} -.075 & -.006 \\ -.505 & -.34 \end{bmatrix}$$
$$A_{21} = \begin{bmatrix} -.35 & -.245 \\ -.495 & .049 \end{bmatrix} \quad A_{22} = \begin{bmatrix} .185 & -.137 \\ -.298 & -.653 \end{bmatrix}.$$

Note that $Z_1 = Z_{11} = 1$, $Z_2 = Z_{12}$, $Z_3 = Z_{21}$, and $Z_4 = Z_{22} = 1$. Let $z_1 = [1, 0, 0, 1]^T$, $z_2 = [1, 0, 1, 1]^T$, $z_3 = [1, 1, 0, 1]^T$, and $z_4 = [1, 1, 1, 1]^T$. Then $\mathcal{Z} = \{z_1, z_2, z_3, z_4\}$ and $\rho(z_1) = 0.517$, $\rho(z_2) = 1.038$, $\rho(z_3) = 1.044$, and $\rho(z_4) = 0.925$. We also have constraints on p_z: $0 \leq p_{z_1} \leq 0.6$, $0.1 \leq p_{z_2} \leq 0.5$, $0.1 \leq p_{z_3} \leq 0.5$, and $0.1 \leq p_{z_4} \leq 0.3$. Since not all ρ are less than 1, it is not clear that the DNCS system is MSS with the constraints on p_z. By solving the LP (7.33) for this problem, one finds that there is a feasible solution: $p^* = [p_{z_1}, p_{z_2}, p_{z_3}, p_{z_4}]^T = [0.6, 0.1, 0.1, 0.2]^T$ with $c = 0.704$. Hence, the DNCS system is MSS with p^*.

Now consider the same example as before except $1.5A_{22}$ is used instead of A_{22}. Then $\rho(z_1) = 1.164$, $\rho(z_2) = 1.56$, $\rho(z_3) = 1.558$, and $\rho(z_4) = 1.531$ and there is no feasible solution to the LP (7.33) with $c < 1$ and the system is not MSS. The state evolutions of these two systems are shown in Figure 7.4. □

The LP (7.33) is an efficient way to check the existence of a stable communication control. But it does not provide the solution in the form we want. We want to find the communication matrix P_{com}, not $\{p_z\}$. For the notational convenience, we again use the index $n \in \{1, \ldots, M = N^2\}$ described in Section 7.1, where ij is indexed by $n = N(i -$

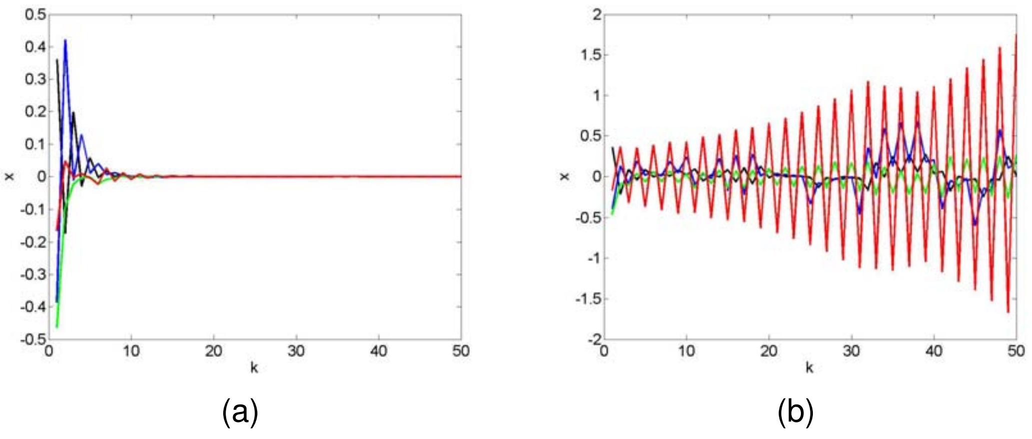


Figure 7.4: (a) The state evolution of the MSS system given in Example 2. (b) The state evolution of the non-MSS system given in Example 2.

$1) + j$. Then

$$p_z = \prod_{i,j} p_{ij}^{z_{N(i-1)+j}} (1 - p_{ij})^{1 - z_{N(i-1)+j}} = \prod_{n}^{M} p_n^{z_n} (1 - p_n)^{1 - z_n}. \tag{7.34}$$

The problem we want to solve is:

$$\begin{aligned} &\text{find} && \{p_n\} \\ &\text{subject to} && \textstyle\sum_{z \in \mathcal{Z}} \rho(z) \prod_n^M p_n^{z_n} (1 - p_n)^{1 - z_n} < 1 \\ & && 0 \le p_n \le 1, \qquad \forall n \in \{1, \ldots, M\}. \end{aligned} \tag{7.35}$$

The problem (7.35) is a special case of signomial programming which is non-convex and only a locally optimal solution can be computed efficiently [Boyd *et al.*, 2005]. Instead of solving the problem (7.35) directly, we relax the problem and use geometric programming (GP), which is a convex optimization problem [Boyd and Vandenberghe, 2004; Boyd *et al.*, 2005].

The relaxed GP is

$$\begin{array}{ll} \max & \frac{1}{\epsilon}\prod_{m=1}^{2M} q_n \\ \text{subject to} & \sum_{z\in\mathcal{Z}} \rho(z)\prod_{n}^{M} q_{2(n-1)+1}^{z_n} q_{2n}^{1-z_n} + \epsilon \leq 1 \\ & q_{2(n-1)+1} + q_{2n} \leq 1, \qquad \forall n \in \{1,\ldots,M\} \\ & 0 \leq q_m \leq 1, \qquad \forall m \in \{1,\ldots,2M\} \end{array} \tag{7.36}$$

As in the LP (7.33), we can add restrictions on q_n to reflect physical constraints in the DNCS. Using the GP (7.36), we can find ranges of P_{com} by running (7.36) multiple times with different upper bounds on q_m.

Example 3. *Consider the DNCS system described in Example 2 but replace A_{22} by $1.35A_{22}$. When the upper bounds $p_{12} \leq 0.4$ and $p_{21} \leq 0.4$ are used, the GP finds $q = [0.4, 0.6, 0.4, 0.6]^T$ and $\sum_{z\in\mathcal{Z}} p_z\rho(z) = 0.987$. Since $q_1 + q_2 = q_3 + q_4 = 1$, this is a feasible communication control. However, when the upper bounds $p_{12} \leq 1$ and $p_{21} \leq 1$ are used, the GP finds $q = [0.426, 0.571, 0.436, 0.562]^T$. But $q_1 + q_2 \neq 1$ and $q_3 + q_4 \neq 1$, hence, this is not a feasible communication control.* □

7.7 Summary

In this chapter, we have described a distributed networked control system (DNCS) consisting of multiple agents communicating over a lossy communication channel, *e.g.* wireless channel. A DNCS is an extension of an NCS to model a distributed multi-agent system such as the Vicsek model. Optimal linear filtering algorithms based on the Kalman filter are developed to estimate the states of DNCSs. Due to the interaction among agents in the DNCS dynamic model, the dynamics of each agent can show high nonlinearity. But the filtering algorithm was able to estimate the states correctly using the knowledge of the communication matrix.

Then, we described efficient approximate filtering algorithms for estimating states of a distributed networked control system (DNCS). While the time complexity of the exact estimation method can be exponential in the number of possible communication link configurations, the time complexity of an approximate method is not dependent on the number of possible configurations. Using the approximate filtering method, we have also derived a condition under which the stable state estimation is not possible for the general case with an arbitrary number of lossy communication links. We still need a better approximation method for the ub-KF. A better approximation method may enable us to find conditions under which the stable state estimation is guaranteed for the general case.

In the last part of this chapter, the stabilizing communication control problem of a DNCS is described, where one finds the acceptable ranges of packet loss rates at which the overall system is stable. We developed algorithms based on convex optimization to check the existence of stabilizing communication control and for solving the stabilizing communication control problem.

We have assumed that the communication matrix is independent from other parameters. In our future work, we will relax this assumption by making the matrix a function of the states of the agents. This is a valid model for wireless communication, since the transmission power decreases with an increasing distance between a transmitter and a receiver. We are currently extending the stabilizing communication control problem using the necessary condition for stability.

CHAPTER 8

CONCLUSIONS

Distributed networked sensing and control systems such as sensor networks, networked control systems, distributed control systems, multi-agent systems, and heterogeneous sensor networks, is based on a collection of resource-constrained agents or nodes. The limited supply of power and other constraints, such as manufacturing costs and limited package sizes, limit the capabilities of each agent. These constraints create measurement inconsistency, such as noise and false alarms, and communication unreliability, such as transmission failures and delays. Hence, they are the major challenges in realizing a real-time situation understanding and control system based on a distributed networked sensing and control system.

In this thesis, two mathematical frameworks are utilized to address the challenges with distributed networked sensing and control systems: multi-target tracking and pursuit evasion games. Multi-target tracking is a general framework which can be used to describe many estimation and inference problems appearing in distributed networked sensing and control systems. The pursuit evasion game can be viewed as the worst-case control problem since the pursuers must find the best controls to pursue evaders while the equally capable evaders try to avoid the pursuers with their best efforts. Hence, if one can solve the pursuit evasion games, she can also solve the other control problems.

Based on a general Bayesian framework for multi-target tracking problems, we have developed an efficient approximation method, called Markov chain Monte Carlo data association (MCMCDA), to efficiently solve the data association problems appearing in multi-target tracking problems. MCMCDA is applied to manage identity of moving objects using the identity-mass-flow framework and used in the development of a real-time hierarchical control system, *LochNess*. *LochNess* decouples the estimation of evader states from the control of pursuers via multiple layers of data fusion. We have shown a successful demonstration of *LochNess* using a large-scale outdoor sensor network. We believe the design methodology used in *LochNess* can be extended to more sophisticated data acquisition and control systems.

We have also presented a general framework for modeling a distributed networked control system consisting of multiple agents communicating over a lossy communication channel. We have described exact and approximate filtering methods to estimate states of a distributed networked control system. We have also described the problem of finding a communication control which stabilizes a distributed networked control system.

Future Work

The sensor network technology is no longer in the domain of researchers. There is a growing number of companies developing the sensor network technology and providing services. The IEEE 802.15.4 standard for low-power digital radios and the protocol standardization by the ZigBee Alliance have been the driving force toward the commercialization of sensor networks. We will soon see a number of products based on the sensor network technology in our daily lives.

