Efficient Multi-Target Tracking Using Graphical Models

by

Zhexu (Michael) Chen

S.B. in Electrical Engineering and Computer Science
S.B. in Management Science
Massachusetts Institute of Technology, 2008

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2008

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by .....
Alan S. Willsky
Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Efficient Multi-Target Tracking Using Graphical Models

by

Zhexu (Michael) Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The objective of this thesis is to develop a new framework for Multi-Target Tracking (MTT) algorithms that are distinguished by the use of statistical machine learning techniques. MTT is a crucial problem for many important practical applications such as military surveillance. Despite being a well-studied research problem, MTT remains challenging, mostly because of the challenges of computational complexity faced by current algorithms.

Taking a very different approach from any existing MTT algorithms, we use the formalism of graphical models to model the MTT problem according to its probabilistic structure, and subsequently develop efficient, approximate message passing algorithms to solve the MTT problem. Our modeling approach is able to take into account issues such as false alarms and missed detections. Although exact inference is intractable in graphs with a mix of both discrete and continuous random variables, such as the ones for MTT, our message passing algorithms utilize efficient particle reduction techniques to make approximate inference tractable on these graphs.

Experimental results show that our approach, while maintaining acceptable tracking quality, leads to linear running time complexity with respect to the duration of the tracking window. Moreover, our results demonstrate that, with the graphical model structure, our approach can easily handle special situations, such as out-of-sequence observations and track stitching.

Thesis Supervisor: Alan S. Willsky
Title: Professor of Electrical Engineering

Acknowledgments

It was my pleasure to have the opportunity to follow Professor Alan Willsky from his lectures to the Stochastic System Group (SSG). Prof. Willsky strikes me as not only an outstanding professor, but also a decisive leader, an engaging speaker, and simply someone with rich and well-rounded experience. I thank him for encouraging and supporting me to do what I believe is right, including my pursuit of the MEng degree. This thesis would not have been possible without his insight and guidance. His meticulous revision of this thesis is also greatly appreciated.

I benefited enormously from conversations with fellow graduate students. In particular, Jason Johnson significantly contributed to the research presented in this thesis. Also, Emily Fox, Dmitry Malioutov, and Vincent Tan generously devoted their time to help me understand much background work done prior to this thesis. I thank John Fisher, Mujdat Cetin, Sujay Sanghavi, and Justin Dauwels, for providing me helpful feedback on my research.

I want to thank SSG members, including Venkat Chandrasekaran, Lei Chen, Jin Choi, Ayres Fan, Mike Siracusa, Kush Varshney, and all those mentioned already, for creating a vibrant and supportive environment. They made my life as a graduate student so much more enjoyable.

Last but not least, my successes in school and career development so far are in large part due to the love and encouragement from my family. My parents, Liyun and Guilin, have given me so much guidance, support and encouragement along the way that I can find no adequate words to describe my appreciation. Also, I thank Yinuo for always being the positive drive of my happiness and inspiration during my last three years at MIT. Her love has given me the strength and energy to proceed forward.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Multi-Target Tracking (MTT) is a challenging problem that has a variety of applications, ranging from military target tracking to civilian surveillance. With the recent increased popularity of sensor networks, much research effort has been devoted to developing efficient MTT algorithms that scale well with the problem's size, the benchmarks of which include the duration of the tracking window, the number of targets and the number of sensors. Challenges of MTT are multi-faceted. Some of these challenges are due to practical considerations, while others are more fundamental. Even if we avoid some important practical considerations for MTT systems, including track initialization / termination and potential maneuvering behavior of targets, we are faced with a fundamental problem in which the number of data association hypothesis explodes exponentially with the duration of the tracking window.

This thesis focuses on solving the dynamic tracking problem over time and managing the exponentially exploding complexity experienced by existing methods. The Multi-Hypothesis Tracking (MHT) algorithm has been the preferred solution to MTT for the past two decades. Since the first paper that motivated the development of this field by D. Reid [18], it took us almost 20 years to turn the theory into practical system that can be deployed in the field.

MHT explicitly enumerates all data association hypothesis over time in the form

of a hypothesis tree. The hypothesis tree is extended by one more level when new observations are received each time. Because the size of the hypothesis tree grows exponentially with its depth (which is also the duration of the tracking window), we cannot afford to keep growing the tree forever. Thus, at some point, some hypotheses have to be eliminated. As a decision deferral logic, MHT defers difficult hypothesis pruning decisions until more future data are received. It uses a so-called N-Scan algorithm to manage the size of the hypothesis tree, by collapsing hypothesis that only differ at the back end (root) of the tree (at depth N). One common way to do so is to simply keep the branch with largest likelihoods and remove others. By doing so, information in hypotheses being removed can never be recovered. An alternative is to use a new Gaussian distribution to approximate the Gaussian sum at the root of the hypothesis tree. The parameters of such an approximation often are different from the ones in the original Gaussian sum and may not be a good approximation. In theory, one needs to restart the tracker using the new parameters, increasing the computational costs.

Despite being a popular algorithm, MHT has several problems. First and foremost, the complexity within the tracking window grows exponentially, thus limiting the maximal duration of the tracking window. This issue, compounded with the pruning techniques used in MHT, makes it difficult in the basic MHT algorithm to deal with out-of-sequence data and track stitching with long gaps, which may arise in practice. If such situations happen outside of the current tracking window, then a tracker based on MHT can only make use of the new information (e.g., late data or unambiguous data associations) if the data association hypothesis still exists. Otherwise, the tracker has to roll back the hypothesis tree back to that time and re-grow the tree, which can translate into significant computational costs. Second, although future data are used to prune hypotheses at earlier times, they are never used to correct, or smooth, estimates at earlier times. Part of the reason is because it is not easy to do smoothing on a hypothesis tree.

In this thesis, we take a radically different approach to solve the MTT problem by exploiting the use of graphical models and message passing algorithms. To focus our

attention on solving the dynamic tracking problem over time, we start by assuming that the static data association problem at each single time frame is tractable. The goal of this thesis is to develop an efficient MTT algorithm that performs just as well as any existing methods but leads to much reduced computational complexity. While we are not trying to compete with the most state-of-art MTT algorithm, as we simply do not have the resources and bandwidth to do so, our algorithm, by taking a radically different approach than any existing MTT problam, offers many promising properties, such as achieving a near-linear complexity with respect to the duration of the tracking window and overcoming some inherent limitations faced by existing methods.

We recognize that there are many ways to model the same problem, and the hypothesis tree approach and the graphical model approach are just two of them. Simply by changing the way we model the problem will not change the complexity of solving it. As we know, the exact solution to MTT is exponential in the duration of the tracking window. So is the case for exact MTT using graphical models. The exponential complexity in exact MTT using graphical models is the result of the fact that on the graph there is a mix of both discrete random variables (representing data associations) and continuous random variables (representing target states). Thus, to make target tracking over time tractable, it is necessary to use some approximate inference algorithms.

Approximate inference in mixture models may be done using Nonparametric Belief Propagation (NBP), which, in order to manage the size of messages being passed on the graph, employs a sampling technique to approximate them. Although NBP is an excellent approach to solve approximate inference problems on mixture models in general, it is unnecessary for our particular graphical structure constructed for MTT (as discussed in Chapter 4), for the following two reasons. First, as the first attempt to use graphical models to solve the MTT problem, we have avoided all nonlinearity in the model, by making everything linear Gaussian. Second, on the particular MTT graphical structure that we propose, only a small number of mixture messages are multiplied at each step. NBP is more valuable for inference problems where many

Gaussian mixture messages are multiplied together. Therefore, in this thesis, we employ some deterministic clustering techniques as hypothesis reduction methods.

The value of using the graphical models to solve MTT is also manifested in the flexibility it offers. We want to point out that all hypothesis reductions for the purpose of transmitting messages are only temporary. Unlike MHT in which hypotheses eliminated are permanently lost, all data association hypotheses are kept even though some of them are be approximated in the transmission of messages. As a result, MTT based on graphical models are well-posed to handle the practical adversaries mentioned earlier.

1.2 Contributions

1.2.1 Use of The Graphical Model Formalism

Graphical models are powerful tools to model the probability distributions in a variety of practical applications. We show that the graphical model formalism is well-matched to model the Multi-Target Tracking problem. Moreover, we show that the graphical model structure allows us to circumvent several limitations faced by existing MTT algorithms. In particular, we show that tracking algorithms based on graphical models can readily handle issues such as out-of-sequence data and track-stitching, that often arise in practical MTT systems.

1.2.2 Efficient Message Passing Algorithm for MTT

Once a proper graphical model structure is defined for MTT, one may use any standard inference algorithms, such as Belief Propagation, to solve the tracking problem; however, solving the problem using Belief Propagation in an exact fashion leads to exponential complexity with respect to the duration of the tracking window. We develop several alternatives to reduce the computational complexity, on the premise that they maintain an acceptable level of tracking quality. Most importantly, We show that in many cases, our efficient, approximate message-passing algorithm can

achieve near linear complexity with respect to the duration of the tracking window.

1.3 Thesis Organization

Chapter 2, Background We review some background from the tracking literatures. There are two sub-problems in MTT, namely, one that deals with the static data association problem at a single time frame, and one that solves the dynamic tracking problem over time. Although the latter is the focus of this thesis, we review both topics in this chapter. Also, as the motivation for the development presented in this thesis, we point out some challenges and limitations faced by the existing MHT algorithm.

Chapter 3, Mathematical Preliminaries We review some background from the machine learning and graphical model literatures. We first review some basic concepts of the graphical model formalism, and as an example, we review a particular case of graphical models called Hidden Markov Models (HMM). Lastly, we review a standard inference algorithm used in graphical models, called Belief Propagation. Several miscellaneous topics related to the development of this thesis are also covered in this chapter.

Chapter 4, Exact Multi-Target Tracking Using Graphical Models We use the graphical model formalism to model the MTT problem, and we develop message-passing algorithms to conduct exact inference and tracking on the graphs. When we conduct exact inference on these graphs, we end up dealing with messages that are Gaussian mixture distributions. For this reason, some modifications of the standard Belief Propagation algorithm are required, as discussed in this chapter.

Chapter 5, Efficient Multi-Target Tracking Using Graphical Models Because the exact message-passing algorithm for MTT leads to exponential complexity with respect to the duration of the tracking window, some approximation is required to reduce the complexity to a manageable level. We present three alternatives to

reduce the complexity over time. From our experiments, we conclude that these efficient message-passing algorithms are able to achieve linear time complexity with respect to the duration of the tracking window.

Chapter 6, Conclusion We give the summary of this thesis and suggest several directions for future research and development.

Chapter 2

Background

2.1 Overview

We review the background of Multi-Target Tracking (MTT) in this chapter, the organization of which is as follows. In this Overview section, we review MTT's problem definition, its importance and the challenges it faces. Then, we review the development of data association algorithms for static problems. Although solving static data association problems is not the focus of this thesis (while the dynamic tracking problem is), we believe it is important to understand what has been done in the field of static data association, for the reason that any practical tracking system has to have a static data association component. Lastly, we review the well-known Multi-Hypothesis Tracking (MHT) algorithm that is at the heart of many practical tracking systems.

The starting point of solving the MTT problem is to have a dynamic model and an observation model. A dynamic model is a state-space model that describes how targets' kinetic states evolve over time. An observation model relates noisy observations to targets' true kinetic states. Throughout this thesis, we assume that we are given these two models a-priori or we have estimated them using past data. Then the tracking problem takes the following form.

In tracking multiple targets, we are provided with a sequence of noisy observations about the targets' kinetic states, and we are asked to track, or estimate, the

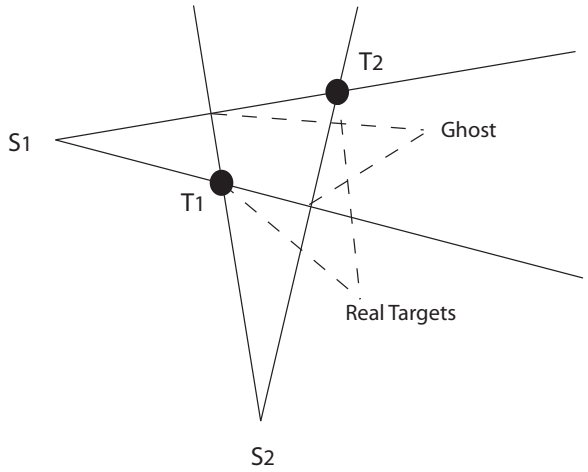


Figure 2-1: Tracking is especially difficult when sensors can only provide bearing-only measurements. In this example, there are two bearing-only sensors tracking two targets, and according to bearing measurements, we are not able distinguish between targets and ghosts.

targets' true kinetic states using these observations. The sequence of observations consists of a set of reports, each of which corresponds to a scan of the surveillance area. Usually, these reports are able to provide some or all information about position, velocity, range, or bearing information about potential targets. Reports are usually provided by radars, airborne sensors, ground or underwater sensors. Major applications of MTT include missile defense systems and surface target tracking. In such applications, it is not uncommon to be tracking up to 1000 targets using tens or perhaps hundreds of sensors. Thus, how to manage the computational complexity of the tracking algorithm is the major challenge of MTT. Moreover, if sensors are only able to provide bearing-only measurements, then the tracking complexity is even greater (see Figure 2-1).

Because reports don't convey targets' identities, tracking algorithms need to solve the data association problem, associating observations with targets. Conventional tracking algorithms, including the Global Nearest Neighbor (GNN) and the Joint Probability Data Association (JPDA), break down the MTT process into two steps. First, upon receiving each report, the tracking algorithm performs data association between measurements and targets. Then, in the second step, according to the observation-to-target assignment computed in the first step, the algorithm uses

some filtering techniques, usually Kalman Filtering[1], to estimate targets' true kinetic states. Such single hypothesis methods provide simple solutions to the data association problem, but they suffer from low accuracy, the detail of which shall be discussed later in this chapter. We shall also review a recent new approach to data association methods - data association based on graphical models.