However, the conventional sensor network research has focused largely on the low-bandwidth sensors, such as temperature, acoustic, infrared, etc. As a result, we have limited applications of sensor networks. This limitation can be addressed by integrating high-bandwidth sensors such as image sensors. The high-bandwidth sensors can provide in-depth situation awareness and recognition. But the bandwidth limitation in sensor networks complicates this integration. The use of high-bandwidth sensors requires efficient in-network processing techniques to reduce communication load. The methods developed in this thesis are currently applied to develop algorithms for heterogeneous sensor networks with low-bandwidth and high-bandwidth sensors. In particular, the identity management technique developed in Chapter 5 is shown to be an efficient mechanism to integrate high-bandwidth and low-bandwidth sensors due to its compact representation of identities. We are also extending the Bayesian framework to a number of estimation problems in heterogeneous sensor networks.

The security is a major issue which must be addressed before the sensor network technology can be applied to a wide range of applications. It is the author's view that the security must be considered at every level of a system. It is essential to improve security in hardware and software of sensor networks. But it is also required to develop secure applications. In a large-scale distributed system, there is a close relationship between the security of the system and the robustness of the system. If a system is robust, it will be able to tolerate regional security bleaches. Hence, in a sense, the robust algorithms developed in this thesis is already secure to some level of security bleaches. We will investigate the relationship between the level of security and the robustness of a distributed system and develop algorithms to improve security of the overall sensor network system.

In the most part of this thesis, we have assumed that sensor nodes are stationary. The mobility of sensor networks can make the system more robust and accurate in decision making. Mobile sensors can be used as active sensors in a highly cluttered environment, *e.g.*, detecting a suspicious activity in a crowd. In some situations, stationary sensors can not collect enough information to make a reliable decision or inference due to occlusion. The mobile sensors will be used in these cases to collect additional information required to make a reliable decision or inference. They can be also used to expand the coverage of a sensor network by active sensing and improve connectivity of a sensor network by sharing communication load of bottleneck nodes. However, the task of controlling a swarm of mobile sensors is difficult and challenging. Chapter 7 described a simple framework for distributed networked control systems and discussed the connection between communication and stability of the system. We will extend this result by finding the connection between communication and the robustness of in-network estimation. We will also extend our results in Chapter 6 and develop a distributed networked system of mobile and stationary sensors with real-time estimation and control capabilities, hence, closing the loop around the heterogeneous sensor networks.

Appendix A

Proofs of Theorems in Chapter 2

A.1 Proof of Theorem 1

We first need the following lemma to prove Theorem 1 and Theorem 2.

Lemma 1. *Let* $Y_1, \ldots, Y_t$ *be independent random variables such that* $P(Y_t = 1) = p_t$ *and* $P(Y_t = 0) = 1 - p_t$*, for arbitrary* $0 < p_t < 1$*. Let* Z_t *be a Bernoulli process with parameter* p *and, for all* t*,* $p \leq p_t$*. Let* $S_Y^t = \sum_{k=1}^{t} Y_k$ *and* $S_Z^t = \sum_{k=1}^{t} Z_k$*. Then, for all* $t \in \mathbb{N}$ *and* $d \leq t$*,*

$$P(S_Y^t < d) \leq P(S_Z^t < d). \tag{A.1}$$

Proof: We prove the claim by induction. For $t = 1$, $(1 - p_1) \leq (1 - p)$. So $P(Y_1 < 1) \leq P(Z_1 < 1)$. Now suppose that (A.1) is true for t. Then, for $t + 1$ and any

$d \leq t+1$, we have, using the induction hypothesis,

$$\begin{aligned}
P(S_Y^{t+1} < d) &= P(S_Y^t + Y_{t+1} < d) \\
&= P(S_Y^t + Y_{t+1} < d | Y_{t+1} = 0) P(Y_{t+1} = 0) \\
&+ P(S_Y^t + Y_{t+1} < d | Y_{t+1} = 1) P(Y_{t+1} = 1) \\
&= P(S_Y^t < d) P(Y_{t+1} = 0) + P(S_Y^t < d-1) P(Y_{t+1} = 1) \\
&\leq P(S_Z^t < d) P(Y_{t+1} = 0) + P(S_Z^t < d-1) P(Y_{t+1} = 1) \\
&= P(S_Z^t < d) - P(S_Z^t = d-1) P(Y_{t+1} = 1) \\
&\leq P(S_Z^t < d) - P(S_Z^t = d-1) P(Z_{t+1} = 1) \\
&= P(S_Z^t < d) P(Z_{t+1} = 0) + P(S_Z^t < d-1) P(Z_{t+1} = 1) \\
&= P(S_Z^{t+1} < d).
\end{aligned}$$

■

We now prove Theorem 1. Let Z_t be a Bernoulli process with parameter η and let $S_Z^t = \sum_{k=1}^t Z_k$. Let $X_t \in \{1, \ldots, N\}$ be a Markov chain on G with the initial state distribution and transition probabilities given in Section 2.1. Let Y_t be an observation made at time t. $Y_t = 1$ with probability η_{X_t} and $Y_t = 0$ with probability $(1 - \eta_{X_t})$. Let $S_Y^t = \sum_{k=1}^t Y_k$. Notice that, for all t, $\eta_{X_t} \geq \eta$. Given a track $x_{1:t}$, the observations are independent. By Lemma 1, for any t, $P(S_Y^t < d | x_{1:t}) \leq P(S_Z^t < d)$ for $d \leq t$. Furthermore, the inequality holds for any $x_{1:t}$. Hence,

$$\begin{aligned}
P(S_Y^t < d) &= \sum_{x_{1:t}} P(S_Y^t < d | x_{1:t}) P(x_{1:t}) \\
&\leq \left(\sum_{x_{1:t}} P(x_{1:t}) \right) P(S_Z^t < d) \\
&= P(S_Z^t < d).
\end{aligned}$$

Let $\delta_t = \sqrt{\frac{2}{\eta t}\log\left(\frac{T}{s\epsilon}\right)}$. $\delta_t < 1$ for all $t \geq t_0$ since $s > \frac{T}{\epsilon}\exp\left(-\frac{\eta t_0}{2}\right)$. Thus,

$$\begin{aligned}
P(W_T) &\leq \sum_{t=1}^{T}\sum_{k=1}^{\infty}\mathbb{I}(t = ks)P(S_Y^t < \theta_t) \\
&\leq \sum_{t=1}^{T}\sum_{k=1}^{\infty}\mathbb{I}(t = ks)P(S_Z^t < \theta_t) \\
&= \sum_{k=1}^{\lfloor \frac{T}{s} \rfloor} P(S_Z^{ks} < \theta_{ks}) \\
&\leq \sum_{k=1}^{\lfloor \frac{T}{s} \rfloor} P(S_Z^{ks} < (1-\delta_{ks})\eta ks) \\
&< \sum_{k=1}^{\lfloor \frac{T}{s} \rfloor} \exp\left(-\frac{\eta ks\delta_{ks}^2}{2}\right) \\
&= \sum_{k=1}^{\lfloor \frac{T}{s} \rfloor} \exp\left(-\frac{\eta ks}{2}\frac{2}{\eta ks}\log\left(\frac{T}{s\epsilon}\right)\right) \\
&= \sum_{k=1}^{\lfloor \frac{T}{s} \rfloor} \epsilon \cdot \frac{s}{T} \leq \epsilon \cdot \frac{s}{T} \cdot \frac{T}{s} = \epsilon
\end{aligned}$$

where $\mathbb{I}$ is an indicator function and the Chernoff bound [Motwani and Raghavan, 1995] is used in the fourth inequality.

A.2 Proof of Theorem 2

Let Z_t be a Bernoulli process with parameter η and let $S_Z^t = \sum_{k=1}^{t} Z_k$. Let $S_Y^t = \sum_{k=1}^{t} Y_k^{X_k}$. Then, as in the proof of Theorem 1, $P(S_Y^t < d) \leq P(S_Z^t < d)$. Let $\delta = \sqrt{\frac{2}{\eta s}\log\left(\frac{1+\epsilon}{\epsilon}\right)}$. Then $\delta < 1$ by our choice of s. Thus,

$$
\begin{aligned}
P(W_\infty) &\le \sum_{t=1}^{\infty}\sum_{k=1}^{\infty} \mathbb{I}(t = ks) P(S_Y^t < \theta_t) \\
&\le \sum_{t=1}^{\infty}\sum_{k=1}^{\infty} \mathbb{I}(t = ks) P(S_Z^t < \theta_t) \\
&= \sum_{k=1}^{\infty} P(S_Z^{ks} < \theta_{ks}) \\
&\le \sum_{k=1}^{\infty} P(S_Z^{ks} < (1-\delta)\eta ks) \\
&< \sum_{k=1}^{\infty} \exp\left(-\frac{\eta ks\delta^2}{2}\right) \\
&= \sum_{k=1}^{\infty} \exp\left(-\frac{\eta ks}{2}\frac{2}{\eta s}\log\left(\frac{1+\epsilon}{\epsilon}\right)\right) \\
&= \sum_{k=1}^{\infty} \left(\frac{\epsilon}{1+\epsilon}\right)^k = \epsilon,
\end{aligned}
$$

where the Chernoff bound [Motwani and Raghavan, 1995] is used in the fourth inequality.