The dramatic increase in computational capabilities of computers has allowed the development of more powerful Multi-Hypothesis Tracking (MHT) algorithm. Essentially, MHT is a deferred decision logic that postpones difficult data association decisions until more data are received. MHT has proven to be the preferred tracking algorithm for practical systems.

Even though MHT appears to be the most-preferred tracking algorithm, it inherently suffers from an exponential explosion of computational complexity in the length of the tracking time window. There is no good way to manage the number of hypotheses and to prevent it from becoming intractable. As a result, MHT systems[2] can only handle a limited length of the tracking time window. Furthermore, this limitation renders it impossible for MHT to handle out-of-sequence data and occasional track stitching that happen outside of the tracking time window. The ability to handle these two situations is very important in modern sensor networks, as these situations often emerge in large-scale sensor networks.

2.2 Static Data Association

In this section, we review three data association methods for static problems, two conventional methods, and one novel method that is based on graphical models. Data association algorithms strive to produce an observation-to-track assignment that makes tracking most accurate.

[1]Because Kalman Filter is a standard estimation method for any state-space model, we shall review it in the next chapter, Mathematical Preliminaries.

[2]To be more precise, the N-Scan algorithm.

2.2.1 Global Nearest Neighbors

The Global Nearest Neighbor (GNN) [4] method is the simplest data association approach. To associate observations to targets, GNN finds the best (most likely) assignment of observations to tracks, subject to the constraint that one observation can only be assigned to at most one track. The fact that GNN is subject to the above constraint distinguishes itself from the naive Nearest Neighbor (NN) approach where one observation can be assigned to multiple tracks at the same time. GNN initiates new tracks to account for observations that are not assigned to any existing tracks, and it deletes existing tracks that have not received any observations for a period of time.

What comes with GNN's simplicity is its relative low accuracy. The GNN approach only works well when targets are separated. In cluttered tracking scenarios, the GNN approach rigidly assigns an observation to a track, even though there are several closely competing tracks for this observation. As a result, the GNN approach lacks the flexibility to retain multiple hypotheses in ambiguous situations, and, in turn, it does not work well under dense tracking situations. To alleviate this issue, we can increase the error variance matrix in the tracker (usually a Kalman filter) to account for the additional source of uncertainty due to rigid data associations.

2.2.2 Joint Probability Data Association

Contrary to the GNN method, which assigns at most one observation to one track, the Probabilistic Data Association (PDA) and Joint Probabilistic Data Association (JPDA) approach allow one track to be updated with a weighted average of multiple observations from the same point in time. While PDA deals with situations where there is only one track, JPDA deals with situations where there are multiple tracks [1–3].

In the PDA method, given n observations, we can form $n + 1$ hypotheses. The first hypothesis, denoted as H_0, is the data association in which no observation is valid (i.e., all n observations are false alarms). The likelihood for hypothesis H_0 can

be calculated as:

$$Pr(H_0) = (1 - P_D)Pr(n \text{ false alarms}), \tag{2.1}$$

where P_D is the detection probability for the target. Similarly, the probability of the hypothesis H_j, $j = 1, 2, \ldots, n$, where observation y_j is the valid measurement (and the other $n - 1$ observations are false alarms) can be calculated as:

$$Pr(H_j) = p(y_j; T)P_D Pr(n - 1\text{false alarms}), \tag{2.2}$$

where $p(y_j; T)$ is the likelihood for track T to generate observation y_j. The PDA approach updates the track with all $n + 1$ hypotheses. The contribution of each hypothesis is determined by the weight:

$$w_i = \frac{Pr(H_i)}{\sum_{j=0}^{n} Pr(H_j)}. \tag{2.3}$$

In the JPDA approach, because there exists multiple tracks, we need to construct hypotheses in a global fashion over all tracks, and the weights are calculated in a global fashion as well. For example, if we have established two tracks, then upon receiving n new observations, the likelihood of the first hypothesis H_0 that assigns all observations to false alarm is calculated as the follows:

$$Pr(H_0) = (1 - P_D)^2 Pr(n \text{ false alarms}), \tag{2.4}$$

and the likelihood for H_k that assigns y_i to track T_1 and assigns $y_j, j \neq i$ to track T_2 is given by:

$$Pr(H_k) = p(y_i; T_1)p(y_j; T_2)P_D^2 Pr(n - 2 \text{ false alarms}). \tag{2.5}$$

The weights can be calculated using Equation 2.3.

When each track is updated with a weighted average of several observations, the error covariance of each track would be greater than what it really should be. The fact that both GNN and JPDA (as well as PDA) have greater-than-real error covariance

often results in increased ambiguity and unnecessary computational complexity for future data associations[3]. Moreover, the JPDA approach tends to draw closely-spaced tracks together as these tracks are assigned identical observations [5]. As a result, tracks that are closely-spaced would all converge into one track eventually.

2.2.3 Data Association Based on Graphical Models

A recent development in distributed data association was made by Lei Chen. In [6–8], Lei Chen applied the graphical model framework to the problem of multi-sensor multi-target tracking[4]. The challenge is in how to define a graph for the tracking problem such that estimation / inference can be efficiently conducted using message passing algorithms. Before Chen's work, there had been two existing approaches, both of which had their drawbacks. The first approach, called the sensor-centric approach, defines each sensor as a node. In doing so, this approach often ends up with many high-order clique compatibility functions that in turn lead to intractable computational complexity when doing inference. The second approach, called the target-centric approach, defines each target as a node. This approach often produces in a highly connected graph that is badly-posed for inference tasks.

Contrary to both sensor-centric and target centric approaches, Chen proposed an alternative modeling approach - a sensor-target hybrid model. This approach starts with a sensor-centric graph, and it adds an additional target node to the graph if a certain target is seen by more than 2 sensors. It can be shown that, by doing so, the hybrid modeling approach avoids both high-order compatibility functions and highly connected graphs.

Once a graphical model is constructed, we can apply message passing algorithms to conduct inference. Although the graph is not highly connected, it could be quite loopy. While no message passing algorithm is guaranteed to converge in loopy graphs,

[3]This is so because the larger the error covariance, the larger the gating area. We shall review the gating technique later this section.

[4]This subsection involves many machine learning concepts, such as graphical models, loopy graphs, message-passing algorithms, compatibility functions, and Belief Propagation, that are reviewed in 3.2.

more advanced algorithms such as the Tree-Reweighted Max-Product (TRMP) [20] work much better than standard ones such as Belief Propagation. Usually, these algorithms need to be run for many iterations before they converge. Sending messages between nodes on the graph corresponds to sending messages between sensors in the context of sensor networks. Therefore, running distributed inference algorithm in sensor networks could be quite energy-consuming. To save sensors' energy, Chen proposed a communication-sensitive approach that "censors" the messages sent between sensors by comparing the difference between messages sent in the current iteration and the previous iteration.

By using graphical models and their inference algorithms, the work [6–8] bridged between the machine learning community and the tracking community, and thus marked a significant development in the field of multi-sensor multi-target data association. Contrary to the work's focus on data association problems in static, single time frames, this thesis tackles the tracking problem over multiple time frames.

2.3 Dynamic Multi-Target Tracking

In this section, we first review a general Recursive Bayesian Estimation framework that is applicable to any recurring estimation tasks. Then, we briefly discuss the basic setup of a recursive multi-target tracking (MTT) system. Finally, we review the multi-hypothesis tracking (MHT) algorithm and discuss some practical issues and challenges faced by the MHT algorithm.

2.3.1 Recursive Bayesian Estimation

A Recursive Bayesian Estimator (see Figure 2-2) is a general framework for recurring estimation problems in stochastic systems. In terms of the problem setting, we have noisy measurements y_t over the hidden state x_t in a stochastic system. At time $t-1$, according to all observations received up to time $t-1$, the Bayesian estimator predicts (in terms of a probability density function, or PDF) the value of the unobserved random variable x_t at the next time point t. Then, upon receiving a new observation

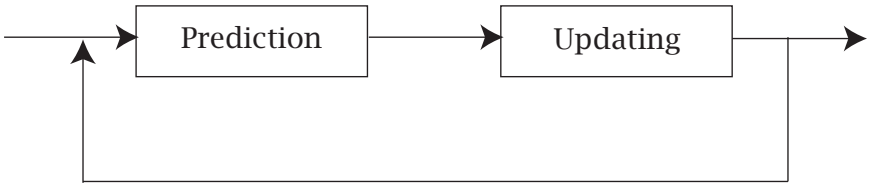


Figure 2-2: A Recursive Bayesian Estimation system predicts the state of the underlying stochastic system at the next time point using measurements up to the current time point. Upon receiving new measurements, it updates its knowledge about the state of the underlying stochastic system. These two steps are carried out recursively.

y_t, the estimator updates its estimation on x_t with the augmented observation set that now consists of all observations up to time $t+1$. Such a prediction and updating process occur recursively over time.

To use the Recursive Bayesian Estimator, we need to model the stochastic system via a Markov state-space transition model and an observation model. The transition model describes the evolution of the hidden states x_t. Since the transition model is Markov, the states only depend on the previous time points

$$x_t = F_t(x_{t-1}, u_{t-1}), \tag{2.6}$$

where u_t is the noise, drawn from an independent, identical distribution (idd). The measurement model takes the form

$$y_t = H_t(x_t, v_t) \tag{2.7}$$

where v_t is also drawn from a idd distribution. With these two models, we can calculate $p(x_t|x_{t-1})$ and $p(y_t|x_t)$.

In the prediction step at time $t-1$, the estimator estimates the PDF of the hidden state at the next time point t as follows

$$p(x_t|y_{1,\ldots,t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1,\ldots,t-1})\, dx_{t-1} \tag{2.8}$$

The updating operation uses the latest observation y_t to update the predicted PDF,

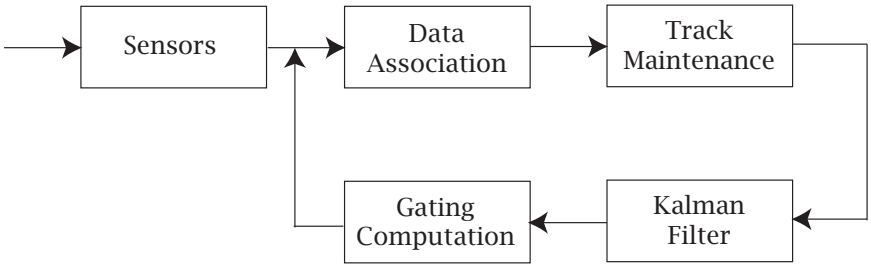


Figure 2-3: Basic setup of a Multi-Target Tracking (MTT) system consists of five core components.

according to the Bayes' rule

$$p(x_t|y_{1,\dots,t}) = \frac{p(y_t|x_t)p(x_t|y_{1,\dots,t-1})}{p(y_t|y_{1,\dots,t-1})} \tag{2.9}$$

where

$$p(y_t|y_{1,\dots,t-1}) = \int p(y_t|x_t)\, p(x_t|y_{1,\dots,t-1})\, dx_t \tag{2.10}$$

Since the estimator is recursive, it obtains $p(x_{t-1}|y_{1,\dots,t-1})$ from the previous iteration.

If both the dynamic state-space transitional model and the measurement model are linear Gaussian, then the recursive procedure is realized using the Kalman Filter, which will be reviewed in Section 3.3.2.

2.3.2 Basic Setup of Recursive MTT

As in Figure 2-3, the basic elements of a conventional MTT tracking system include five components. First, observation data are collected by sensors (e.g., range-only sensors, radar, or unmanned aerial vehicles), the number of which could be on the order of thousands. Second, upon receiving new observations, the tracker computes all possible observation-to-track associations. Note that when there are many targets and many sensors, it is intractable to enumerate all possible data associations. To make data association more computationally feasible, gating constraints are usually imposed. Third, the tracker initiates new tracks for observations that are not assigned to any tracks, and deletes tracks that haven't received any observations for a while. Fourth, with a configuration of data association, the tracker uses a recursive Bayesian

estimator[5] to estimate the targets' positions at the current time and to predict the targets' positions at the next time point. Fifth, gating constraints are computed from the output of Kalman Filter. The location of the gating region is determined by the mean of the prediction, and the size of the gating region is determined by the error variance of the prediction. This process is carried out recursively.

2.3.3 Multi-Hypothesis Tracking

Static data association algorithms, including the GNN method and the JPDA method, force data association decision to be made causally (i.e., the decision at time t is made only using observations up to that time point t.) Under such data association schemes, observation-to-track assignments are irrevocable once assignment decisions are made. As a result, neither the GNN nor the JPDA methods works well under cluttered tracking environments.

Issues such as this have led the Multi-Hypothesis Tracking (MHT) algorithm to become the algorithm of choice when it comes to Multi-Target Tracking [5]. As a deferred decision logic, the MHT algorithm defers difficult data association decisions to a later point in time when more data have been received.

The algorithmic development of MHT was proposed by Reid [18], dating back about three decades. In his original paper, Reid presented an algorithmic approach to formulate hypothesis, evaluate tracks, prune hypothesis, and form clusters. We will review the former two techniques in the "Basics" subsection, and the latter two in the "Hypotheses Reduction" subsection.

Basics The fundamental difference between MHT and static data association methods is that MHT tolerates the ambiguity in the data association problem and thus allows the co-existence of multiple data association hypotheses. To determine which observations could be assigned to each track, a gating technique is used. Once the set of possible observations is determined for each track, MHT formulates new data association hypotheses, in the form of hypothesis trees. Then for each hypothesis,

[5]In linear Gaussian case, the general recursive Bayssian estimator reduces to the Kalman Filter

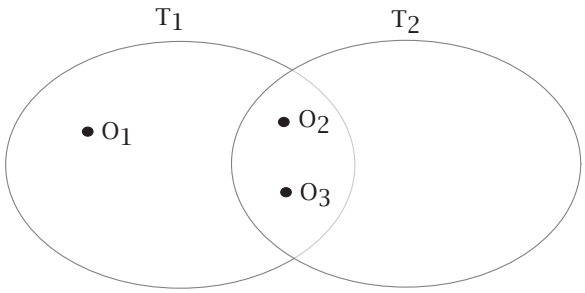


Figure 2-4: In this tracking scenario, there are two existing tracks T_1 and T_2, and three observations O_1, O_2 and O_3. The ellipses represent the gating regions of tracks.