A.3 Proof of Theorem 3

To avoid confusion, let us denote the joint distribution without false detections (2.1) by $Q(X_{1:T}, Y_{1:T})$ and the joint distribution with false detections (2.9) by $P(X_{1:T}, Y_{1:T})$. By the optimality, for any feasible track $x_{1:T}$, *i.e.*, $P(x_{1:T}) > 0$,

$$
\begin{aligned}
P(x_{1:T}^*, y_{1:T}) &\ge P(x_{1:T}, y_{1:T}) \\
Q(q_{1:T}, y_{1:T}) &\ge Q(x_{1:T}, y_{1:T}).
\end{aligned}
$$

In particular, we have

$$\begin{aligned} P(x^*_{1:T}, y_{1:T}) &\geq P(q_{1:T}, y_{1:T}) && \text{(A.2)} \\ Q(q_{1:T}, y_{1:T}) &\geq Q(x^*_{1:T}, y_{1:T}). && \text{(A.3)} \end{aligned}$$

From (A.2), we obtain the first inequality in (2.11). Now consider the following joint distribution ratio

$$\begin{aligned} R &:= \frac{P(q_{1:T}, y_{1:T})}{P(x^*_{1:T}, y_{1:T})} \\ &= \frac{P(q_{1:T})}{P(x^*_{1:T})} \frac{\prod_{t=1}^T P(y_t|q_t)}{\prod_{t=1}^T P(y_t|x^*_t)} \\ &= \frac{P(q_{1:T}) \prod_{t=1}^T \left[\eta^{y_t^{q_t}} (1-\eta)^{1-y_t^{q_t}} \prod_{i\neq q_t} \xi^{y_t^i} (1-\xi)^{1-y_t^i}\right]}{P(x^*_{1:T}) \prod_{t=1}^T \left[\eta^{y_t^{x_t^*}} (1-\eta)^{1-y_t^{x_t^*}} \prod_{i\neq x_t^*} \xi^{y_t^i} (1-\xi)^{1-y_t^i}\right]} \\ &= \frac{P(q_{1:T})}{P(x^*_{1:T})} \frac{\prod_{t=1}^T \eta^{y_t^{q_t}} (1-\eta)^{1-y_t^{q_t}}}{\prod_{t=1}^T \eta^{y_t^{x_t^*}} (1-\eta)^{1-y_t^{x_t^*}}} \frac{\prod_{t=1}^T \xi^{y_t^{x_t^*}} (1-\xi)^{1-y_t^{x_t^*}}}{\prod_{t=1}^T \xi^{y_t^{q_t}} (1-\xi)^{1-y_t^{q_t}}}. \end{aligned}$$

Let $S_q = \sum_{t=1}^T y_t^{q_t}$ and $S_x = \sum_{t=1}^T y_t^{x_t^*}$. Then

$$\begin{aligned} R &= \frac{P(q_{1:T})}{P(x^*_{1:T})} \frac{\eta^{S_q}(1-\eta)^{T-S_q}}{\eta^{S_x}(1-\eta)^{T-S_x}} \frac{\xi^{S_x}(1-\xi)^{T-S_x}}{\xi^{S_q}(1-\xi)^{T-S_q}} \\ &= \frac{P(q_{1:T})}{P(x^*_{1:T})} \left(\frac{\eta}{1-\eta}\right)^{S_q-S_x} \left(\frac{\xi}{1-\xi}\right)^{S_x-S_q} && \text{(A.4)} \\ &= \frac{Q(q_{1:T}, y_{1:T})}{Q(x^*_{1:T}, y_{1:T})} \left(\frac{\xi}{1-\xi}\right)^{S_x-S_q}. \end{aligned}$$

By (A.3), $\frac{Q(q_{1:T},y_{1:T})}{Q(x^*_{1:T},y_{1:T})} \geq 1$. But by (A.2), $\frac{P(q_{1:T},y_{1:T})}{P(x^*_{1:T},y_{1:T})} \leq 1$. Hence, $\left(\frac{\xi}{1-\xi}\right)^{S_x-S_q} \leq 1$. Since $\xi \leq .5$, $\left(\frac{\xi}{1-\xi}\right) \leq 1$ and we must have $S_x \geq S_q$. But S_q cannot be arbitrarily small. Since $\frac{Q(q_{1:T},y_{1:T})}{Q(x^*_{1:T},y_{1:T})} \geq 1$,

$$\begin{aligned}
\frac{P(q_{1:T})}{P(x^*_{1:T})}\left(\frac{\eta}{1-\eta}\right)^{S_q-S_x} &\geq 1 \\
S_q - S_x &\geq \frac{\log\left(\frac{P(x^*_{1:T})}{P(q_{1:T})}\right)}{\log\left(\frac{\eta}{1-\eta}\right)} \\
S_x &\leq S_q + \frac{\log\left(\frac{P(q_{1:T})}{P(x^*_{1:T})}\right)}{\log\left(\frac{\eta}{1-\eta}\right)}.
\end{aligned}$$

Let $b = \frac{\log(P(q_{1:T})/P(x^*_{1:T}))}{\log(\eta/(1-\eta))}$. Then, we can bound S_x as

$$S_q \leq S_x \leq S_q + b.$$

Notice that if $P(q_{1:T}) = P(x^*_{1:T})$, $S_q \geq S_x$ and we get an equality $S_q = S_x$. Hence, $\alpha = 1$ and $q_{1:T} = x^*_{1:T}$. On the other hand, we cannot have $P(q_{1:T}) < P(x^*_{1:T})$, since this requires $S_q > S_x$, contradicting the optimality of $x^*_{1:T}$. Now R takes the minimum value if $S_x = S_q + b$. So

$$\begin{aligned}
R &\geq \frac{P(q_{1:T})}{P(x^*_{1:T})}\left(\frac{\eta}{1-\eta}\right)^{-b}\left(\frac{\xi}{1-\xi}\right)^{b} \\
&= \left(\frac{\xi}{1-\xi}\right)^{b} \\
&\geq \left(\frac{\xi}{1-\xi}\right)^{\frac{\log(P(q_{1:T})/m)}{\log(\eta/(1-\eta))}} \\
&= \frac{1}{\alpha}.
\end{aligned}$$

Hence $P(q_{1:T}, y_{1:T})/P(x^*_{1:T}, y_{1:T}) \geq \frac{1}{\alpha}$ and we get the second inequality in (2.11).

A.4 Proof of Corollary 2

Since $\eta \geq \frac{c}{1+c}$, $\frac{\eta}{1-\eta} \geq c$. So

$$\begin{aligned}
\alpha &= \left(\frac{1-\xi}{\xi}\right)^{\frac{\log(P(q_{1:T})/m)}{\log(\eta/(1-\eta))}} \\
&\leq \left(\frac{1-\xi}{\xi}\right)^{\frac{\log r}{\log(\eta/(1-\eta))}} \\
&\leq \left(\frac{1-\xi}{\xi}\right)^{\frac{\log r}{\log c}} \\
&= \left(\frac{1-\xi}{\xi}\right)^{\frac{\log(1+\epsilon_1)}{\log\left(\frac{1-\xi}{\xi}\right)}} \\
&= 1+\epsilon_1.
\end{aligned}$$

Appendix B

PROOFS OF THEOREMS IN CHAPTER 4

B.1 Proof of Theorem 5

To prove Theorem 5, we need the following lemmas.

Lemma 2. *Let* $C = \frac{p_d \bar{L}}{\lambda_f (1-p_d)}$ *and* $D = \frac{\lambda_f (1-p_d)}{\underline{L} p_d}$. *For any* $\omega_0, \omega_1, \omega_2 \in \Omega$*, if* $\omega_1 = \omega_0 - e_0$*, for some edge* $e_0 \in \omega_0$*, and* $\omega_2 = \omega_1 - e_1$*, for some edge* $e_1 \in \omega_1$*, then:*

$$\begin{aligned} \pi(\omega_0)/\pi(\omega_1) &\leq C \\ \pi(\omega_0)/\pi(\omega_2) &\leq C^2 \end{aligned} \quad \textit{and} \quad \begin{aligned} \pi(\omega_1)/\pi(\omega_0) &\leq D \\ \pi(\omega_2)/\pi(\omega_0) &\leq D^2. \end{aligned}$$

Proof: ω_0 and ω_1 are identical except that ω_1 is missing the edge e_0. So $|\omega_0| = |\omega_1| + 1$. If $e_0 = (u, v)$ and $k = |\omega_0|$,

$$\begin{aligned} \pi(\omega_0)/\pi(\omega_1) &= \frac{\lambda_{\mathrm{f}}^{N-k} p_{\mathrm{d}}^{k} (1-p_{\mathrm{d}})^{K-k}}{\lambda_{\mathrm{f}}^{N-(k-1)} p_{\mathrm{d}}^{k-1} (1-p_{\mathrm{d}})^{K-(k-1)}} \hat{P}^v(u|y_{1:t-1}) \\ &= \frac{p_{\mathrm{d}}}{\lambda_{\mathrm{f}}(1-p_{\mathrm{d}})} \hat{P}^v(u|y_{1:t-1}) \\ &\leq C. \end{aligned}$$

On the other hand,

$$\begin{aligned}
\pi(\omega_1)/\pi(\omega_0) &= \frac{\lambda_{\mathrm{f}}^{N-(k-1)} p_{\mathrm{d}}^{k-1}(1-p_{\mathrm{d}})^{K-(k-1)}}{\lambda_{\mathrm{f}}^{N-k} p_{\mathrm{d}}^{k}(1-p_{\mathrm{d}})^{K-k}} \frac{1}{\hat{P}^v(u|y_{1:t-1})} \\
&= \frac{\lambda_{\mathrm{f}}(1-p_{\mathrm{d}})}{p_{\mathrm{d}}} \frac{1}{\hat{P}^v(u|y_{1:t-1})} \\
&\leq D.
\end{aligned}$$

Since $\pi(\omega_0)/\pi(\omega_2) = \pi(\omega_0)/\pi(\omega_1) \times \pi(\omega_1)/\pi(\omega_2)$, by repeating the above argument twice, we get $\pi(\omega_0)/\pi(\omega_2) \leq C^2$. Similarly, we have $\pi(\omega_2)/\pi(\omega_0) \leq D^2$. ■

Lemma 3. *Let $R = \max\{1, C, D\}$, where C and D are defined in Lemma 2. Then the maximum edge loading of the Markov chain $\mathcal{M}$ is bounded as $\bar{\rho} \leq 4R^4K^2N$.*

Proof: For each pair of matchings X, Y in G, we define the canonical path γ_{XY} as in [Jerrum and Sinclair, 1996]. Consider the symmetric difference $X \oplus Y$, where $X \oplus Y = (X - Y) \cup (Y - X)$. $X \oplus Y$ is a disjoint collection of paths in G including closed cycles, each of which has edges that belong to X and Y alternately. Suppose that we have fixed some arbitrary ordering on all simple paths in G, and designate a "start vertex" to each of the paths, which is arbitrary if the path is a closed cycle but must be an endpoint otherwise. This gives a unique ordering $P_1, P_2, \ldots, P_m$ on the paths appearing in $X \oplus Y$. The canonical path from X to Y involves "unwinding" each of the P_i in turn as follows. We need to consider two cases:

(i) P_i is not a cycle. Let P_i consist of the sequence $(v_0, v_1, \ldots, v_l)$ of vertices with the start vertex v_0. If $(v_0, v_1) \in Y$, perform a sequence of switching moves replacing (v_{2j+1}, v_{2j+2}) by (v_{2j}, v_{2j+1}) for $j = 0, 1, \ldots$, and finish with an addition move if l is odd. If $(v_0, v_1) \in X$, remove (v_0, v_1) and proceed as before for the reduced path $(v_1, \ldots, v_l)$.