MHT utilizes a Kalman Filter to make predictions and estimations. Below, we review two basic techniques used in MHT: hypothesis formulation and track evaluation.

There are two kinds of hypothesis formulation methods: the global-oriented method and the track-oriented method. In the global-oriented method, we consider all targets together, and we formulate globally compatible data associations (i.e., at most one target can be assigned to one observation from each sensor at each time, and at most one observation from each sensor at each time can be assigned to one target.) For example, in the tracking scenario depicted in Figure 2-4, there are 2 targets T_1, T_2 and 3 observations O_1, O_2 and O_3. We formulate data association hypotheses using the global-oriented method in Table 2.1. This method is conceptually clear, but it is not able to output the most likely assignment for each single track (because hypotheses are considered in a global fashion.)

The second approach, or the track-oriented approach, has two steps. In the first step, it formulates hypotheses for each track, without considering any global compatibility constraints (see Figure 2-5). Then in the second step, we output observation-to-track assignments by picking one hypothesis from each track, with global compatibility constraints. The track-oriented method allows users to consider tracks separately, and therefore is suitable when asked to output only one single assignment for each track.

Once all data association hypotheses are formulated, they are evaluated by a probabilistic expression that takes into account many factors, such as prior knowledge about targets' presence, the probability of false alarms, and the dynamic consistency

	O_1	O_2	O_3
H_1	FA	FA	FA
H_2	FA	FA	T_1
H_3	FA	FA	T_2
H_4	FA	T_1	FA
H_5	FA	T_1	T_2
H_6	FA	T_2	FA
H_7	FA	T_2	T_1
H_8	T_1	FA	FA
H_9	T_1	FA	T_2
H_{10}	T_1	T_2	FA

Table 2.1: For the tracking scenario in Figure 2-4, we formulate 10 observation-to-track hypotheses, using the global-oriented approach. FA denotes false alarms, and T_i denotes track i. Note that, when formulating data association hypotheses using the global-oriented approach, we impose two constraints: at most one target can be assigned to each observation from each sensor at each time, and at most one observation from each sensor at each time can be assigned to each target. To simplify this example, we ignore the possibility of track initiation and deletion.

between new observation and old observations assigned to the same track. Typically this expression is computed in the form of a log-likelihood ratio, or so-called the track score, which is more preferred from a computational efficiency point of view.

Hypotheses Reduction In dense tracking environments, the number of hypotheses in MHT grows exponentially in the length of the tracking time window. Therefore, in any practical multi-target tracking system, there must be hypotheses reduction techniques of some sorts. Here, we review two commonly used hypotheses reduction techniques: clustering and N-Scan.

Essentially, clustering divides a large problem into several smaller problems. A common result of the MHT algorithm is that several tracks only differ in their data associations at a very early point of time, and as a result these tracks all become indistinguishable eventually. Based on this observation, the simplest way to do clustering is to group together tracks that share at least one common observation. Despite its simplicity, such a clustering algorithm often works well in practice, because it significantly reduces the dimension of the data association problem.

An even more commonly-employed algorithm is called the multiple scan, or N-

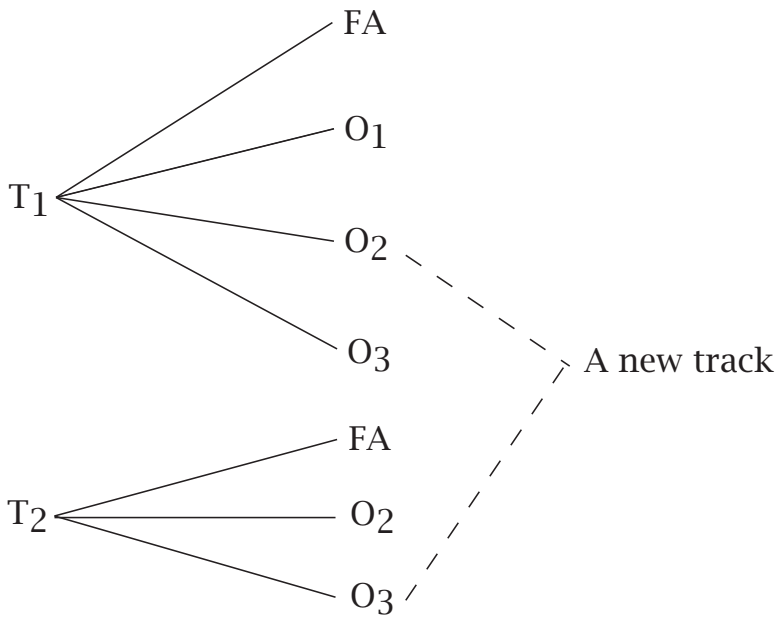


Figure 2-5: For the tracking scenario in Figure 2-4, we formulate observation-to-track hypotheses, using the track-oriented approach. Then, when we output a global data association assignment, we impose the two consistency constraints: at most one target can be assigned to each observation from each sensor at each time, and at most one observation from each sensor at each time can be assigned to each target. Again, to simplify this example, we ignore the possibility of track initiation and deletion.

Scan, algorithm. In fact, the N-Scan algorithm is at the heart of almost all practical multi-target tracking systems. As a deferred decision-making logic, the N-Scan MHT algorithm prunes the hypotheses at time t when observation data have been received at time $t + N$. More specifically, after formulating the hypotheses at time $t + N$, the algorithm calculates the sum of probability mass of hypotheses under each branch at time t, and it keeps the branch with the highest probability mass and removes all other branches. Note that although the traditional N-Scan MHT algorithm utilizes future data to make the pruning decision backwards in time, it does not do smoothing backwards in time; rather, it only does causal filtering.

2.3.4 Challenges

Although the N-Scan algorithm is able to prune hypotheses at the beginning of the window, the number of hypotheses still explodes within the window. In practice, the

length of N-Scan window should scale with the difficulty of the tracking situation. However, because of the exponentially-growing number of hypotheses, the length (N) of the tracking time window is limited. For example, suppose there are 10 targets and that the length of the window in N-Scan is 10. Then, this yields the number of hypotheses at the end of the N-Scan window to an intractable level of about $((10)!)^{10} \approx 4 \times 10^{65}$. Despite some engineering tricks put forth to alleviate this issue, how to manage the exponentially-exploding number of hypotheses is still a great challenge for tracking system researchers.

Another issue that challenges the MHT formalism is unscheduled intervention to the hypotheses tree. Examples of such unscheduled intervention include handling out-of-sequence data (late-arrival data) and incorporating expert (special sensors or human) intervention. It is common for practical, large-scale multi-sensor tracking system to encounter out-of-sequence data, because when the sensor network is large enough, it is inevitable that some data will arrive later than others. Moreover, it is important for tracking systems to have the build-in ability to handle human intervention or infrequent target identity information provided by special sensors.

Suppose, for example, that we receive out-of-sequence data (observed earlier but received much later) that could, if handled properly, eliminate many hypotheses. With the MHT structure, this either requires rolling back the hypothesis tree to that earlier time (a significant computational cost), or making approximate use of the data that may not have the desired effect. Similarly, if data become available that unambiguously make clear that one hypothesis is the "true" hypothesis, MHT can only make use of this information if that hypothesis still exists, which may not be the case if the hypothesis has already been eliminated because of N-Scan limitation. More precisely, the so-called *track stitching* is only possible within the N-Scan window.

It is a contribution of this thesis that, with the graphical model / messaging passing structure, challenges such as these can be addressed without much difficulty or any extra functionality.

Chapter 3

Mathematical Preliminaries

3.1 Overview

In this chapter, we review the mathematical background of several machine learning techniques that are used in the rest of the thesis. In terms of this chapter's organization, first, we review some basic concepts of graphical models. Particularly, we are interested in a special class of graphical models, the Gaussian Markov Random Field (GMRF), as it is the prototype on which we develop our graphical model for tracking applications. Second, we review the Hidden Markov Model (HMM) as an example of the graphical model formalism and the solution to inference problems on HMMs in linear Gaussian cases. Third, we review the standard message passing algorithm for inference problems in graphical models, namely the Belief Propagation algorithm. We shall also briefly review inference in loopy graphs and the Nonparametric Belief Propagation (NBP). Lastly, we review several miscellaneous topics, including the concept of KL-divergence from information theory, clustering algorithms, and a data structure known as a k-dimensional tree (KD-tree).

Recent popularity and success of graphical models can be attributed to two reasons. First, the graphical model framework powerfully captures the sparsity in the statistical dependencies embedded in many practical applications, ranging from computer vision, speech processing to computational biology. Second, message passing algorithms allow us to efficiently solve the inference problem in many graphical models

that are commonly seen. Essentially, by taking advantage of the statistical independence conveyed by the graph structure, message passing algorithms break down the global inference problem into localized operations that can be efficiently computed.

In many applications, we are only interested in random variables that either are all discrete or are all continuous. If graphs are tree-structured and if they consist of only discrete variables or only Gaussian continuous variables, then inference can be solved efficiently using message passing algorithms, such as Belief Propagation. However, if there are loops in the graphs, then Belief Propagation is not guaranteed to converge. What's more difficult is the case where there are continuous variables that are non-Gaussian or where there is a combination of both discrete and continuous variables. In such scenarios, exact inference often is intractable. Unfortunately, the fact that graphical models we construct for tracking applications consist of a mix of both continuous and discrete variables renders it impossible to do exact inference. To tackle these cases, Nonparametric Belief Propagation (NBP) combines standard Belief Propagation and Particle Filtering, to conduct approximate inference. We review NBP in this chapter because the efficient tracking algorithm developed later in this thesis is inspired by NBP.

As part of the approximate inference algorithm developed in this thesis, we make use of two clustering techniques. The first one involves modifying the conventional k-means algorithm, and although it requires more running time, it yields better clustering results than the second cluster technique. The second one involves using the k-dimensional tree to do efficient, coarse clustering. The basics of both the k-means algorithm and the k-dimensional tree are reviewed in this chapter.

3.2 Graphical Models

3.2.1 Brief Review of Graph Theory

We review some graph theoretic terminologies that are commonly used in describing graphical models. A *graph* $\mathcal{G} = (V, \mathcal{E})$ consists of a set of *nodes* or *vertices* V and a

set of *edges* $\mathcal{E}$ that connect pairs of nodes. In an *undirected* graph, an edge $\mathcal{E}_{i,j} \in \mathcal{E}$ if and only if $\mathcal{E}_{j,i} \in \mathcal{E}$. Edges in undirected graphs are denoted pictorially using a line segment. The *neighbors* $\mathcal{N}_i$ of a node i are the ones to which node i connects with an edge. In a *directed* graph, an edge $\mathcal{E}_{i,j} \in \mathcal{E}$ is denoted using an arrow, pointing from the *parent* node i to the *child* node j.

The accuracy and the convergence behavior of message-passing algorithms on a certain graphical model depend on the structure of the graph. A *path* is a route that does not pass any edge more than once. A *loop*, or *cycle*, is a path which ends at the node where it began. A *tree* is a connected acyclic (i.e., loop-free) graph. The *degree* of a vertex in a graph is the number of edges incident to the vertex. A vertex of degree 1 is called a *leaf*. A *clique* in a graph is a set of pairwise adjacent vertices. If A, B, C are three disjoint subset of vertices and every path between A and C passes through some node in B, then we say set B *separates* sets A and C. The *size* of a graph is the number of its edges, $|\mathcal{E}|$.

3.2.2 Graphical Models

A graphical model is a graph $\mathcal{G} = (V, \mathcal{E})$ with a probability distribution $p(X)$ defined according to the structure of the graph, over a collection of variables $X = (x_i | i \in V)$. Edges on the graph represent probabilistic dependencies between the nodes. By putting a graph structure on a large number of random variables, we are exposing the sparsity in probabilistic dependencies between these variables.

There are many different kinds of graphical models, including undirected and directed models. Examples of undirected graphical models include Factor Graphs and Markov Random Field (MRF), while examples of directed ones include Bayesian Networks and Hidden Markov Models (HMM). It is worthwhile to point out that a directed graphical model can be converted into an undirected model at the cost of losing some probabilistic properties, using a conversion procedure, the detail of which, however, is beyond the scope of this thesis. Since in this thesis we shall only be dealing with MRFs, we shall choose to introduce only MRFs.

3.2.3 Markov Random Fields

Before we introduce the main subject of this subsection, let's briefly review probability theory. Consider a pair of random variables (x, y) defined by a probability distribution $p(x, y)$ on a product set $\mathcal{X} \times \mathcal{Y}$. The *marginal distribution* of x is given by $p(x) = \int_{\mathcal{Y}} p(x, y)\, dy$, and the *conditional distribution* of x given y is calculated as $p(x|y) = p(x, y)/p(y)$ for $p(y) > 0$. If $p(x, y) = p(x)p(y)$ for all x, y, then we say x and y are *independent*. Given another set of random variables (x, y, z), if $p(x, y|z) = p(x|z)p(y|z)$ for all z with $p(z) > 0$ and all x, y, then we say x and y are *conditionally independent* given z.

A *random field* consists of a vector of random variables $x_\Gamma = (x_\gamma | \gamma \in \Gamma)$, and a probability distribution $p(x)$ defined over a product set $\mathcal{X}_\Gamma = \prod_{\gamma \in \Gamma} \mathcal{X}_\gamma$. A Markov Random Field (MRF) is a random field with the following Markov property. The set of random variables x_Γ is *Markov* if, given two disjoint subsets of vertices A and C that are separated by another subset of vertices B, x_A and x_C are conditionally independent of variables x_B for any separating set B. That is, for every separating set B of sets A and C, the Markov property implies

$$p(x_A, x_C|x_B) = p(x_A|x_B)p(x_C|x_B) \tag{3.1}$$

The Hammersley-Clifford theorem provides a sufficient condition for the form of the joint probability distribution $p(x)$ in order for it to be Markov with respect to the graph structure. It states that the random vector x_Γ is Markov with respect to graph $\mathcal{G}$ if its distribution can be factored into a product of functions defined on cliques of $\mathcal{G}$:

$$p(x) = \frac{1}{\kappa} \prod_{C \in \mathcal{C}} \psi_C(x_C) \tag{3.2}$$

These functions are called *compatibility* functions, or *potential* functions. We shall be using the two interchangeably in the rest of this thesis.

In the rest of this thesis, we shall focus on a special class of MRFs, namely pairwise MRFs. Given a graph $\mathcal{G} = (V, \mathcal{E})$, a pairwise MRF expresses the joint probability

distribution in terms of the product of potentials on edges and nodes:

$$p(x) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) \prod_{i\in V} \psi_i(x_i) \tag{3.3}$$

where $\psi_{ij}(x_i, x_j)$ is the edge potential between node i and node j, and $\psi_i(x_i, y)$ is the node potential of node i. According to the Hammersley-Clifford theorem, graphical models defined in this way must be Markov with respect to the graph $\mathcal{G}$, because an edge itself defines a clique.

In many applications including target tracking, we are interested in doing inference on a set of hidden, or latent, variables (denoted as x's) based on noisy observations (denoted as y's). Because we are interested in estimating the values of x's based on values of y's, the key quantity of interest to us is $p(x|y)$, which can be expressed using Baye's rule as follows:

$$p(x|y) = \frac{p(x,y)}{p(y)} \propto p(x,y) \tag{3.4}$$

It turns out that focusing on the joint distribution $p(x, y)$ is enough, because the distribution of empirical observations $p(y)$ can be seen as a constant.

If we assume that each observation is localized (i.e., y_i is conditionally independent of all other y's and x's when conditioned on x_i), then we can define a graphical model by factoring $p(x|y)$ into the following form:

$$p(x|y) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) \prod_{i\in V} \psi_i(x_i, y_i) \tag{3.5}$$

where $\psi_{ij}(x_i, x_j)$ is the edge potential, and $\psi_i(x_i, y_i)$ is the node potential. According to the Hammersley-Clifford theorem, the random vector x and y defined by 3.5 is Markov with respect to $\mathcal{G} = (V, \mathcal{E})$.

3.2.4 Gaussian Markov Random Fields

A Gaussian Markov Random Field (GMRF) is an MRF in which the state x is normally distributed. Since the graphical models used in this thesis later have some

attributes of GMRFs, we introduce them in this section.

The joint distribution defined by a GMRF is a normal distribution, and it can be written in two forms, namely, the standard form and the information form. In the regular form, the normal distribution is defined as

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \triangleq \mathcal{N}(x;\mu,\Sigma) \tag{3.6}$$

where $\mu \triangleq \mathrm{E}[x]$ is the mean vector and $\Sigma \triangleq \mathrm{E}[(x-\mu)(x-\mu)^T]$ is the covariance matrix. These two moments (μ, Σ) fully parameterize the structure of the GMRF. According to the definition of a covariance matrix, it has to be positive definite, or semi-positive definite. If the covariance matrix of a GMRF is also positive definite, then we say this GMRF is regular.

For a regular GMRF, we can also define its joint probability distribution in information form:

$$p(x) = \exp\left(-\frac{1}{2}x^T Jx + h^T x + \gamma_0(J,h)\right) \triangleq \mathcal{K}(x;J,h,\gamma_0) \tag{3.7}$$

where J is the information matrix, h is the potential vector, and $\gamma_0(J,h)$ is a normalization function, which are given by

$$\begin{aligned} J &= \Sigma^{-1} && (3.8)\\ h &= \Sigma^{-1}\mu && (3.9)\\ \gamma_0(J,h) &= -\frac{1}{2}h^T Jh + \frac{1}{2}\log\det J - \frac{1}{2}n\log 2\pi && (3.10) \end{aligned}$$

Like moments (μ, Σ), these information parameters (J, h) fully specifies the structure of the GMRF. If the function is not normalized, for example $q(x) = \alpha p(x)$ where α is a constant, then:

$$q(x) = \alpha p(x) = \mathcal{K}(J,h,\gamma_0(J,h) + \log(\alpha)) \tag{3.11}$$

Moreover, the sparsity of the graphical model is reflected in the J matrix. In fact,

one can "read off" the structure of the graph from the J matrix, because $J_{ij} \neq 0$ if and only if $\mathcal{E}_{ij} \in \mathcal{E}$.

In order to construct a MRF based on a normal distribution $p(x)$, according to the Hammersley-Clifford theorem, we need to factor the distribution into a pairwise form (i.e., as a product of functions that involve at most two variables). Such a factorization always exists[1]. Specifically, we factor $p(x)$ into the follows:

$$p(x) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) \prod_{i\in V} \psi_i(x_i) \tag{3.12}$$

where the node and edge potential functions takes the following forms:

$$\psi_i(x_i) = \exp(-\frac{1}{2} x_i^T A_i x_i + h_i^T x_i) \tag{3.13}$$

$$\psi_{ij}(x_i, x_j) = \exp(-\frac{1}{2} \begin{bmatrix} x_i^T & x_j^T \end{bmatrix} B_{ij} \begin{bmatrix} x_i \\ x_j \end{bmatrix}) \tag{3.14}$$

To make the factorization accurate, the A_i and B_{ij} must satisfy the following constraint:

$$x^T J x = \sum_i A_i x_i^2 + \sum_{(i,j)\in\mathcal{E}} \begin{bmatrix} x_i^T & x_j^T \end{bmatrix} B_{ij} \begin{bmatrix} x_i \\ x_j \end{bmatrix} \tag{3.15}$$

As long as this above constraint is satisfied, the decomposition can be arbitrary. Thus, given a probability distribution, there are infinite number of ways to define the potential functions.

3.3 Example: Hidden Markov Models (HMMs)

To apply the formalism of graphical models in solving inference problems, the first step is to construct a graphical model that describes the statistical dependencies embedded in the underlying system. We illustrate the process of constructing a graphical model by reviewing Hidden Markov Models (HMMs). HMMs are central

[1] To see this, write the distribution in the information form, and expand the matrix operations in terms of individual x_i's

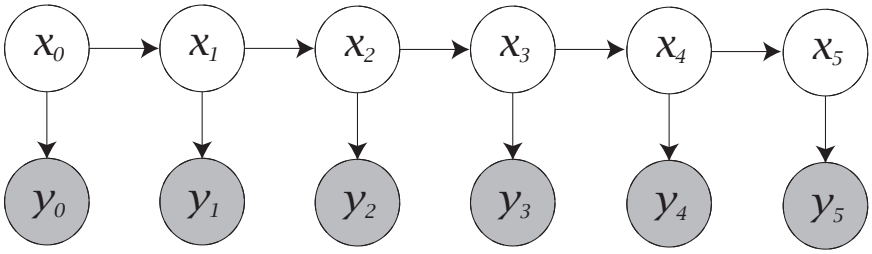


Figure 3-1: This Hidden Markov Model (HMM) describes a temporal Markov process of length 6. Observable nodes are shaded, while hidden nodes are white.

to many practical applications, such as speech recognition and visual object tracking. In this subsection, we first proceed with the definition of HMMs, and then we discuss the inference problem on HMMs and its solution.

3.3.1 Definition

HMMs are used to model temporal stochastic processes. These HMMs can be converted to undirected models without any change to the graph structure nor to the potential functions.

Let $y = \{y\}_{t=0}^{T-1}$ be the conditionally independent observations of a temporal process, and let $x = \{x\}_{t=0}^{T-1}$ be the hidden variables that generate y. Assume that x evolve under a first-order Markov process (i.e., x_t only depends on x_{t-1}). Then the joint probability distribution is given by

$$p(x, y) = p(x_0)p(y_0|x_0) \prod_{t=1}^{T-1} p(x_t|x_{t-1})p(y_t|x_t) \tag{3.16}$$

With this joint distribution, we can define the graph as in Figure 3-1. As in Equation 3.5, we can define the potentials for x conditioned on y as follows:

$$\psi_{t-1,t}(x_{t-1}, x_t) = p(x_t|x_{t-1}) \tag{3.17}$$

$$\psi_t(x_t) = \begin{cases} p(x_0)p(y_0|x_0) & \text{if t=0} \\ p(y_t|x_t) & \text{otherwise} \end{cases} \tag{3.18}$$

While in many applications that involve HMMs (e.g., speech recognition), the state

values are drawn from a finite, discrete space, in this thesis we are interested in cases where states take on continuous values that are modeled by Gaussian distributions. The graphical models developed later in this thesis are built on HMMs, with the addition of discrete nodes. Despite the difference, the methodology of defining the graph and potential functions is similar to the one reviewed in this section.

3.3.2 Inference in Linear Gaussian HMMs

We are interested in solving the inference problem in linear Gaussian, tree-structured HMMs. Given a series of observations $y = \{y\}_{t=0}^{T-1}$, we want to find the posterior probability distribution $p(x_t|y_{1:s})$, where if $s \leq t$ then it's called filtering and if $s > t$ then it's called smoothing. Typically, we solve inference problems in general graphical models using Belief Propagation, which we will review in Section 3.4. It turns out that, in linear Gaussian, tree-structured HMMs, Belief Propagation reduces to Kalman Filter, which shall be the subject of this subsection.

In the control literature, the Kalman Filter is used to estimate the hidden states of linear dynamic systems. In the machine learning literatures, the Kalman Filter is the natural solution to the estimation problem in Hidden Markov Models (HMM) where potential functions are all Gaussian.

The input to the Kalman Filter are two linear Gaussian models, namely, the dynamic model and the observation model. The dynamic model corresponds to a first-order Markov process described by the HMM, and it takes the form:

$$x_t = Ax_{t-1} + u_{t-1} \tag{3.19}$$

where x_t is the state at time t, A is the transition matrix, and u_{t-1} is a white noise with mean zero and covariance matrix Q. The observation model is also modeled by a linear Gaussian model:

$$y_t = Cx_t + v_t \tag{3.20}$$

where y_t is the observation at time t, C is a matrix, and v_t is a white noise $\mathcal{N}(0, R)$.

Filtering When we are interested in estimating the hidden state at a certain time given observations up to that time point, we use the Kalman Filter recursively to compute the following quantities:

$$\begin{aligned}
\hat{x}_{t+1|t} &= A\hat{x}_{t|t} && (3.21)\\
P_{t+1|t} &= AP_{t|t}A^T + P && (3.22)\\
\hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + P_{t+1|t}C^T(CP_{t+1|t}C^T + Q)^{-1}(y_{t+1} - C\hat{x}_{t+1|t}) && (3.23)\\
P_{t+1|t+1} &= P_{t+1|t} - P_{t+1|t}C^T(CP_{t+1|t}C^T + Q)^{-1}CP_{t+1|t} && (3.24)
\end{aligned}$$

where $\hat{x}_{t+1|t}$ is the estimated mean for state vector at time $t+1$ based on all observations up to time t, and $P_{t+1|t}$ is the estimated covariance matrix for the state vector at time $t+1$ based on all observations up to time t.

Smoothing When we are estimating the hidden state using observations both from the past and from the future, the problem is called smoothing. There are two ways to solve the smoothing problem. The first approach computes backward-filtered estimates and combines them with forward-estimates. The second approach calculates directly the filtered-and-smoothed estimates. Here, we only introduce the second approach.

The algorithm we describe here is called the Rauch-Tung-Striebel smoother. In a tracking window of length T, it takes as input the quantities $\hat{x}_{t+1|t}$, $P_{t|t}$, and $P^{-1}_{t+1|t}$ from the filtering algorithm, and it computes the following quantities:

$$\begin{aligned}
\hat{x}_{t|T} &= \hat{x}_{t|t} + L_t(x_{t+1|T} - \hat{x}_{t+1|t}) && (3.25)\\
P_{t|T} &= P_{t|t} + L_t(P_{t+1|T} - P_{t+1|t})P_{t|t}L_t^T && (3.26)
\end{aligned}$$

where $L_t \triangleq P_{t|t}A^T P^{-1}_{t+1|t}$.

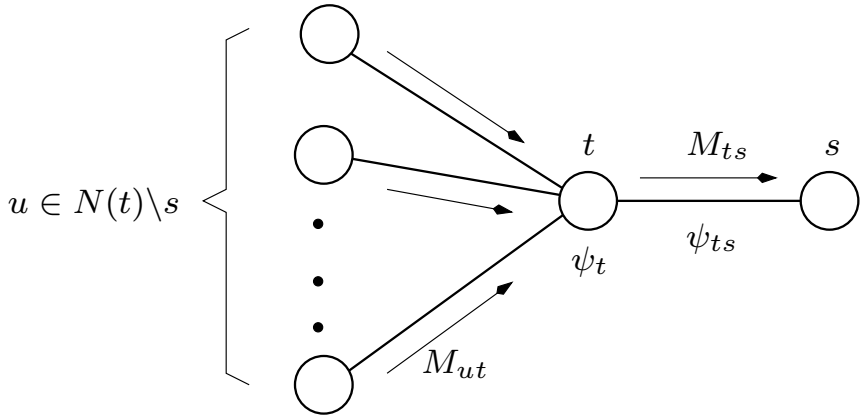


Figure 3-2: In Belief Propagation, to computer a message sent from node t to node s, the node t aggregates messages sent to t from neighbors u, $u \in \mathcal{N}(t)\backslash s$.

3.4 Belief Propagation

Message passing algorithms in graphical models, also known as Belief Propagation (BP) [16], including the sum product algorithm and the max product algorithm, are powerful tools to solve inference problems in graphical models. The sum product algorithm is used to calculate the node marginal distribution, and the max product algorithm is used to calculate the maximal a-posterior (MAP) configuration for the graph. Since the sum product algorithm can be converted to the max product algorithm with an easy change of operator in the equations, our review here focuses on the sum product algorithm.