(ii) P_i is a cycle. Let P_i consist of the sequence $(v_0, v_1, \ldots, v_{2l+1})$ of vertices, for $l \geq 1$, where v_0 is the start vertex, and $(v_{2j}, v_{2j+1}) \in X$ for $j = 0, \ldots, l$, with remaining edges belonging to Y. We first remove the edge (v_0, v_1). Now we are left with an open path O

with endpoints v_0, v_1, with the start vertex v_k of O, for $k \in \{0, 1\}$. Then we unwind O as in (i) above but treating v_{1-k} as the start vertex to identify that it was a cycle.

Let t be an arbitrary edge in the Markov chain $\mathcal{M}$, *i.e.*, a transition from ω to $\omega' \neq \omega$. Let $cp(t) = \{(X, Y) : \gamma_{XY} \ni t\}$ be the set of canonical paths that use t. We define a function $\eta_t : cp(t) \to \Omega$ as in [Jerrum and Sinclair, 1996],

$$\eta_t(X, Y) = \begin{cases} X \oplus Y \oplus (\omega \cup \omega') - e_{XY_t}, & \\ \qquad \text{if } t \text{ is a switch move and the current path is a cycle;} & \\ X \oplus Y \oplus (\omega \cup \omega'), \qquad \text{otherwise,} & \end{cases}$$

where e_{XY_t} is the edge in X adjacent to the start vertex that was removed first in (ii) above. $\eta_t(X, Y)$ is always a matching in G and η_t is injective as shown in [Jerrum and Sinclair, 1996]. Notice that the bipartite graph G considered here is a subset of the graphs considered in [Jerrum and Sinclair, 1996] so the arguments about η_t can be directly applied here.

Notice that

$$Q(t) \quad = \quad Q(\omega, \omega') = \pi(\omega) P(\omega, \omega') = \frac{1}{2|E|} \min\{\pi(\omega), \pi(\omega')\}. \tag{B.1}$$

Next, we bound $\pi(X)\pi(Y)$ and we need to consider four cases:

(i) t is a deletion move. We have $\omega' = \omega - e$ and $\eta_t(X, Y) = X \oplus Y \oplus (\omega \cup \omega')$. Since $\omega \cup \eta_t(X, Y)$ and $X \cup Y$ are identical when viewed as multisets,

$$\begin{aligned} \pi(X)\pi(Y) &= \pi(\omega)\pi(\eta_t(X, Y)) \\ &= \frac{2|E|Q(t)}{\min\{\pi(\omega), \pi(\omega')\}} \pi(\omega)\pi(\eta_t(X, Y)) \\ &= 2|E|Q(t) \max\left\{1, \frac{\pi(\omega)}{\pi(\omega')}\right\} \pi(\eta_t(X, Y)) \\ &\leq 2R|E|Q(t)\pi(\eta_t(X, Y)), \end{aligned}$$

where we used the identity (B.1) in the second equality and Lemma 2 for the last inequality.

(ii) t is an addition move. We have $\omega' = \omega + e$ and $\eta_t(X,Y) = X \oplus Y \oplus (\omega \cup \omega')$. Since $\omega \cup \eta_t(X,Y)$ and $X \cup Y$ are identical when viewed as multisets, using the arguments from (i),

$$\pi(X)\pi(Y) \;\leq\; 2R|E|Q(t)\pi(\eta_t(X,Y)).$$

(iii) t is a switch move and the current path is a cycle. Suppose $\omega' = \omega + e - e'$. Let $\omega_1 = \omega + e$. Then $\omega' = \omega_1 - e'$. Since $\frac{\pi(\omega)}{\pi(\omega')} = \frac{\pi(\omega_1)}{\pi(\omega')}\frac{\pi(\omega)}{\pi(\omega_1)}$, by Lemma 2, $\frac{\pi(\omega)}{\pi(\omega')} \leq CD \leq R^2$. Since $\eta_t(X,Y) = X \oplus Y \oplus (\omega \cup \omega') - e_{XY_t}$, the multisets $\omega \cup \eta_t(X,Y)$ differs from $X \cup Y$ only in that e and e_{XY_t} are missing from it. Hence, by Lemma 2,

$$\begin{aligned}\pi(X)\pi(Y) &\leq C^2\pi(\omega)\pi(\eta_t(X,Y))\\ &= 2C^2|E|Q(t)\max\left\{1, \frac{\pi(\omega)}{\pi(\omega')}\right\}\pi(\eta_t(X,Y))\\ &\leq 2R^4|E|Q(t)\pi(\eta_t(X,Y)).\end{aligned}$$

(iv) t is a switch move and the current path is not a cycle. This case is similar to (iii) but the multisets $\omega \cup \eta_t(X,Y)$ differs from $X \cup Y$ only in that e is missing from it. Hence, by Lemma 2,

$$\begin{aligned}\pi(X)\pi(Y) &\leq C\pi(\omega)\pi(\eta_t(X,Y))\\ &= 2C|E|Q(t)\max\left\{1, \frac{\pi(\omega)}{\pi(\omega')}\right\}\pi(\eta_t(X,Y))\\ &\leq 2R^3|E|Q(t)\pi(\eta_t(X,Y)).\end{aligned}$$

In summary, we have, in all cases, $\pi(X)\pi(Y) \leq 2R^4|E|Q(t)\pi(\eta_t(X,Y))$. Thus, for any transition t,

$$\begin{aligned}
\frac{1}{Q(t)} \sum_{\gamma_{XY} \ni t} \pi(X)\pi(Y)|\gamma_{XY}| &\leq 2R^4|E| \sum_{\gamma_{XY} \ni t} \pi(\eta_t(X,Y))|\gamma_{XY}| \\
&\leq 4R^4K|E| \sum_{\gamma_{XY} \ni t} \pi(\eta_t(X,Y)) \\
&\leq 4R^4K|E| \\
&\leq 4R^4K^2N
\end{aligned}$$

where the second inequality follows from the fact that the length of any canonical path is bounded by $2K$, the third equality is due to the fact that η_t is injective and π is a probability distribution, and the last inequality follows from $|E| \leq KN$. Hence, $\bar{\rho} \leq 4R^4K^2N$. ■

We now prove Theorem 5. $\mathcal{M}$ is a finite, reversible, ergodic Markov chain with loop probabilities $P(\omega,\omega) \geq \frac{1}{2}$ for all states ω (see Section 4.2.2). Hence, by Theorem 4, we have

$$\tau_\omega(\epsilon) \leq \bar{\rho}(\log \pi(\omega)^{-1} + \log \epsilon^{-1}). \tag{B.2}$$

The upper bound for $\bar{\rho}$ is computed from Lemma 3. Now we just need to find the upper bound for $\pi(\omega)^{-1}$. The normalizing constant in (4.12) is

$$Z = \sum_{\omega \in \Omega} \left(\lambda_{\mathrm{f}}^{N-|\omega|} p_{\mathrm{d}}^{|\omega|} (1-p_{\mathrm{d}})^{K-|\omega|} \prod_{(u,v)\in\omega} \hat{P}^v(u|y_{1:t-1}) \right). \tag{B.3}$$

Hence,

$$\begin{aligned} Z &\leq \sum_{\omega \in \Omega} m_1^K m_3(K,N) \\ &= m_1^K m_3(K,N)|\Omega| \\ &\leq m_1^K m_3(K,N) \sum_{k=0}^{K} \binom{K}{k} \frac{N!}{(N-k)!} \\ &\leq m_1^K m_3(K,N)(K+1)!N!, \end{aligned}$$

where the second inequality is by (4.13). Although this bound on Z is not tight, it will serve our purpose. For any $\omega \in \Omega$, $\pi(\omega) \geq \frac{1}{Z} m_2^K m_4(K,N)$ so

$$\begin{aligned} \frac{1}{\pi(\omega)} &\leq \frac{Z}{m_2^K m_4(K,N)} \\ &\leq \left(\frac{m_1}{m_2}\right)^K \frac{m_3(K,N)}{m_4(K,N)}(K+1)!N!. \end{aligned}$$

Hence,

$$\begin{aligned} \log \frac{1}{\pi(\omega)} &\leq \log \left(\left(\frac{m_1}{m_2}\right)^K \frac{m_3(K,N)}{m_4(K,N)}(K+1)!N! \right) \\ &= m_0(K,N). \end{aligned}$$

Putting all together, we have, for all initial state $\omega \in \Omega$, $\tau_\omega(\epsilon) \leq 4R^4K^2N(m_0(K,N) + \log \epsilon^{-1})$.

B.2 Proof of Theorem 6

To prove Theorem 6, we apply the techniques developed to prove Lemma 3 in [Jerrum and Sinclair, 1993] to our problem. Let $\beta_{\epsilon_2} = \{(j,k) : \beta_{jk} \geq \epsilon_2\}$. For now, assume $(j,k) \in \beta_{\epsilon_2}$, *i.e.*, $\beta_{jk} \geq \epsilon_2$. Let $X_{jk}(\omega) = \mathbb{I}((\hat{y}^k, y^j) \in \omega)$ where $\mathbb{I}$ is an indicator function. Notice that

$\mathbb{E}_\pi(X_{jk}) = \pi(\omega_{jk}) = \beta_{jk}$, where $\omega_{jk} = \{\omega \in \Omega : (y^j, k) \in \omega\}$. Since $\|p - \pi\|_{\text{tv}} \le \epsilon$ and $\epsilon \le \epsilon_1\epsilon_2/8$,

$$|p(\omega_{jk}) - \pi(\omega_{jk})| \le \epsilon \le \frac{\epsilon_1 \pi(\omega_{jk})}{8} \tag{B.4}$$

Let $\bar\beta_{jk} = \frac{1}{s}\sum_{i=1}^s X_{jk}(\omega_i)$ be the sample mean of s samples from p. Then $\mathbb{E}(\bar\beta_{jk}) = p(\omega_{jk})$ and $\mathbb{V}\text{ar}(\bar\beta_{jk}) = \frac{1}{s}\mathbb{V}\text{ar}_p(X_{jk})$. By Chebyshev's inequality,

$$P\left(\left|\bar\beta_{jk} - p(\omega_{jk})\right| > \frac{\epsilon_1}{3}p(\omega_{jk})\right) \le \frac{9}{\epsilon_1^2 s}\frac{\mathbb{V}\text{ar}_p(X_{jk})}{p(\omega_{jk})^2}. \tag{B.5}$$

Now if $|\bar\beta_{jk} - p(\omega_{jk})| \le \frac{\epsilon_1}{3}p(\omega_{jk})$, from (B.4),

$$\begin{aligned}
|\bar\beta_{jk} - \pi(\omega_{jk})| &\le |\bar\beta_{jk} - p(\omega_{jk})| + |p(\omega_{jk}) - \pi(\omega_{jk})| \\
&\le \frac{\epsilon_1}{3}p(\omega_{jk}) + \frac{\epsilon_1}{8}\pi(\omega_{jk}) \\
&\le \frac{\epsilon_1}{3}\left(1 + \frac{\epsilon_1}{8}\right)\pi(\omega_{jk}) + \frac{\epsilon_1}{8}\pi(\omega_{jk}) \\
&\le \frac{\epsilon_1}{2}\pi(\omega_{jk}) \qquad (B.6)
\end{aligned}$$

and $\bar\beta_{jk}$ estimates $\pi(\omega_{jk})$ within ratio $1 + \epsilon_1$.