As a distributed algorithm, the sum-product algorithm iteratively passes messages between nodes. In iteration n, in order to send the message from node t to node s, upon receiving messages from neighbors of node s at the previous iteration (i.e., iteration $n-1$), we calculate the message (Figure 3-2) using the following formula:

$$M_{t\to s}^{n}(x_s) \propto \int_{x_t} \left\{ \psi_{st}(x_s, x_t)\psi_t(x_t) \prod_{u\in\mathcal{N}(t)\backslash s} M_{ut}^{n-1}(x_t) \right\} dx_t \tag{3.27}$$

Equation 3.27 is used for graphical models in which all random variables are continuous. For the discrete case, replace the integration by the summation. In either case, we can normalize the messages such that they integrate (or sum) to one.

For acyclic, connected graphs, messages computed by the sum-product algorithm are guaranteed to converge in a finite number of iterations. The final steady-state messages (denoted as $M^*_{us}(x_s)$) are used to calculate the marginal distributions:

$$p(x_s) \propto \psi_s(x_s) \prod_{u \in \mathcal{N}(s)} M^*_{us}(x_s). \tag{3.28}$$

The output of equation 3.28 should be normalized to one. By replacing the integration (or summation) in equation 3.27 by a max operator, we arrive at the update equation for the max-product algorithm.

For graphical models with discrete random variables, running BP involves matrix operations. For graphical models with general continuous random variables, running BP can be hard to implement. However, if all random variables on the graph are Gaussian (i.e., the model is a GMRF), then BP equations can be simplified to a series of matrix / vector operations.

Messages in Gaussian models can be parameterized in the information form. The message from node t to node s in iteration n is given by:

$$M^n_{t \to s}(x_s) \triangleq \exp(-\frac{1}{2}\Delta J^n_{t \to s} x_s^2 + \Delta h^n_{t \to s} x_s) \tag{3.29}$$

Due to this information form representation, we just need to pass around parameters $(\Delta J^n_{t \to s}, \Delta h^n_{t \to s})$. Note that multiplication of two Gaussian random variables, expressed in information form, parameterized by J and h matrices, is equivalent to addition of their J and h matrices, respectively. And integration of a such a variable is equivalent to Gaussian Elimination on its J and h matrices. Below, we give the information equivalent of equation 3.27:

$$\Delta J^n_{t \to s} = -J_{st} \left(J_{tt} + \sum_{u \in \mathcal{N}(t) \setminus s} \Delta J^{n-1}_{u \to t} \right)^{-1} J_{st} \tag{3.30}$$

and

$$\Delta h^n_{t\to s} = -J_{st}\left(J_{tt} + \sum_{u\in\mathcal{N}(t)\setminus s}\Delta J^{n-1}_{u\to t}\right)^{-1}\left(h_t + \sum_{u\in\mathcal{N}(t)\setminus s}\Delta h^{n-1}_{u\to t}\right) \tag{3.31}$$

where J_{ts} comes from the edge potential, and J_{tt} and h_{tt} come from the node potential respectively. Note that, given a joint probability distribution, there are multiple ways to factor the distribution and to define edge and node potentials. These J_{ts}, J_{tt} and h_{tt} only correspond to a particular factorization of the joint probability distribution.

Once messages have converged, we can calculate the marginals. The information form equivalent of equation 3.28 is:

$$\hat{J}_t = J_{tt} + \sum_{u\in\mathcal{N}(t)}\Delta J^*_{u\to t} \tag{3.32}$$

and

$$\hat{h_t} = h_t + \sum_{u\in\mathcal{N}(t)}\Delta h^*_{u\to t} \tag{3.33}$$

The form of 3.30 and 3.31 coincides with that of Gaussian Elimination. As a result, inference in GMRFs can be solved by any methods that are developed to solve Gaussian Elimination, and iterative massage passing is one of these methods [17].

3.4.1 Loopy Belief Propagation

In many practical problems, it is impossible to construct a loop-free graphical model that exploits the sparsity in the data in the maximal fashion. In such difficult problems, we need to construct a so-call *junction tree* to conduct inference, the complexity of which is exponential with respect to the size of the largest clique of the graph [15]. To avoid that complexity, we have to use approximate inference algorithms. Therefore, in many practical problems, we have to make the tradeoff between running time and accuracy. In this thesis, we need to make the same tradeoff as well, and we shall defer such discussion to later in the thesis.

3.4.2 Nonparametric Belief Propagation

Because the tracking algorithm developed in this thesis was inspired by the development of NBP, we believe that it is important to understand the basic ideas behind NBP and thus we briefly review it here.

The difficulty of doing inference on a certain graphical model not only has to do with whether the graph has loops, but also has to do with the types of random variables. For tree-structured graphs with only discrete random variables or only Gaussian continuous variables, the inference problem can be solved efficiently by Belief Propagation. However, in general, inference becomes much harder for non-Gaussian continuous variables, including Gaussian mixtures. In fact, there is no closed-form solution to the inference problem on graphs with non-Gaussian random variables.

Nonparametric Belief Propagation (NBP) [19] was developed to address the inference problem on graphical models with general continuous variables. As the Particle Filter is the variant of Kalman Filter for non-Gaussian models, NBP is the variant of Belief Propagation for non-Gaussian graphical models. By using sampling methods, NBP addressed three major challenges as outlined below.

First, NBP uses kernel density methods to describe arbitrary continuous probability distributions. More specifically, messages in NBP are represented by a sum of several Gaussian distributions with different means and the same covariance. Each mode is termed a *particle*. Second, NBP uses a sampling method to avoid the explosion of number of particles when messages are multiplied together. Third, while it takes $O(M^d)$ time to construct the exact product of d Gaussian mixtures with M modes each, NBP uses a Gibbs sampler to approximate the product in time $O(dM^2)$ without explicitly constructing the product.

3.5 Other Topics

3.5.1 KL-Divergence

In information theory, the Kullback-Leibler (KL) Divergence [14], as defined below, is an important similarity metric between two probability distributions. Let p and q be two probability distributions on $\mathcal{X}$, then the Kullback-Leibler (KL) divergence of p relative to q is given by:

$$D_{KL}(p\|q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}\, dx \tag{3.34}$$

It can be shown that $D(p\|q)$ is always non-negative and is zero if and only if $p(x) = q(x)$ for essentially all x [9]. Although KL-divergence can be considered as a way to measure the "distance" between two probability distributions, it is not a distance metric in the sense that it is asymmetric (i.e., $D(p\|q) \neq D(q\|p)$). The symmetrised KL-divergence is defined as:

$$D_{SKL}(P, Q) = D(p\|q) + D(q\|p) \tag{3.35}$$

Relevant to the development of this thesis, the KL-divergence between two Gaussian distribution $\mathcal{N}_0(\mu_0, \Sigma_0)$ and $\mathcal{N}_1(\mu_1, \Sigma_1)$ is given by

$$D_{\mathrm{KL}}(\mathcal{N}_0\|\mathcal{N}_1) = \frac{1}{2}\left(\log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) + \mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) - N\right) \tag{3.36}$$

3.5.2 Clustering Algorithms

In recent years, statistical machine learning techniques have played an important role in the development of clustering algorithms. With a set of input data points, these algorithm strive to separate them into different clusters, according to some distance metric. Popular clustering algorithms include the k-means algorithm [12], the EM algorithm [10] and Affinity Propagation [11]. Because in this thesis we augmented the

k-means algorithm such that it could be used to cluster a set of distributions (rather than data points), we review its basics here.

The conventional k-means algorithm begins by partitioning all data points into k clusters, and then it calculates the mean points, or centroids, of each cluster. Then, it does the following two steps iteratively until convergence.

- Constructs a new partition by associating each point with the nearest centroid.
- Calculates the new centroids of each cluster specified the new partitioning

Clustering algorithms typically only apply to data points in Euclidian space, rather than distributions. However, if we can define a similarity metric for distributions, then most clustering algorithms can be modified to cluster distributions as well.

3.5.3 KD-Trees

A K-Dimensional Tree (KD-tree) is a space-partitioning data structure used to organize a large set of data points. KD-trees have been used extensively in search algorithms that involve high-dimensional data. In this thesis, we are motivated to use KD-trees not only because it is an effective tool to organize high-dimensional data (such as the ones in tracking applications), but also because it stores data in a multi-scale fashion, which allows us to develop efficient, coarse clustering algorithms that are of special interests to tracking applications.

As a binary tree, a KD-tree splits the data set into two equal-sized subsets at each split (non-leaf node). Each leaf node corresponds to a data point in the data set, and each non-leaf node in a KD-tree stores some aggregate statistics of all the leaf nodes that are its children. In other words, a KD-tree representing N data points would have a depth of $\log(N)$ and N leaves (see Figure 3-3 for an example).

Usually, at each split, a KD-tree splits the K-dimensional data set on one of the data set's K axis, and how to pick this axis has to do with the construction methods of the KD-tree. Although there are many different ways to construct a KD-tree and the criterion for making that choice depends on the actual application, it is beyond the

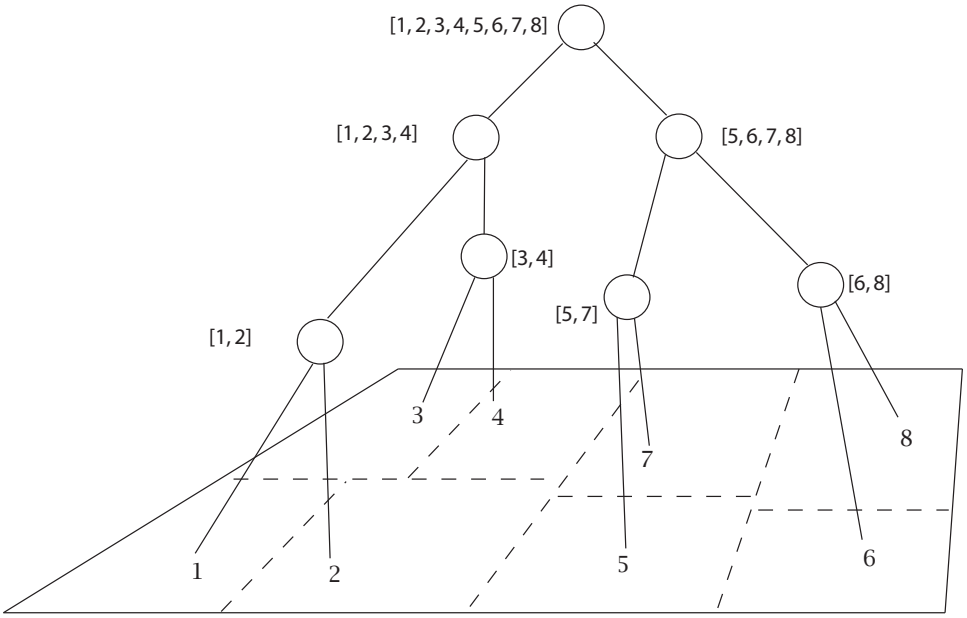


Figure 3-3: KD-Tree is a multi-scale data structure used to organize a set of k-dimensional data points. At each split of the tree, the data set is split into two subsets. Every leaf node corresponds to an original data point, and a non-leaf node stores some aggregate statistics of the original data points that are its children. In constructing a KD-tree, we divide the data set along the dimension that are of largest variance. In this graph, there are 8 data points in a 2-dimensional space. The numbers next to each non-leaf node represent the children of the node.

scope of this thesis to engage in an in-depth discussion of these construction methods. In this thesis, we shall use one of the simplest construction methods [13] as discussed below.

The method we use is a top-down procedure that recursively divides the data set until the set is a singleton. At each split, we calculate the variance of the data set, and pick the dimension that has the largest variance among all K dimensions to conduct the split. Then, we line up the data points in the set and divide the set into two at the median[2]. The smaller ones are assigned to the left children, and the larger ones are assigned to the right children.

Once all data points are put into the tree, we start to calculate the aggregate statistics for each node. Since leaf nodes correspond to data points themselves, no aggregate statistics is needed for leaf nodes. At each non-leaf node, we store the

[2]We use median, because it is resistant to outliers.

aggregate statistics of the node's children. What we store depends on our specific application. Since our tracking application requires clustering data points according to their spatial relationship, we choose to store three statistics at each non-leaf node: weight, mean, and variance. Consider a node s, and denote its left and right children as L and R, then the weight, mean and variance of node s are given by:

$$w_s = w_L + w_R \tag{3.37}$$

$$\mu_s = \frac{w_L}{w_s}\mu_L + \frac{w_R}{w_s}\mu_R \tag{3.38}$$

$$\mu_s\mu_s^T + \Sigma_s = \frac{w_L}{w_s}(\mu_L\mu_L^T + \Sigma_L) + \frac{w_R}{w_s}(\mu_R\mu_R^T + \Sigma_R) \tag{3.39}$$

Here, we assume data points are represented in vector form. We calculate the aggregate statistics for non-leaf nodes starting from the bottom of the tree, and work all the way up to the root of the tree. If the KD-tree is used to store a weighted set of data points, then the leaves correspond to the original data points (i.e., they have non-zero values for μ's, but zero values for Σ's). Nodes at the second level (from the bottom) store aggregate statistics of their child nodes, which are simply data points. Then, starting from the third level (from the bottom), we can use the above set of equations to calculate the nodes' statistics.

In this thesis, we want to be able to organize all the modes in a Gaussian mixture distributions using a KD-tree. There are two methods to do so. In the first approach, we construct the KD-tree based on only the means of the modes in the Gaussian mixture. Each leaf would correspond to a single mode in the Gaussian mixture, and thus would have its own mean, covariance, and weight. Then, we can use the above set of equations to calculate the statistics in the KD-tree starting from the second level from the bottom. By doing so, modes with similar mean would be cluster together by the construction process, regardless how different their covariances are when compared with each others'.