Now let us bound the difference between $\mathbb{V}\text{ar}_\pi(X_{jk})$ and $\mathbb{V}\text{ar}_p(X_{jk})$. Notice that $\mathbb{V}\text{ar}_p(X_{jk}) = \sum_{\omega\in\Omega} p(\omega)X_{jk}(\omega)^2 - (\mathbb{E}_p X_{jk})^2$ and $\mathbb{V}\text{ar}_\pi(X_{jk}) = \sum_{\omega\in\Omega}\pi(\omega)X_{jk}(\omega)^2 - (\mathbb{E}_\pi X_{jk})^2$.

$$\begin{aligned}
|\mathbb{V}\text{ar}_p(X_{jk}) - \mathbb{V}\text{ar}_\pi(X_{jk})| &= \left|\sum_{\omega\in\Omega} X_{jk}(\omega)^2\left(p(\omega) - \pi(\omega)\right) + (\mathbb{E}_\pi X_{jk})^2 - (\mathbb{E}_p X_{jk})^2\right| \\
&\le \left|\sum_{\omega\in\Omega} X_{jk}(\omega)^2\left(p(\omega) - \pi(\omega)\right)\right| + \left|(\mathbb{E}_\pi X_{jk})^2 - (\mathbb{E}_p X_{jk})^2\right|. \qquad (B.7)
\end{aligned}$$

Since $X_{jk}(\omega) \le 1$,

$$\left|\sum_{\omega\in\Omega} X_{jk}(\omega)^2\left(p(\omega) - \pi(\omega)\right)\right| \le \left|\sum_{\omega\in\Omega} p(\omega) - \pi(\omega)\right| \le \epsilon. \tag{B.8}$$

On the other hand,

$$\begin{aligned}
\left|(\mathbb{E}_\pi X_{jk})^2 - (\mathbb{E}_p X_{jk})^2\right| &= \left|(\mathbb{E}_\pi X_{jk} + \mathbb{E}_p X_{jk})(\mathbb{E}_\pi X_{jk} - \mathbb{E}_p X_{jk})\right| \\
&\leq 2\left|\mathbb{E}_\pi X_{jk} - \mathbb{E}_p X_{jk}\right| \\
&\leq 2\left|\sum_{\omega\in\Omega} \pi(\omega) - p(\omega)\right| \qquad \text{(B.9)} \\
&\leq 2\epsilon. \qquad \text{(B.10)}
\end{aligned}$$

Using (B.8) and (B.9) in (B.7), we have

$$|\mathbb{V}\text{ar}_p(X_{jk}) - \mathbb{V}\text{ar}_\pi(X_{jk})| \leq 3\epsilon \leq \frac{3\epsilon_1 \pi(\omega_{jk})}{8}, \qquad \text{(B.11)}$$

where the last inequality is due to the condition $\epsilon \leq \epsilon_1\epsilon_2/8$ and $\pi(\omega_{jk}) \geq \epsilon_2$.

Since $\epsilon_1 < 1$ and $\mathbb{V}\text{ar}_\pi(X_{jk}) \leq \pi(\omega_{jk})$, we find the following bound using (B.4) end (B.11):

$$\frac{\mathbb{V}\text{ar}_p(X_{jk})}{p(\omega_{jk})^2} \leq \frac{\mathbb{V}\text{ar}_\pi(X_{jk}) + \frac{3}{8}\pi(\omega_{jk})}{\left(\frac{7}{8}\pi(\omega_{jk})\right)^2} \leq \frac{2}{\pi(\omega_{jk})}. \qquad \text{(B.12)}$$

Hence, by choosing $s = 72\epsilon_1^{-2}\epsilon_2^{-1}$ and using (B.5) and (B.12),

$$P\left(|\bar{\beta}_{jk} - p(\omega_{jk})| > \frac{\epsilon_1}{3} p(\omega_{jk})\right) \leq \frac{1}{4}, \qquad \text{(B.13)}$$

that is, $\bar{\beta}_{jk}$ estimates $\pi(\omega_{jk})$ within ratio $1 + \epsilon_1$ with probability at least $3/4$.

To make the analysis easier, we assume a sampling technique which is slightly different from Algorithm 5. We consider repeating the above experiment by an odd number t times, independently. Let $\hat{\beta}_{jk}$ be the median of the resulting t values of $\bar{\beta}_{jk}$. From above, the probability that $\hat{\beta}_{jk}$ fails to approximate β_{jk} within ratio $1+\epsilon_1$ is at most

$$\begin{aligned}\sum_{i=(t+1)/2}^{t}\binom{t}{i}\left(\frac{1}{4}\right)^{i}\left(\frac{3}{4}\right)^{t-i} &\leq \left(\frac{1}{4}\right)^{t/2}\left(\frac{3}{4}\right)^{t/2}\sum_{i=(t+1)/2}^{t}\binom{t}{i} \\ &\leq \left(\frac{3}{16}\right)^{t/2} 2^{t} \\ &= \left(\frac{3}{4}\right)^{t/2}.\end{aligned}$$

Now let $t = 7\lceil \log \eta^{-1} \rceil$, then

$$\left(\frac{3}{4}\right)^{t/2} \leq \left(\frac{3}{4}\right)^{3.5\lceil \log \eta^{-1}\rceil} \leq \eta^{3.5\log(4/3)} \leq \eta.$$

Hence, with a total of $st = 504\epsilon_1^{-2}\epsilon_2^{-1}\lceil \log \eta^{-1} \rceil$ samples, $\hat{\beta}_{jk}$ estimates $\pi(\omega_{jk})$ within ratio $1+\epsilon_1$ with probability at least $1-\eta$ for $\beta_{jk} \geq \epsilon_2$.

Now consider β_{jk} that are smaller than ϵ_2. With probability at least $1-\eta$, for $(j,k) \in \beta_{\epsilon_2}$, $(1-\epsilon_1)\beta_{jk} \leq \hat{\beta}_{jk} \leq (1+\epsilon_1)\beta_{jk}$. So if $\hat{\beta}_{jk} \geq (1+\epsilon_1)\epsilon_2$, we must have $(j,k) \in \beta_{\epsilon_2}$. Hence, $\hat{\beta}_{jk} \leq (1+\epsilon_1)\epsilon_2$ or $|\hat{\beta}_{jk} - \beta_{jk}| \leq (1+\epsilon_1)\epsilon_2$ for $\beta_{jk} < \epsilon_2$.

B.3 Proof of Theorem 7

Lemma 4. *Suppose that* $0 < p_z, p_d < 1$ *and* $\lambda_b, \lambda_f > 0$. *If* $\zeta(d) > 0$, *for all* $d \in \{1, \ldots, \bar{d}\}$, *then the Markov chain* $\mathcal{M}$ *is irreducible.*

Proof: The birth and death moves are sufficient to illustrate the irreducibility of the chain. Since $0 < p_{\mathrm{z}}, p_{\mathrm{d}} < 1$ and $\lambda_{\mathrm{b}}, \lambda_{\mathrm{f}} > 0$, $P(\omega|Y) > 0$ for all $\omega \in \Omega$. Take an arbitrary partition $\omega \in \Omega$, say $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$. Now consider the partition $\omega' \in \Omega$, such that $\omega' = \{\tau_0'\}$, *i.e.*, ω' assigns all observations as false alarms. Since ω is arbitrary, the chain is irreducible if the chain can move from ω' to ω and from ω to ω'.

For the move from ω' to ω, consider K consecutive birth moves: $\omega_0 = \omega', \omega_1 = \{\{\tau_0' \setminus \tau_1\}, \tau_1\}, \ldots, \omega_K = \{\{\tau_0' \setminus \{\cup_{i=1}^K \tau_i\}\}, \tau_1, \ldots, \tau_K\} = \omega$. Since $\omega \in \Omega$, all tracks τ_k are legal, *i.e.*, τ_k satisfies the constraints described in Section 3.2 and, for $i = 1, \ldots, |\tau_k| - 1$, $\tau_k(t_{i+1}) \in L_d(\tau_k(t_i))$ for $1 \leq d = t_{i+1} - t_i \leq \bar{d}$. Thus, $\omega_k \in \Omega$ for all k. Because $\zeta(d) > 0$ and all tracks τ_k are legal, the probability of proposing τ_k at ω_{k-1} by the birth move is positive and $q(\omega_k, \omega_{k+1}) > 0$. For the move from ω to ω', consider K consecutive death moves: $\omega_K = \omega, \omega_{K-1}, \ldots, \omega_0 = \omega'$. The probability of removing the track τ_k at ω_k by the death move is positive and $q(\omega_{k+1}, \omega_k) > 0$. Since $P(\omega_k|Y) > 0$ for all k, the chain can move from ω' to ω and from ω to ω'. Hence, the chain is irreducible. ■

From Lemma 4, $\mathcal{M}$ is irreducible. $\mathcal{M}$ is aperiodic since there is always a positive probability of staying at the current state in the track update move. Now the transitions described in Algorithm 7 satisfy the detailed balance condition since it uses the Metropolis-Hastings kernel (4.1). Hence, by the ergodic theorem [Roberts, 1996], the chain converges to its stationary distribution $\pi(\omega)$ almost surely and $\hat{X} \to \mathbb{E}_\pi X$ as $n_{\text{mc}} \to \infty$.

Appendix C

PROOFS OF THEOREMS IN CHAPTER 7

C.1 Proof of Theorem 9

Lemma 5. *If $\underline{P}(k|k) \preceq P(k|k)$, then $\underline{P}(k+1|k) \preceq P(k+1|k)$.*

Proof: Using (7.23), we have

$$\begin{aligned} P(k+1|k) - \underline{P}(k+1|k) &= \mathbb{E}[A(Z)P(k|k)A(Z)^T] + \mathbb{E}[A(Z)\hat{x}(k|k)\hat{x}(k|k)^T A(Z)^T] \\ &- \hat{A}\hat{x}(k|k)\hat{x}(k|k)^T \hat{A}^T - \hat{A}\underline{P}(k|k)\hat{A}^T \\ &= P_1 + P_2, \end{aligned} \tag{C.1}$$

where $P_1 = \mathbb{E}[A(Z)P(k|k)A(Z)^T] - \hat{A}\underline{P}(k|k)\hat{A}^T$ and $P_2 = \mathbb{E}[A(Z)\hat{x}(k|k)\hat{x}(k|k)^T A(Z)] - \hat{A}\hat{x}(k|k)\hat{x}(k|k)^T\hat{A}^T$.

If $P_1 \succeq 0$ and $P_2 \succeq 0$, then $P(k+1|k) - \underline{P}(k+1|k) \succeq 0$ and it completes the proof.

$$\begin{aligned} P_1 &= \mathbb{E}[A(Z)P(k|k)A(Z)^T] - \hat{A}\underline{P}(k|k)\hat{A}^T - \hat{A}P(k|k)\hat{A}^T + \hat{A}P(k|k)\hat{A}^T \\ &= \mathbb{E}[A(Z)P(k|k)A(Z)^T] - \hat{A}P(k|k)\hat{A}^T + \hat{A}(P(k|k) - \underline{P}(k|k))\hat{A}^T. \end{aligned} \tag{C.2}$$

Since $P(k|k)$ is a symmetric matrix, $P(k|k)$ can be decomposed into $P(k|k) = U_1 D_1 U_1^T$, where U_1 is a unitary matrix and D_1 is a diagonal matrix. Hence,

$$\begin{aligned} P_1 &= \mathbb{E}[(A(Z)U_1 D_1^{1/2})(A(Z)U_1 D_1^{1/2})^T] - \mathbb{E}[A(Z)U_1 D_1^{1/2}]\mathbb{E}[A(Z)U_1 D_1^{1/2}]^T \\ &+ \hat{A}(P(k|k) - \underline{P}(k|k))\hat{A}^T \\ &= \text{Cov}[A(Z)U_1 D_1^{1/2}] + \hat{A}(P(k|k) - \underline{P}(k|k))\hat{A}^T, \end{aligned} \tag{C.3}$$

where $\text{Cov}[H]$ denotes the covariance matrix of H. Since a covariance matrix is positive definite and $P(k|k) - \underline{P}(k|k) \succeq 0$ by assumption, $P_1 \succeq 0$. P_2 is a covariance matrix since $\hat{x}(k|k)\hat{x}(k|k)^T$ is symmetric, hence $P_2 \succeq 0$. ■

Lemma 6. *If $\underline{P}(k+1|k) \preceq P(k+1|k)$, then $\underline{P}(k+1|k+1) \preceq P(k+1|k+1)$.*

Proof: Applying the matrix inversion lemma to (7.20), we have

$$P(k+1|k+1) = \left(P(k+1|k)^{-1} + C^T R^{-1} C\right)^{-1}. \tag{C.4}$$

Let $P = P(k+1|k)$ and $\underline{P} = \underline{P}(k+1|k)$. Then

$$\begin{aligned} P &\succeq \underline{P} \\ P^{-1} &\preceq \underline{P}^{-1} \\ P^{-1} + C^T R^{-1} C &\preceq \underline{P}^{-1} + C^T R^{-1} C \\ \left(P^{-1} + C^T R^{-1} C\right)^{-1} &\succeq \left(\underline{P}^{-1} + C^T R^{-1} C\right)^{-1} \\ P(k+1|k+1) &\succeq \underline{P}(k+1|k+1). \end{aligned}$$

■

Now assume that the lb-KF starts with an initial covariance $\underline{P}(0|0)$, such that $\underline{P}(0|0) \preceq P(0|0)$. Then, using Lemma 5, Lemma 6, and the induction hypothesis, we see that

$\underline{P}(k|k) \preceq P(k|k)$ for all $k \geq 0$.

C.2 Proof of Theorem 10

Lemma 7. *If* $\bar{P}(k|k) \succeq P(k|k)$*, then* $\bar{P}(k+1|k) \succeq P(k+1|k)$.

Proof: Let $M = \hat{x}(k|k)\hat{x}(k|k)^T$ and I be an identity matrix. Then using (7.23),

$$\begin{aligned} \bar{P}(k|k) - P(k|k) &= \lambda_{\max}\mathbb{E}[A(Z)A(Z)^T] - \mathbb{E}[A(Z)P(k|k)A(Z)^T] - \mathbb{E}[A(Z)MA(Z)^T] \\ &= \mathbb{E}[A(Z)(\lambda_{\max}(\bar{P}(k|k))I - P(k|k))A(Z)^T] \\ &+ \mathbb{E}[A(Z)(\lambda_{\max}(M)I - M)A(Z)^T]. \end{aligned} \tag{C.5}$$

Since $\bar{P}(k|k) \succeq P(k|k)$ and $\lambda_{\max}(S)I - S \succeq 0$ for any symmetric matrix S, we have $\bar{P}(k|k) - P(k|k) \succeq 0$. ■

Using Lemma 7, Lemma 6, and the induction hypothesis, we obtain the theorem.

C.3 Proof of Theorem 11

Let us consider the lb-KF. Let $\underline{P}_k = \underline{P}(k|k)$, $\psi = GQG^T$, $\hat{A} = \mathbb{E}[A]$, and

$$F = -(C\hat{A}\underline{P}_k\hat{A}^TC^T + C\psi C^T + R)^{-1}(C\psi + C\hat{A}\underline{P}_k\hat{A}^T). \tag{C.6}$$

Then, based on the Riccati difference equation [Mosca, 1995], we can express $\underline{P}_{k+1}$ as

$$\begin{aligned} \underline{P}_{k+1} &= \hat{A}\underline{P}_k\hat{A}^T + \psi - F^T\left(C\hat{A}\underline{P}_k\hat{A}^TC^T + C\psi C^T + R\right)F \\ &= (\hat{A}^T + \hat{A}^TC^TF)^T\underline{P}_k(\hat{A}^T + \hat{A}^TC^TF) \\ &+ F^T(C\psi C^T + R)F + \psi C^TF + F^TC\psi + \psi. \end{aligned} \tag{C.7}$$

Hence, if $(\hat{A}^T + \hat{A}^T C^T F)$ is not a stability matrix, for some $\underline{P}_0 \preceq P(0|0)$, $\underline{P}_k$ diverges as $k \to \infty$. Since the state error covariance of the lb-KF diverges and $\underline{P}(k|k) \preceq P(k|k)$ for all $k \geq 0$ (Theorem 9), $P(k|k)$ diverges as $k \to \infty$.

BIBLIOGRAPHY

[Akyidliz *et al.*, 2002] I.F. Akyidliz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–116, August 2002.

[Arora *et al.*, 2004] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, Dec. 2004.

[Aslam *et al.*, 2003] J. Aslam, Z. Butler, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *ACM International Conference on Embedded Networked Sensor Systems*, 2003.

[Balakrishnan *et al.*, 2004] H. Balakrishnan, I. Hwang, and C. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *Proc. of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.

[Bar-Shalom and Fortmann, 1988] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Academic Press, San Diego, CA, 1988.

[Basar and Olsder, 1995] T. Basar and G.J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, London and San Diego, second edition, 1995.

[Beichl and Sullivan, 2000] I. Beichl and F. Sullivan. The Metropolis algorithm. *Computing in Science and Engineering*, 2(1):65–69, 2000.

[Belta *et al.*, 2005] C. Belta, V. Isler, and G.J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, October 2005.

[Bergman and Doucet, 2000] N. Bergman and A. Doucet. Markov chain Monte Carlo data association for target tracking. In *Proc. of the IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, June 2000.

[Blom and Bar-Shalom, 1988] H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Trans. Automatic Control*, 33:780–783, 1988.

[Blondel *et al.*, 2005] V.D. Blondel, J.M. Hendrickx, A. Olshevsky, and J.N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proc. of the 44th IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Boyd *et al.*, 2005] S. Boyd, S.J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. Technical report, Stanford University EE Technical Report, 2005.

[Brooks *et al.*, 2004] Richard R. Brooks, David Friedlander, John Koch, and Shashi Phoha. Tracking multiple targets with self-organizing distributed ground sensors. *J. Parallel Distrib. Comput.*, 64:874–884, 2004.

[Burkard and Çela, 1998] R.E. Burkard and R. Çela. Linear assignment problem and extensions. Technical Report 127, Karl-Franzens University of Graz, Graz, Austria, 1998.

[Chen *et al.*, 2003] W.P. Chen, J.C. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. In *Proc. of the 11th IEEE International Conference on Network Protocols*, November 2003.

[Chong *et al.*, 1990] C.Y. Chong, S. Mori, and K.C. Chang. Distributed multitarget multisensor tracking. In Y. Bar-Shalom, editor, *Multitarget-Multisensor Tracking: Advanced Applications*, pages 247–295. Artech House: Norwood, MA, 1990.

[Chu *et al.*, 2004] M. Chu, S.K. Mitter, and F. Zhao. Distributed multiple target tracking and data association in ad hoc sensor networks. In *Proc. of the 6th International Conference on Information Fusion*, July 2004.

[Coates, 2004] Mark Coates. Distributed particle filters for sensor networks. In *Proc. of the 3nd workshop on Information Processing in Sensor Networks*, April 2004.

[Collins and Uhlmann, 1992] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multivariate distributed states. *IEEE Trans. Aerospace and Electronic Systems*, 28(3):909–916, July 1992.