In the second approach, for each mode in the mixture, we collapse entries in the mean vector and the covariance matrix into one big vector, and construct the KD-tree based on that vector. By doing so, modes with similar means but different covariances

would still be separated. This is useful in the context of tracking, because if there are two targets that are closely located but one moves much more erratically than the other, then we do want to separate the two targets into different clusters. Building a KD-tree using the second approach can help us do so.

Chapter 4

Exact Muti-Target Tracking Using Graphical Models

4.1 Overview

When it comes to doing inference among a large set of random variables, graphical models are great tools to exploit the sparsity in the probabilistic dependencies among random variables. In this chapter, we first present the approach we take to construct graphical models for Multi-Target Tracking (MTT) applications. Then, with a set of assumptions on target dynamics and sensor characteristics, we present the process we used to define potential functions and to run message-passing algorithms. Lastly, we point out some connections between our message-passing algorithm and the conventional Multi-Hypothesis Tracking (MHT) algorithm.

As in MHT, the computational complexity in our message-passing algorithm, if carried out in an exact fashion, grows exponentially with regard to the duration of the time window. In this chapter, we present how exact tracking is done using graphical models and message-passing algorithms, then in the next chapter, we shall present several innovative ways to manage the computational complexity to a level as low as linear with regard to the duration of the time window.

There are three major challenges in solving MTT using graphical models. The first one is in picking the right graphical structure, one that faithfully represents

the underlying probability dependencies in MTT and one in which efficient inference can be conducted. It turns out that there are multiple graphs one can construct to solve the MTT problem, and the issue is in choosing one that improves the tradeoff between computational complexity and tracking accuracy. The second challenge is in developing a message-passing algorithm to conduct exact inference (i.e., tracking). As we shall see in this chapter, running Belief Propagation in tracking graphs requires a change in the message update equations, due to the fact that messages are Gaussian mixtures. Last but not least, the toughest challenge is how to do approximate, efficient inference on these tracking graphs.

As we shall see in the next chapter, the graphical model structure we propose allows us to solve the MTT problem with enormous efficiency and flexibility. But before that, in this chapter, we address the first two of these challenges, by presenting the graphical structure and corresponding exact (as opposed to approximate) message-passing algorithm for MTT.

4.2 Structures of Graphical Models

The challenge of constructing graphical models for Multi-Target Tracking (MTT) lies in the question of how to seamlessly model both the filtering problem and the data association problem. In terms of their own nature, the filtering problem is a continuous one, while the data association problem is a discrete one. Thus, it is natural to use continuous nodes to represent target kinematic states, and to use discrete nodes to represent data associations at each time point.

To be more specific, we use continuous random variables, denoted by circles on graphs, to represent target kinematic states, which typically include position, velocity and acceleration, or a subset of these. We call these *target nodes*. Then, we use discrete random variables, denoted by squares on graphs, to represent data associations between observations and targets, subject to two constraints. First, no target can be assigned more than one observation from each sensor at each point of time. Second, no observation from each sensor at each point of time can be assigned to more than

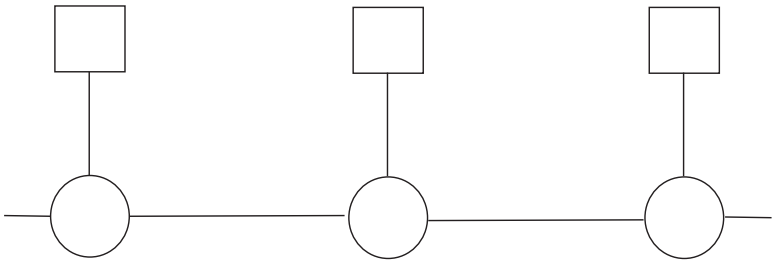


Figure 4-1: As the simplest graph one can construct for multi-target tracking, this graph collapses all targets and all sensors at a single point of time into one target node and one data association node, respectively.

one target. Also note that we include the possibility of false alarm and missed detection. In a false alarm, the observation is not assigned to any target; in a missed detection, a target is not assigned with any observations. We call such discrete nodes *assignment nodes*.

Although the static data association problem at a single point in time is already a challenging problem, it is not the focus of this thesis; rather, the focus here is in finding an efficient way to do tracking over a period of time. For more elaborated work on using graphical models to solve the static data association problem, see Section 2.2.3 and [6–8].

We present four different ways to construct graphical models for multi-target tracking. Each model has its advantages and disadvantages, because of the tradeoff between computational complexity and tracking accuracy. In the first model (Figure 4-1), at each time point, all targets are lumped together to form one global target node and all sensors are lumped together to form one global assignment node. Here, every assignment node takes on discrete values, each of which represents a possible global data association assignment at that time point. Each target node is the collection of kinematic states of all targets at that time point: $x_t = [x_{t,1}^T, x_{t,2}^T, ..., x_{t,M}^T]^T$, where $x_{t,i}$ is the kinematic state of target i at time t.

If there are M targets and K sensors, then the complexity to enumerate global data associations at a single point of time is $(M!)^K$. To reduce that complexity, we propose a second model (Figure 4-2), in which global assignment nodes at each point in time are separated, reducing the complexity of data association at each point of

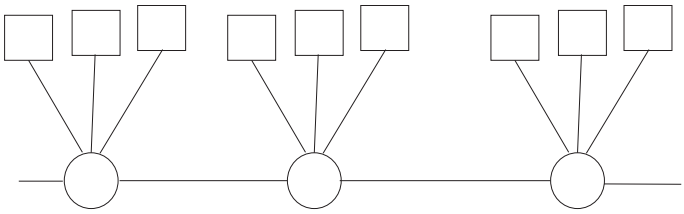


Figure 4-2: As a more elaborated variant of the first graph, this graph separates global assignment nodes at each point of time into individual data association nodes, reducing the complexity to enumerate all data association hypothesis.

time to $K(M!)$. Note that each assignment node now corresponds to a sensor, and the value of such an assignment node indicates the data association between observations generated by that sensor and the targets it observes.

Analogous to separating global assignment nodes into local assignment nodes that correspond to individual sensors, we can separate global target nodes into individual target nodes that correspond to individual targets. There need not be an edge connecting two different targets, as long as targets all move independently, which is assumed throughout this thesis[1]. In particular, because targets move independently, the J matrix in the global kinematic state distribution is a block diagonal matrix. To exploit the sparsity in the J matrix, we construct the third graph (Figure 4-3), in which target nodes are separated within each time point. If we separate global assignment nodes into individual nodes corresponding to individual sensors, then we would have a fully distributed graph (Figure 4-4).

Although the third model and the fourth model better exploit the sparsity in the data than the first model and the second model do, there are loops in these graphs (see the arrows on Figure 4-3). Since belief propagation (BP) is not guaranteed to converge in loopy graphs, running BP on these loopy graphs may not yield the correct result. Furthermore, similar to MHT (Chapter 2.3.3), there has to be some global consistency check when outputting estimated positions in graphs where targets are separated, and the complexity of such a consistency check is another layer of

[1]If targets don't move independently and yet we want to separate them into different nodes, we need to add edges between targets that have probabilistic dependencies between each other.

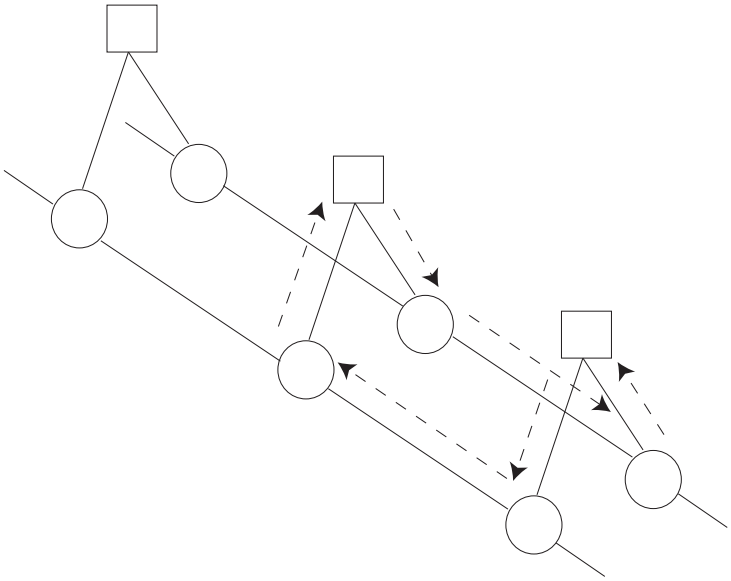


Figure 4-3: When targets are separated to different nodes while all sensors are aggregated into one data association node, although the resulting graph has lower computational complexity than the previous two, it contains loops, as illustrated by the arrow dashed lines on the graph.

complexity for fully distributed models as Figure 4-3 and Figure 4-4.

In this thesis, we shall focus our attention exclusively on the first and second models. For the sake of notational and theoretical simplicity, we derive our formula using model 1, while for large scale experiments we use model 2 to expedite the algorithm.

4.3 Dynamic Models and Observation Models

From now on, in all our derivations, we focus on Figure 4-1 where, at each point in time, all sensors are lumped into one assignment node, and all targets are lumped into one target node. We assume there are M targets, K sensors, and at time t sensor k produces $O_{t,k}$ observations.

Target dynamics and sensor observations are modeled by linear Gaussian models. Target kinematic states at a certain time point depend on those at the previous time point, and a white noise:

$$x_t = Ax_{t-1} + u_{t-1} \tag{4.1}$$

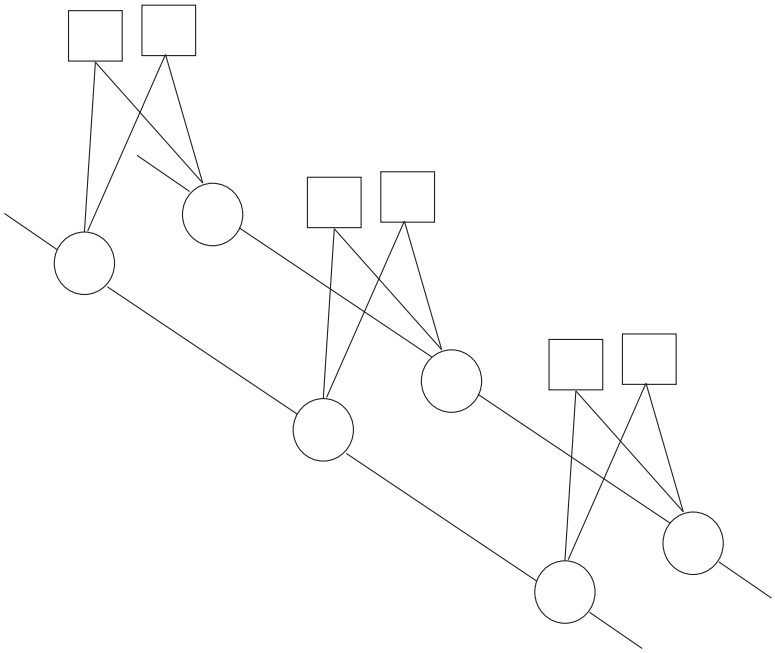


Figure 4-4: When the graph is fully distributed (i.e., all targets and all sensors are separated), although the resulting graph has the lowest computational complexity compared with all previous graphs, it is highly loopy, a situation under which Belief Propagation is not guaranteed to converge.

where x_t is the kinematic state at time t of all targets, or more elaborately:

$$x_t = \begin{pmatrix} x_{t,1} \\ x_{t,2} \\ \vdots \\ x_{t,M} \end{pmatrix}$$

where $x_{t,m}$ is the kinematic state of target m at time t. In Equation 4.1, A is a transition matrix, and u_{t-1} is a stationary white noise with mean zero and covariance matrix Q. The transition matrix A and the noise covariance Q could be either time-dependent or time-independent. In our experiments, we assume a stationary, time-independent transition matrix.

Because from a tracker's perspective, it never knows the true correspondence between observations and trackers, we here introduce an assignment variable, indicating such a correspondence. Suppose sensor k has $O_{t,k}$ observations at time t, then for the

ith observation ($i = 1, \ldots, O_{t,k}$), the assignment variable is defined as:

$$a_{t,k}(i) = \begin{cases} 0 & \text{if observation } i \text{ is assigned as a false alarm} \\ m & \text{if observation } i \text{ is assigned to target } m \end{cases}$$

By collapsing all assignment variables at the same time, we have the global assignment variable at time t as:

$$a_t = \begin{pmatrix} a_{t,1} \\ a_{t,2} \\ \vdots \\ a_{t,K} \end{pmatrix}$$

At time t, we use $y_{t,k}$ to denote all the observations produced by sensor k. Then, we use a linear Gaussian model to model sensor characteristics. If the ith observation from sensor k at time t is not assigned to false alarm, then its value depends on the kinematic state of target $a_{t,k}(i)$ at time t:

$$y_{t,k}(i) = C_{t,k} x_{t,a_{t,k}(i)} + v \tag{4.2}$$

where $C_{t,k}$ is a matrix, and v is a stationary white noise $\mathcal{N}(0, R)$. By collapsing all observations at the same time, we denote the set of observations at time t as:

$$y_t = \begin{pmatrix} y_{t,1} \\ y_{t,2} \\ \vdots \\ y_{t,K} \end{pmatrix}$$

As shown in Equations 4.1 and 4.2, the kinematic state is a Markov process, while the likelihood of the observation depends on the assignment variable and the kinematic state of the target that is assigned to the observation. In the next section, we make use of this dependency structure to define potential functions on our graphs.

4.4 Definitions of Potential Functions

Once a graphical structure has been defined, we can define its potential functions according to the joint probability distribution. Here, we show a definition of potential functions that is based on a particular factorization of the joint probability density function. Again, we here focus our attention to the graph structure given in Figure 4-1. Definitions of potential functions for other graphs discussed previously can be found similarly.

For a time period from $t = 1$ to $t = T$, let X denote the kinematic states of all targets at all time points, Y denote the collection of all observations from all sensors at all time points, and $\mathcal{A}$ denote all data association assignments for all observations at all time points. Then global joint probability for the whole time window is:

$$\begin{aligned} p(X, Y, \mathcal{A}) &= p(Y|\mathcal{A}, X)p(\mathcal{A}|X)p(X) && (4.3) \\ &= p(Y|\mathcal{A}, X)p(\mathcal{A})p(X) && (4.4) \\ &= \prod_{t=1}^{T} p(y_t|a_t, x_t) \cdot \prod_{t=1}^{T} p(a_t) \cdot p(x_1) \prod_{t=1}^{T-1} p(x_{t+1}|p_t) && (4.5) \end{aligned}$$

where x_t, y_t, and a_t are hidden target kinematic states, observations, and assignment variables at time t. The last equality follows from the dynamic model and the observation model. To ease computations, we define all potentials in terms of their information forms.[2] Below, we first define the node potentials and then proceed to define the edge potentials.

We define the potential function for an assignment node in such a way that it takes into account the effects of false alarm and missed detections. Suppose at time t, sensor k observes $O_{t,k}$ observations, in which $O_{t,k}^{FA}$ of them are assigned as false alarms and $O_{t,k}^{DT}$ of them are assigned to targets (i.e., $O_{t,k}^{DT}$ targets are detected by sensor k at time t). Then, the node potential for assignment node a_t is:

$$\Psi_a(a_t) = p(a_t) = \prod_{k=1}^{K} p(a_{t,k}) = \prod_{k=1}^{K} P_D^{O_{t,k}^{DT}} (1 - P_D)^{M - O_{t,k}^{DT}} P_{FA}^{O_{t,k}^{FA}} (1 - P_{FA})^{O_{t,k} - O_{t,k}^{FA}} \quad (4.6)$$

[2] We introduced the $\mathcal{K}(x; J, h, \gamma)$ notation in Equations 3.7 and 3.11.

where M is total number of targets, P_D is probability of detection, and P_{FA} is probability of false alarm. The potential function for target nodes is given by:

$$\Psi_x(x_t) = \begin{cases} 1 & \text{if } t > 1 \\ p(x_1) = \mathcal{N}(x_1; \mu_1, \Sigma_1) = \mathcal{K}(x_1; J_1, h_1, \gamma_0(J_1, h_1)) & \text{if } t = 1 \end{cases}$$

where μ_1, Σ_1, J_1, and h_1 are initial parameters specified by the user when the tracker is started.

Edge potentials are derived according to the dynamic model and the observation model. Again, a target node at time t here represent the collection of all target kinematic states at time t. For the edge connected two target nodes x_t and x_{t+1}, its potential is given by:

$$\begin{aligned} \Psi_{t,t+1}(x_t, x_{t+1}) &= p(x_{t+1}|x_t) \\ &= \frac{1}{\sqrt{\det(2\pi Q)}} \exp\left(-\frac{1}{2}(x_{t+1} - Ax_t)^{\mathrm{T}} Q^{-1}(x_{t+1} - Ax_t)\right) \\ &= \frac{1}{\sqrt{\det(2\pi Q)}} \exp(-\frac{1}{2}(x_{t+1}^{\mathrm{T}} Q^{-1} x_{t+1} - x_{t+1}^{\mathrm{T}} Q^{-1} A x_t - x_t^{\mathrm{T}} A^{\mathrm{T}} Q^{-1} x_{t+1} \\ &+ x_t^{\mathrm{T}} A^{\mathrm{T}} Q^{-1} A x_t)) \\ &= \frac{1}{\sqrt{\det(2\pi Q)}} \exp\left(-\frac{1}{2} \begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} A^{\mathrm{T}} Q^{-1} A & -A^{\mathrm{T}} Q^{-1} \\ -Q^{-1} A & Q^{-1} \end{bmatrix} \begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix}\right) \\ &= \mathcal{K}\left(\begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix}; J, h, \gamma\right) \end{aligned} \tag{4.7}$$

where A is the transition matrix, and Q is the covariance matrix of the noise, and J, h and γ are given by:

$$J = \begin{bmatrix} A^{\mathrm{T}} Q^{-1} A & -A^{\mathrm{T}} Q - 1 \\ -Q^{-1} A & Q^{-1} \end{bmatrix} \tag{4.8}$$

$$h = 0 \tag{4.9}$$

$$\gamma = -\frac{1}{2} \log \det(2\pi Q) \tag{4.10}$$

For edges connecting assignment nodes and target nodes, we derive their potential functions as follows:

$$
\begin{aligned}
\Psi_{a,x}(a_t, x_t) &= p(y_t|a_t, x_t) \\
&= \prod_{k=1}^{K} p(y_{t,k}|a_{t,k}, x_t) \\
&= \prod_{k=1}^{K}\prod_{i=1}^{O_{t,k}} \mathcal{N}(y_{t,k}(i); C_{t,k}x_{t,a_{t,k}(i)}, R) \\
&= \prod_{k=1}^{K}\prod_{i=1}^{O_{t,k}} \frac{1}{\sqrt{\det(2\pi R)}} \exp\left(-\frac{1}{2}\left(y_{t,k}(i) - C_{t,k}x_{t,a_{t,k}(i)}\right)^{\mathrm{T}} R^{-1}\left(y_{t,k}(i) - C_{t,k}x_{t,a_{t,k}(i)}\right)\right) \\
&= \prod_{k=1}^{K}\prod_{i=1}^{O_{t,k}} \exp\left(-\frac{1}{2}x_{t,a_{t,k}(i)}^{\mathrm{T}} J_{t,k} x_{t,a_{t,k}(i)} + h_{t,k}(i)x_{t,a_{t,k}(i)} + \gamma_{t,k}(i)\right) \\
&= \prod_{k=1}^{K}\prod_{i=1}^{O_{t,k}} \mathcal{K}(x_{t,a_{t,k}(i)}; J_{t,k}, h_{t,k}(i), \gamma_{t,k}(i))
\end{aligned}
$$

where $C_{t,k}$ is a matrix given by the observation model, R is the covariance matrix of the observation noise, and the parameters in the information form representation is given by:

$$
\begin{aligned}
J_{t,k} &= C_{t,k}^T R^{-1} C_{t,k} && (4.11) \\
h_{t,k}(i) &= C_{t,k}^T R^{-1} y_{t,k}(i) && (4.12) \\
\gamma_{t,k}(i) &= -\frac{1}{2} y_{t,k}^{\mathrm{T}}(i) R^{-1} y_{t,k}(i) - \frac{1}{2}\log\det(2\pi R) && (4.13)
\end{aligned}
$$

Note that, given a joint probability distribution, there are multiple ways to define the potential functions on the graph. We choose this set of definition, because, as we shall see in Section 5.2, it allows us to interpret the messages sent between target nodes as the estimates of target states (both forward and backward).

4.5 Message Passing

We use Belief Propagation (BP) to estimate the posterior probability distribution of target kinematic states X given observations Y. As discussed in Chapter 3, BP behaves nicely (in terms of its convergence properties) on tree-structured graphs with either only discrete or only continuous Gaussian variables. Running exact BP on graphs with a combination of discrete and continuous random variables, such as graphs constructed for tracking, leads to exponentially growing number of particles in BP messages. General inference problems in such graphs are solved using approximated algorithms, such as Nonparametric Belief Propagation (NBP). As we mentioned in Section 3.4.2, a crucial step in NBP is to sample from a product of d Gaussian mixtures with M modes each. Usually this takes $O(M^d)$ time, but NBP reduces that time to $O(dM^2)$ by using a Gibbs sampler. This is an unnecessary improvement for us if we are using the graphical structure in Figure 4-1, because we never need to multiply more than 2 messages together. In order words, even if we carry out the product explicitly, our complexity is still $O(M^2)$, rendering it unnecessary to use NBP. Moreover, this saves much time needed for sampling to converge. As a result, we decide to construct all products of Gaussian mixtures explicitly.

Below, we discuss doing exact Belief Propagation on the graph in Figure 4-1.

4.5.1 Belief Propagation on Tracking Graph

Message passing can only take three forms: from a continuous target node to another continuous target node, from a discrete assignment node to a continuous target node, and from a continuous target node to a discrete assignment node.

- *Discrete assignment node to continuous target node*

  For messages sent from a discrete assignment node to a continuous target node, we calculate as follows:

$$M_{a\to x}(x_t) = \sum_{a_t} \Psi_a(a_t)\Psi_{a,x}(a_t, x_t) \tag{4.14}$$

This message is basically a sum of Gaussian distributions.

- *Continuous target node to continuous target node*

  For forward messages sent from a continuous target node at time t to the next target node at time $t+1$, we calculate as follows:

$$M_{t\to t+1}(x_{t+1}) = \int \Psi_{t,t+1}(x_t, x_{t+1})\Psi_x(x_t)M_{t-1\to t}(x_t)M_{a_t\to x_t}(x_t)dx_t \qquad (4.15)$$

  With both $M_{t-1\to t}(x_t)$ and $M_{a_t\to x_t}(x_t)$ being Gaussian mixtures, $M_{t\to t+1}(x_{t+1})$ is also a Gaussian mixture. As one can imagine, the number of modes in these target-to-target messages increases multiplicatively from time to time. Note that, for backward messages, the equation is similar (with minor changes of subscripts.)

- *Continuous target node to discrete assignment node*

  We need send messages from target nodes to assignment nodes, when we are calculating the marginal distributions of different data associations (e.g., in the N-scan algorithm, see Chapter 5.)

  For upward messages sent from a continuous target node to a discrete assignment node, we calculate as follows.

$$M_{x\to a}(a_t) = \int \Psi_{a,x}(a_t, x_t)\Psi_x(x_t)M_{t-1\to t}(x_t)M_{t+1\to t}(x_t)dx_t \qquad (4.16)$$

  Because the assignment variable a_t is a discrete variable, this message is a scalar. In implementation, $M_{x\to a}(a_t)$ is represented as a vector, with each entry taking on different values with different values of a_t.

4.5.2 Additions to Conventional Belief Propagation

Conventional BP updates (Equation 3.27) as well as BP updates for tracking graphs (Equations 4.14, 4.15 and 4.16), involve two operations: multiplication and summation (or integration). In graphical models where each node is parameterized by a

single Gaussian distribution, BP messages are single Gaussian distributions as well. To update BP messages, one only needs to compute the means and variances in normal form, or J and h matrices in information form. The normalization constant is usually not treated, because it can be easily computed with the parameters of the Gaussian distribution. In mixture models however, since the constant coefficient for each component of the mixture cannot be readily computed from the parameters of that component, we have to keep track of these constants coefficients when updating messages, one by one. Below, we describe how these two operations are implemented when given a Gaussian mixture distribution.

- *Multiplication*

  This operation is used to aggregate information coming from neighbor nodes. Multiplication of several mixture distributions is straightforward: take one component from each mixture, multiply them together, and sum up these products. Suppose there are n Gaussian mixtures, each of which is represented by a sum of Gaussian distributions in the information form:

$$M^{(i)} = \sum_{j=1}^{n_i} \mathcal{K}(J^{(ij)}, h^{(ij)}, \gamma^{(ij)}) \tag{4.17}$$

  Then, their product is given by:

$$M = \prod_{i=1}^{N} M^{(i)}(x) = \prod_{i=1}^{N} \sum_{j=1}^{n_i} \mathcal{K}^{(ij)}(J^{(ij)}, h^{(ij)}, \gamma^{(ij)}) \tag{4.18}$$

  Note that multiplication in exponential (information) form is equivalent to addition of the multiplicand's parameters (i.e., J and h matrices), and that the result is not a valid probability distribution, as it is not normalized.

- *Sum*

  The sum operation is used to marginalize a certain node on the graph. We need this operation when sending messages from one node to another, or when calculating the marginal distribution of a certain node once message-passing

has converged. The sum operation corresponds to Gaussian Elimination in the parameters of the distribution's information form. Note that, in addition to calculating the J matrix and the h vector as in the conventional single Gaussian case, we add two formulae (Equations 4.25 and 4.29) to calculate the constant coefficients.

In our specific tracking context, we need the following two formulae. The first one is used to marginalize one random variable, given a probability distribution with two (continuous) random variables. In the tracking context, it is used when passing messages forward and backward between target nodes (as in Equation 4.15). Given a mixture distribution of variables x_1 and x_2, we can marginalize out x_2 as follows:

$$
\begin{aligned}
M(x_1) &= \int \sum_{i=1}^{n} M^{(i)}(x_1, x_2) dx_2 && (4.19)\\
&= \int \sum_{i=1}^{n} \mathcal{K}(J^{(i)}, h^{(i)}, \gamma^{(i)}) dx_2 && (4.20)\\
&= \sum_{i=1}^{n} \int \mathcal{K}(J^{(i)}, h^{(i)}, \gamma^{(i)}) dx_2 && (4.21)\\
&= \sum_{i=1}^{n} \int \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} J_{11}^{(i)} & J_{12}^{(i)} \\ J_{21}^{(i)} & J_{22}^{(i)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} h_1^{(i)} \\ h_2^{(i)} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \gamma^{(i)} \right) dx_2 \\
&= \sum_{i=1}^{n} \exp(-\frac{1}{2} x_1^{\mathrm{T}} \hat{J}^{(i)} x_1 + \hat{h}^{(i)} x_1 + \hat{\gamma}^{(i)}) && (4.22)
\end{aligned}
$$

where

$$
\begin{aligned}
\hat{J}^{(i)} &= J_{11}^{(i)} - J_{12}^{(i)} (J_{22}^{(i)})^{-1} J_{21}^{(i)} && (4.23)\\
\hat{h}^{(i)} &= h_1^{(i)} - J_{12}^{(i)} (J_{22}^{(i)})^{-1} h_2^{(i)} && (4.24)\\
\hat{\gamma}^{(i)} &= \gamma^{(i)} - \gamma_0(\hat{J}^{(i)}, \hat{h}^{(i)}) && (4.25)
\end{aligned}
$$

and $\gamma_0(J, h)$ is defined in Equation 3.10.