[Cong *et al.*, 2004] S. Cong, L. Hong, and D. Wicker. Markov-chain Monte-Carlo approach for association probability evaluation. *IEE Proceedings of Control, Theory and Applications*, 151(2):185–193, March 2004.

[Costa and Fragoso, 1993] O.L.V. Costa and M.D. Fragoso. Stability results for discrete-time linear systems with Markovian jumping parameters. *Journal of Mathematical Analysis and Applications*, 179:154–178, 1993.

[Cover and Thomas, 1991] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

[Cox,] I.J. Cox. Multiple hypothesis tracking code. http://www.ee.ucl.ac.uk/ ˜icox/.

[Cox and Hingorani, 1996] I.J. Cox and S.L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[Cox, 1993] I.J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[Culler *et al.*, 2004] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor networks. *IEEE Computer, Special Issue in Sensor Networks*, Aug. 2004.

[Cybenko *et al.*, 2004] G. Cybenko, V.H. Berk, V. Crespi, R.S. Gray, and G. Jiang. An overview of process query systems. In *Proc. of SPIE Vol. 5403 Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, Orlando, FL, April 2004.

[de Freitas and Gordon, 2001] A. Doucet J.F.G. de Freitas and N.J. Gordon, editors. *Sequential Monte Carlo Methods In Practice*. Springer Verlag, New York, 2001.

[Dellaert *et al.*, 2003] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun. EM, MCMC, and chain flipping for structure from motion with unknown correspondence. *Machine Learning*, 50:45–71, 2003.

[Diaconis and Stroock, 1991] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *Annals of Applied Probability*, 1:36–61, 1991.

[Doolina and Sitara, 2005] David M. Doolina and Nicholas Sitara. Wireless sensors for wildfire monitoring. In *Proc. of the SPIE Symposium on Smart Structures and Materials: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems.*, volume 5765, pages 477–484, May 2005.

[Dutta *et al.*, 2005] Prabal Dutta, Mike Grimmer, Anish Arora, Steve Bibyk, and David Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[Dutta *et al.*, 2006] Prabal Dutta, Jonathan Hui, Jaein Jeong, Sukun Kim, Cory Sharp, Jay Taneja, Gilman Tolle, Kamin Whitehouse, and David Culler. Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments. In *Proc. of the International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors*, 2006.

[Eraker, 2001] Bjørn Eraker. MCMC analysis of diffusion models with application to finance. *J. Bus. Econom. Statist.*, 19(2):177–191, 2001.

[Estrin *et al.*, 2001] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Salt Lake City, UT, May 2001.

[Estrin *et al.*, 2002] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, January 2002.

[Fitzgerald, 1990] R.J. Fitzgerald. Development of practical PDA logic for multipltarget tracking by microprocessor. In Y. Bar-Shalom, editor, *Multitarget-Multisensor Tracking: Advanced Applications*. Artech House: Norwood, MA, 1990.

[Gao, 2004] Z. Gao. On discrete time optimal control: A closed-form solution. In *Proceeding of the 2004 American Control Conference (ACC)*, pages 52–58, Boston, Massachusetts, U.S.A., June 2004.

[Gay *et al.*, 2003] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: a holistic approach to networked embedded systems. In *Proc. of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, pages 1–11, June 2003.

[Geman and Geman, 1984] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[Gharavi and Kumar, 2003] Hamid Gharavi and Srikanta P. Kumar, editors. *Sensor networks and applications*, volume 91. Proceedings of the IEEE, Special Issue, August 2003.

[Gilks *et al.*, 1996] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics Series. Chapman and Hall, 1996.

[Grocholsky *et al.*, 2004] B. Grocholsky, V. Kumar, and H.F Durrant-Whyte. Anonymous cooperation in robotic sensor networks. In *Proc. of the AAAI-04 Workshop on Sensor Networks*, 2004.

[Gu *et al.*, 2005] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, T. He, A. Tirumala, Q. Cao, J. Stankovic, T. Abdelzaher, and B. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys*, November 2005.

[Guibas *et al.*, 1999] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 9(4/5):471–493, 1999.

[Hastings, 1970] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[Hespanha *et al.*, 1999] J.P. Hespanha, H.J. Kim, and S.S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *IEEE Int. Conf. on Decision and Control*, pages 2432–2437, 1999.

[Hill *et al.*, 2004] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy. The platforms enabling wireless sensor networks. *Communications of the ACM*, 47(6):41–46, 2004.

[Huang and Russell, 1997] Timothy Huang and Stuart J. Russell. Object identification in a Bayesian context. In *Proc. of the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, Aug. 1997.

[Hui and Culler, 2004] Jonathan Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[Hwang *et al.*, 2004a] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin. Multiple-target tracking and identity management algorithm in clutter, with application to aircraft tracking. In *Proceedings of the American Control Conference*, Boston, MA, June 2004.

[Hwang *et al.*, 2004b] I. Hwang, K. Roy, H. Balakrishnan, and C. Tomlin. A distributed multiple-target identity management algorithm in sensor networks. In *Proc. of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.

[Hwang *et al.*, 2006] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin. Multiple-target tracking and identity management with application to aircraft tracking. *AIAA Journal of Guidance, Control and Dynamics*, 2006. in review.

[Imer *et al.*, 2004] O.C. Imer, S. Yuksel, and T. Basar. Optimal control of dynamical systems over unreliable communication links. In *Proc. of the NOLCOS*, Stutgart, Germany, 2004.

[Jadbabaie *et al.*, 2003] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Automatic Control*, 48(6):988–1001, 2003.

[Jerrum and Sinclair, 1993] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087–1116, 1993.

[Jerrum and Sinclair, 1996] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In Dorit Hochbaum, editor, *Approximations for NP-hard Problems*. PWS Publishing, Boston, MA, 1996.

[Jiang *et al.*, 2005] Xiaofan Jiang, Joseph Polastre, and David Culler. Perpetual environmentally powered sensor networks. In *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[Jordan, 2004] M.I. Jordan. *An Introduction to Probabilistic Graphical Models*. in preparation, 2004.

[Joshi *et al.*, 2005] A.A. Joshi, P. Deignan, P.H. Meckl, G.B. King, and K. Jennings. Information theoretic fault detection. In *Proceedings of the American Control Conference*, Portland, OR, June 2005.

[Julier and Uhlmann, 2004] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[Kailath *et al.*, 1999] T. Kailath, A.H. Sayed, and B. Hassibi. *State Space Estimation*. Prentice-Hall, 1999.

[Kapur and Kesavan, 1992] J.N. Kapur and H.K. Kesavan. *Entropy Optimization Principles with Application*. Academic Press, Inc., 1992.

[Kintner-Meyer and Conant, 2005] M. Kintner-Meyer and R. Conant. Opportunities of wireless sensors and controls for building operation. *Energy Engineering Journal*, 102(5):27–48, 2005.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Kurien, 1990] Thomas Kurien. Issues in the design of practical multitarget tracking algorithms. In Y. Bar-Shalom, editor, *Multitarget-Multisensor Tracking: Advanced Applications*. Artech House, Norwood, MA, 1990.

[LaMarca *et al.*, 2002] Anthony LaMarca, Waylon Brunette, David Koizumi, Matthew Lease, Stefan B. Sigurdsson, Kevin Sikorski, Dieter Fox, and Gaetano Borriello. Making sensor networks practical with robots. In *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, pages 152–166, London, UK, 2002.

[Lee and Markus, 1967] E.B. Lee and L. Markus. *Foundations of optimal control theory*. Wiley, New York, 1967.

[Lepetic *et al.*, 2003] M. Lepetic, G. Klancar, I. Skrjanc, D. Matko, and B. Potocnic. Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, 45:199–210, 2003.

[Lerro and Bar-Shalom, 1993] D. Lerro and Y. Bar-Shalom. Interacting multiple model tracking with target amplitude feature. *IEEE Trans. Aerospace and Electronic Systems*, 29:494–509, 1993.

[Li *et al.*, 2002] D. Li, K. Wong, Yu Hen Hu, and A. Sayeed. Detection, classification and tracking of targets. *IEEE Signal Processing Magazine*, 17-29, March 2002.

[Liberzon, 2003] D. Liberzon. On stabilization of linear systems with limited information. *IEEE Trans. Automatic Control*, 48(2):304–307, 2003.

[Linial *et al.*, 2000] N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20:545–568, 2000.

[Liu and Goldsmith, 2004] Xiangheng Liu and Andrea Goldsmith. Kalman filtering with partial observation losses. In *Proc. of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Dec. 2004.

[Liu *et al.*, 2003a] J.J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao. Distributed group management for track initiation and maintenance in target localization applications. In *Proc. of the 2nd workshop on Information Processing in Sensor Networks*, April 2003.

[Liu *et al.*, 2003b] J.J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. *J. of Applied Signal Processing*, April 2003.

[Liu *et al.*, 2004] J.J. Liu, J. Liu, M. Chu, J.E. Reich, and F. Zhao. Distributed state representation for tracking problems in sensor networks. In *Proc. of the 3nd workshop on Information Processing in Sensor Networks*, April 2004.

[Lorincz *et al.*, 2004] Konrad Lorincz, David Malan, Thaddeus R.F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoff Mainland, Steve Moulton, and Matt Welsh. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, Oct-Dec 2004.

[McErlean and Narayanan, 2002] D. McErlean and S. Narayanan. Distributed detection and tracking in sensor networks. In *Proc. of the 36th Asilomar Conference on Signal, System and Computers*, November 2002.

[Metropolis *et al.*, 1953] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[Morefield, 1971] C. L. Morefield. Application of 0-1 integer programming to multitarget tracking problems. *IEEE Trans. Automatic Control*, 22(3):302–312, June 1971.

[Mosca, 1995] Edoardo Mosca, editor. *Optimal, Predictive, Adaptive Control*. Prentice-Hall, New Jersey, 1995.

[Motwani and Raghavan, 1995] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.

[Nekovee, 2005] M. Nekovee. Ad hoc sensor networks on the road: the promises and challenges of vehicular ad hoc networks. In *Workshop on Ubiquitous Computing and e-Research*, Edinburgh, UK, May 2005.

[Nilim and Ghaoui, 2004] A. Nilim and L. El Ghaoui. Algorithms for air traffic flow management under stochastic environments. In *Proc. of American Control Conference*, 2004.

[Nilsson and Bernhardsson, 1997] J. Nilsson and B. Bernhardsson. LQG control over a Markov communication network. In *Proc. of the 36th IEEE Conference on Decision and Control*, Dec. 1997.

[Oh and Sastry, 2005a] Songhwai Oh and Shankar Sastry. A polynomial-time approximation algorithm for joint probabilistic data association. In *Proc. of the American Control Conference*, Portland, OR, June 2005.

[Oh and Sastry, 2005b] Songhwai Oh and Shankar Sastry. Tracking on a graph. In *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, Los Angeles, CA, April 2005.

[Oh and Sastry, 2006] Songhwai Oh and Shankar Sastry. Distributed networked control system with lossy links: State estimation and stabilizing communication control. In *Proc. of the 45th IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006.

[Oh *et al.*, 2004] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *Proc. of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Dec. 2004.

[Oh *et al.*, 2005] Songhwai Oh, Luca Schenato, and Shankar Sastry. A hierarchical multiple-target tracking algorithm for sensor networks. In *Proc. of the International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[Oh *et al.*, 2006a] Songhwai Oh, Inseok Hwang, and Shankar Sastry. Distributed multi-target tracking and identity management. *Journal of Guidance, Control, and Dynamics*, 2006. accepted.

[Oh *et al.*, 2006b] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain Monte Carlo data association for multi-target tracking. *IEEE Trans. Automatic Control*, 2006. submitted.

[Oh *et al.*, 2007] Songhwai Oh, Luca Schenato, Phoebus Chen, and Shankar Sastry. Tracking and coordination of multiple agents using sensor networks: system design, algorithms and experiments. *Proceedings of the IEEE*, 2007. to appear.

[Oh, 2003] Songhwai Oh. Multiple target tracking for surveillance. Technical Report UCB/ERL MO3/54, Univ. of California, Berkeley, 2003.

[Olfati-Saber and Murray, 2004] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Automatic Control*, 49(9):1520–1533, 2004.

[Pakzad *et al.*, 2005] Shamim N. Pakzad, Sukun Kim, Gregory L. Fenves, Steven D. Glaser, David E. Culler, and James W. Demmel. Multi-purpose wireless accelerometers for civil infrastructure monitoring. In *Proc. of the 5th International Workshop on Structural Health Monitoring*, Stanford, CA, September 2005.

[Pasula *et al.*, 1999] Hanna Pasula, Stuart J. Russell, Michael Ostland, and Yaacov Ritov. Tracking many objects with many sensors. In *Proc. of the International Joint Conference on Artificial Intelligence*, Stockholm, 1999.

[Pasula, 2003] Hanna Pasula. *Identity Uncertainty*. Univ. of California, Berkeley, CA, Ph.D. Thesis, Computer Science Division, 2003.

[Polastre *et al.*, 2005] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. In *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[Poore, 1995] A.B. Poore. Multidimensional assignment and multitarget tracking. *Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 19:169–196, 1995.

[Rabiner, 1989] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[Reid, 1979] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automatic Control*, 24(6):843–854, December 1979.

[Roberts, 1996] G.O. Roberts. Markov chain concepts related to sampling algorithms. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*, Interdisciplinary Statistics Series. Chapman and Hall, 1996.

[Roecker and Phillis, 1993] J.A. Roecker and G.L. Phillis. Suboptimal joint probabilistic data association. *IEEE Trans. Aerospace and Electronic Systems*, AES-29, 2:510–517, April 1993.

[Roecker, 1994] J.A. Roecker. A class of near optimal JPDA algorithms. *IEEE Trans. Aerospace and Electronic Systems*, AES-30, 2:504–510, April 1994.

[Rothblum and Schneider, 1989] U.G. Rothblum and H. Schneider. Scaling of matrices which have prescribed row sums and column sums via optimization. *Linear Algebra and Its Applications*, 114/115:737–764, 1989.

[Roundy *et al.*, 2004] S. Roundy, D. Steingart, L. Frchette, P.K. Wright, and J. Rabaey. Power sources for wireless networks. In *Proc. 1st European Workshop on Wireless Sensor Networks (EWSN '04)*, pages 1–17, Berlin, Germany, January 2004.

[Ryan, 1982] E.P. Ryan. *Optimal relay and saturation control synthesys*. Peter Peregrinus Ltd., London, 1982.

[Saccon, 2005] A. Saccon. Minimum time maneuver for nonholonomic car with acceleration constraints: Preliminary results. In *13th Mediterranean Conference on Control and Automation (MED)*, Limassol, Cyprus, 2005.

[Schenato *et al.*, 2005] Luca Schenato, Songhwai Oh, and Shankar Sastry. Swarm coordination for pursuit evasion games using sensor networks. In *Proc. of the International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[Schulz *et al.*, 2001] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2001.

[Shannon, 1948] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:623–656, October 1948.

[Sharp *et al.*, 2005] Cory Sharp, Shawn Schaffert, Alec Woo, Naveen Sastry, Chris Karlof, Shankar Sastry, and David Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proc. of the 2nd European Workshop on Wireless Sensor Networks*, pages 93–107, January 2005.

[Shi *et al.*, 2005] Ling Shi, Michael Epstein, Abhishek Tiwari, and Richard M. Murray. Estimation with information loss: Asymptotic analysis and error bounds. In *Proc. of the 44th IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005.

[Shim *et al.*, 2003] D.H. Shim, H.J. Kim, and S.S. Sastry. Decentralized reflective model predictive control of multiple flying robots in dynamic environment. In *Proc. of IEEE Conf. on Decision and Control*, Las Vegas, 2003.

[Shin *et al.*, 2003] J. Shin, L. Guibas, and F. Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *Proc. of the 2nd workshop on Information Processing in Sensor Networks*, April 2003.

[Sinkhorn, 1967] R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *American Mathematical Monthly*, 74:402–405, 1967.

[Sinopoli *et al.*, 2003] B. Sinopoli, C. Sharp, S. Schaffert, L. Schenato, and S. Sastry. Distributed control applications within sensor networks. *IEEE Proceedings Special Issue on Distributed Sensor Networks*, November 2003.

[Sinopoli *et al.*, 2004] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S. Sastry. Kalman filtering with intermittent observations. *IEEE Trans. Automatic Control*, 49(9):1453–1464, 2004.

[Sinopoli *et al.*, 2005] Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, and Shankar Sastry. Optimal control with unreliable communication: the tcp case. In *Proc. of the American Control Conference*, Portland, OR, June 2005.

[Sittler, 1964] R.W. Sittler. An optimal data association problem on surveillance theory. *IEEE Trans. Military Electronics*, MIL-8:125–139, April 1964.

[Szewczyk *et al.*, 2004] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan M. Mainwaring, and Deborah Estrin. Habitat monitoring with sensor networks. *Communication of the ACM*, 47(6):34–40, 2004.

[Tabuada and Pappas, 2005] P. Tabuada and G.J. Pappas. Hierarchical trajectory refinement for a class of nonlinear systems. *Automatica*, 41(4):701–708, April 2005.

[Thrun *et al.*, 1998] S Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (joint issue)*, 31(5):1–25, 1998.

[TinyOS, 2006] TinyOS. http://www.tinyos.net/, 2006.

[Tolle, 2005] Gilman Tolle. A network management system for wireless sensor networks. Master's thesis, Univ. of California, Berkeley, 2005.

[TR, 2003] 10 emerging tecnology that will change the world. *Technology Review*, 106(1):33–49, February 2003.

[Valiant, 1979] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[van Rijsbergen, 1979] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[Velenis and Tsiotras, 2005] E. Velenis and P. Tsiotras. Optimal velocity profile generation for given acceleration limits: Receding horizon implementation. In *American Control Conference (ACC05)*, pages 2147–2152, Portland, OR, USA, June 2005.

[Vicsek *et al.*, 1995] T. Vicsek, A. Czirok, E. Ben Jacob, I. Cohen, and O. Schochet. Novel type of phase transitions in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229, 1995.

[Vidal *et al.*, 2002] R. Vidal, O. Shakernia, J. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, October 2002.

[Wang *et al.*, 2004] H. Wang, K. Yao, G. Pottie, and D. Estrin. Entropy-based sensor selection heuristic for target localization. In *Proc. of the 3nd workshop on Information Processing in Sensor Networks*, April 2004.

[Warnake *et al.*, 2002] B.A. Warnake, M.D. Scott, B.S. Leibowitz, L. Zhou, C.L. Bellew, J.A. Chediak, J.M. Kahn, and B.E. Boserand K.S.J. Pister. An autonomous $16mm^3$ solar-powered node for distributed wireless sensor networks. In *IEEE International Conference on Sensors 2002*, pages 1510–1515, Orlando, FL, USA, June 2002.

[Whitehouse *et al.*, 2004] Kamin Whitehouse, Fred Jiang, Alec Woo, Chris Karlof, and David Culler. Sensor field localization: a deployment and empirical analysis. Technical Report UCB//CSD-04-1349, Univ. of California, Berkeley, April 9 2004.

[Whitehouse *et al.*, 2006] Kamin Whitehouse, Gilman Tolle, Jay Taneja, Cory Sharp, Sukun Kim, Jaein Jeong, Jonathan Hui, Prabal Dutta, and David Culler. Marionette: Providing an interactive environment for wireless debugging and development. In *Proc. of the International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors*, 2006.

[Willig *et al.*, 2005] A. Willig, K. Matheus, and A.Wolisz. Wireless technology in industrial networks. *Proceedings of the IEEE*, 93(6):1130–1151, June 2005.

[Xie and Evans, 1991] X. Xie and R.J. Evans. Multiple target tracking and multiple frequency line tracking using hidden Markov models. *IEEE Trans. Signal Processing*, pages 2659–2676, 1991.

[Xu and Hespanha, 2005] Yonggang Xu and Joao Hespanha. Estimation under uncontrolled and controlled communications in networked control systems. In *Proc. of the 44th IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005.

[Yang and Liu, 1999] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[Zanasi and Morselli, 2003] R. Zanasi and R. Morselli. Discrete minimum time tracking problem for a chain of three integrators with bounded input. *Automatica*, 39:1643–1649, 2003.

[Zhang *et al.*, 2001] W. Zhang, M.S. Branicky, and S.M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–96, 2001.

[Zhang *et al.*, 2004] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi. Hardware design experiences in zebranet. In *In Proc. of the ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[Zhao *et al.*, 2003] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: An information directed approach. *Proceedings of the IEEE*, 91(8):1999–1209, Aug. 2003.

INDEX