The second formula is used to marginalize the continuous random variable in

a probability distribution with a continuous random variable and a discrete random variable. In the tracking context, it is used when passing messages upwards, from target nodes to assignment nodes (as in Equation 4.16).

$$M(a) = \int \sum_{i=1}^{n} M^{(i)}(x, a)dx \tag{4.26}$$

$$= \int \sum_{i=1}^{n} \mathcal{K}(J^{(i)}(a), h^{(i)}(a), \gamma^{(i)}(a))dx \tag{4.27}$$

$$= \sum_{i=1}^{n} \int \exp\left(x^{\mathrm{T}} J^{(i)}(a)x + (h^{(i)}(a))^{\mathrm{T}} x + \gamma^{(i)}(a)\right) dx \tag{4.28}$$

$$= \sum_{i=1}^{n} \exp\left(\gamma^{(i)}(a) - \gamma_0(J^{(i)}(a), h^{(i)}(a))\right) \tag{4.29}$$

Now, we have the two equations we need for all convolution operations on our tracking graphs.

4.6 Connections with Multi-Hypothesis Tracking

Although the message-passing algorithm developed in this thesis takes a very different approach to solve the multi-target tracking problem, it has some connections with the MHT algorithm. When message-passing is carried out in an exact fashion, each mode in the messages sent between target nodes at two adjacent time points corresponds to an MHT hypothesis. For the rest of this thesis, we also use the term *particles* to refer to the modes in message.[3] In MHT, each hypothesis at time t is the result of a specific data association assignment between tracks at time $t-1$ and observations at time t. MHT utilizes a hypothesis tree structure to explicitly keep track of the evolution of tracks. In our message-passing algorithm, although it does not explicitly keep track of the evolution of tracks, it is possible to have an additional data field in each particle to keep track of such an evolution.

Moreover, the fact that our algorithm focuses on estimating targets' positions

[3]Just as a clarification, since we are not using particle filtering in this thesis, the term particle here is a bit different from its usual meaning in tracking literatures.

as distributions, rather than as tracks, allows us to use many efficient clustering techniques developed in the machine learning community and gives us the flexibility to accommodate special situations, such as late data arrival and track stitching.

Furthermore, MHT is quite inconsistent in the way that it uses future measurements. To be more specific, on the one hand, it uses future measurements to prune hypothesis at the current time; on the other hand, it never uses future measurements to update, or smooth, previous estimates. This is so because it is hard to conduct backward smoothing on a hypothesis tree. However, given our graphical model structure, backward message-passing can be carried out just as easily as forward message-passing, allowing the use of future measurements to update previous estimates.

Chapter 5

Efficient Multi-Target Tracking Using Graphical Models

5.1 Overview

Multi-Target Tracking (MTT) using graphical models, if carried out in an exact fashion, leads to exponentially growing computational complexity with the duration of the tracking window. In this chapter, we introduce several methods to manage that complexity to a level as low as linear with the duration of the tracking window. These efficient methods allow us to conduct approximate inference on graphs constructed for MTT. Some of these methods are modified from techniques used in conventional Multi-Hypothesis Tracking (MHT) algorithm, including gating and the N-Scan algorithm. Other methods involve the innovative use of clustering algorithm in the context of MTT. Lastly, we present our experimental results, including evidence of linear computational complexity with the duration of the tracking window, and evidence of the message-passing algorithm's ability to handle special situations, such as track stitching and late data arrival.

Though nothing new to the tracking research community, gating and the N-Scan algorithm are powerful techniques to reduce tracking complexity and to improve tracking quality. It is quite straightforward to redefine these techniques in the formalisms of graphical models and message-passing algorithms. The way we define the potential

functions on graphical models is crucial to this, as it allows us to interpret messages sent between two target nodes at adjacent time points as estimates of target states.

Two new hypothesis reduction techniques involve using KD-trees clustering and k-means clustering, respectively. The advantage of clustering methods over traditional pruning methods is that hypotheses eliminated by pruning are lost forever, while clustering never eliminates hypotheses. While reducing the number of hypothesis, clustering methods strive to keep some information about every hypothesis. In fact, these clustering methods do yield better tracking quality over rigid pruning methods.

5.2 Interpretation of Messages Sent Between Target Nodes

Given the joint probability distribution on data associations and target kinematic states in MTT, the definition of potential functions for our graphical models is not unique. We would like to point out that the specific definitions, given in Section 4.4, allow us to interpret forward filtering messages as estimates of future states based on all previous observations and backward smoothing messages as estimates of previous states based on all future observations.[1] Mathematically, in the case of filtering, we have:

$$M_{t\to t+1}(x_{t+1}) = \sum_i \alpha_i \mathcal{N}(\mu_i, \Sigma_i) = P(x_{t+1}|y_{1:t}) \tag{5.1}$$

where each hypothesis is represented by a Gaussian distribution with parameters (μ_i, Σ_i), and has a weight α_i. The sum of all weights in such a message should add up to 1. Similarly, in the case of smoothing, we have:

$$M_{t\to t-1}(x_{t-1}) = \sum_i \alpha_i \mathcal{N}(\mu_i, \Sigma_i) = P(x_{t-1}|y_{t:T}) \tag{5.2}$$

where T is the total time duration of the tracking window.

[1] This is generally true for chain-structured graphs, as long as the self-potential of target nodes is 1 and the edge potential is the conditional probability between two nodes.

5.3 Gating

Gating is a standard technique used in the Multi-Hypothesis Tracking (MHT) algorithm to limit the number of data association hypotheses being generated (see Section 2.3.3). Similar ideas can be applied in MTT based on message-passing. Here, gating is done when messages from assignment nodes, or *assignment messages*, are multiplied with messages from target nodes, or *target messages* (as in Equation 4.15). Without gating, a particle (i.e., a mode in the Gaussian mixture) in the target message is multiplied with a particle in assignment message, and this is done between every pair of particles between target message and assignment message. With gating, rather than multiplying a particle in the target message with every single particle from the assignment message, it is only multiplied with the ones in the assignment message with data associations that are consistent with the gating constraints of this particular particle in the target message. The gating regions can be determined by the means and the covariances in target messages, because these messages can be interpreted as estimates of target kinematic states (as in Equation 5.2).

Algorithm 5.1: Gating for MTT Based on Graphical Models

Gating is done when messages from assignment nodes, or *assignment messages*, are multiplied with messages from target nodes, or *target messages* (as in Equation 4.15).

1. Compute the gating region for each particle in $M_{t-1\to t}(x_t)$.
2. Multiply each particle in target message $M_{t-1\to t}(x_t)$ with particles in $M_{a\to x}(x_t)$ corresponding to data associations that are consistent with its gating constraint.
3. Sum up the result as the product of $M_{t-1\to t}(x_t)M_{a\to x}(x_t)$.

5.4 N-Scan Algorithm

In Multi-Hypothesis Tracking (MHT), the N-Scan algorithm defers difficult hypothesis pruning decisions until more data are received. We apply the same idea here. See Algorithm 5.2 for details and see Figure 5-1 for demonstrations.

Algorithm 5.2: N-Scan for MTT Based on Graphical Models

Assume the tracker is running in steady state. Upon receipt of new observations at time t, do the following.

1. Compute downward assignment message at time t: $M_{a\to x}(x_t)$.
2. Compute backward smoothing messages $M_{t\to t-1}(x_{t-1})$, $M_{t-1\to t-2}(x_{t-2})$, ..., $M_{t-N+2\to t-N+1}(x_{t-N+1})$ subsequently.
3. Compute upward message at time $t-N+1$: $M_{x\to a}(a_t)$ and compute marginals at assignment node a_{t-N+1}: $p(a_{t-N+1})$.
4. According to the marginal distribution calculated in last step, eliminate unlikely data associations at time $t-N+1$.
5. With the reduced data association hypothesis space at time $t-N+1$, propagation messages down, and then forward: calculate $M_{a\to x}(x_{t-N+1})$, $M_{t-N+1\to t-N+2}(x_{t-N+2})$, $M_{t-N+2\to t-N+3}(x_{t-N+3})$, ..., $M_{t\to t+1}(x_{t+1})$.
6. Obtain prediction $p(x_{t+1}|y_{1\ldots t}) = M_{t\to t+1}(x_{t+1})$
7. Repeat for the next time point.

5.5 Hypothesis Reduction

In order to prevent the exponential explosion of number of hypothesis over time, we need to either eliminate some hypothesis or approximate the original Gaussian

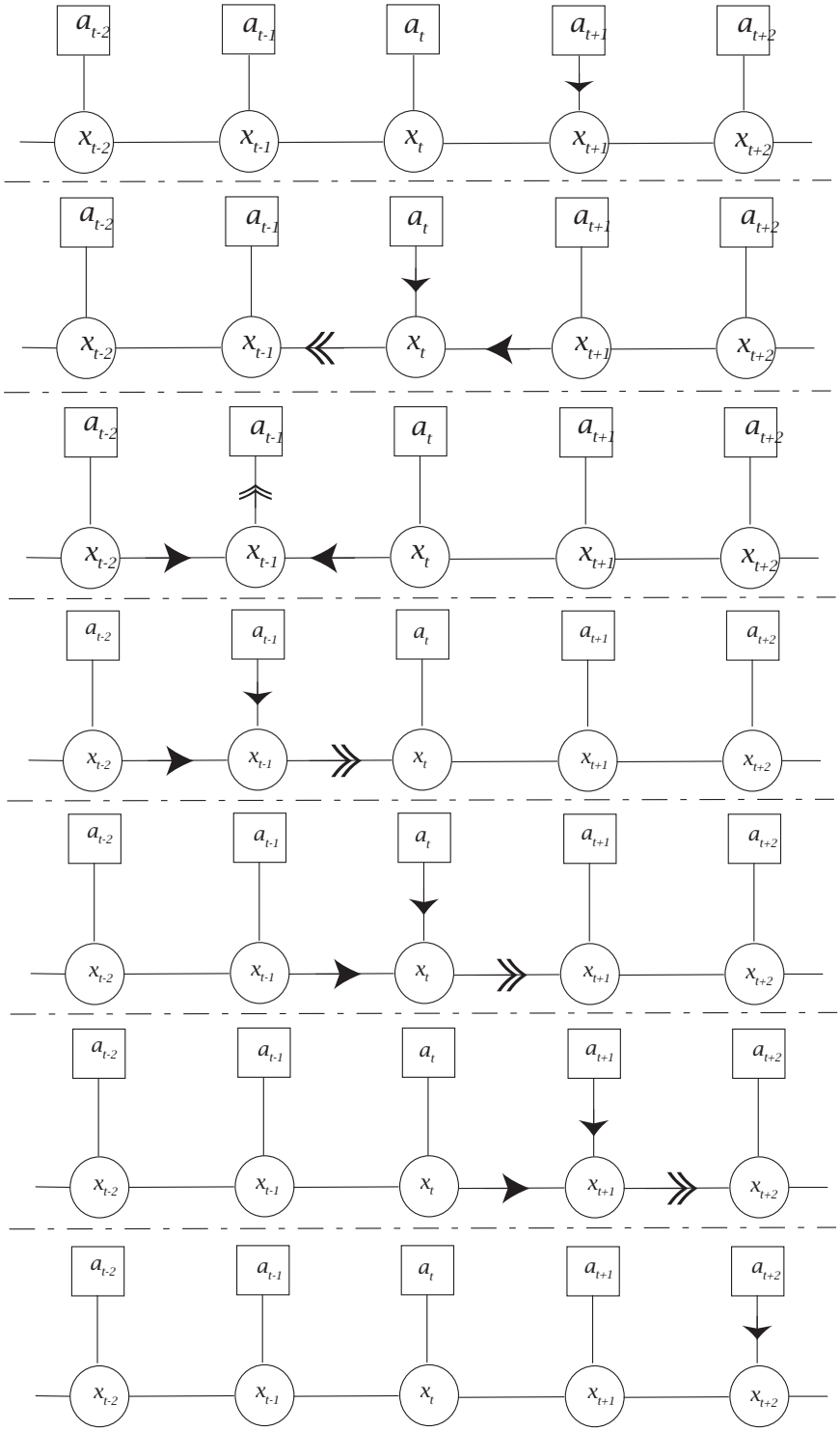


Figure 5-1: Demonstration of N-Scan Algorithm when N=3. From top to bottom, upon receipt of new observation, the algorithm sequentially propagates backwards, eliminates unlikely data associations at earlier time points, and finally propagates forward. This is repeated recursively upon receipt of new data at next time point.

mixture distribution with another one with fewer modes. Here, we present three different approaches to do so: Maximum Probability Mass, KD-trees clustering, and k-means clustering. In the discussion below, we are given an input Gaussian mixture distribution, and we are asked to compute an output Gaussian mixture distribution. On the premise that we preserve as much information as possible, we strive to find an efficient way to eliminate excessive particles in messages being passed on the graph.

5.5.1 Maximum Probability Mass

A natural way to eliminate some of the hypothesis is simply to throw away those unlikely ones, or the ones with smaller weights. There are three alternatives in doing so. In the first alternative, we can eliminate hypothesis whose weights are below certain threshold determined by the leading (heaviest) hypothesis. For example, if the hypothesis with the largest weight is 0.6, by setting the threshold to be 1/10 of the leading weight, hypothesis with weights that are less than 0.06 are eliminated. Although this method does not provide any bound on either the quality of the approximated messages nor the number of particles remaining, it tends to keep more particles when the situation is ambiguous and to keep fewer when unambiguous.

In the second alternative, we can specify the number of particles we want to keep and the algorithm keeps the hypothesis with heaviest weights. This corresponds to the so-call m-best method. By fixing the number of particles in messages, this method gives a worst-case bound on the running time.

The third alternative is an adaptive approach, in which we eliminate low likelihood modes to the point such that the sum of the probability mass of the remaining modes is more than a certain threshold. This method provides some guarantees on the quality of the processed messages, but unfortunately may lead to out-of-control running time in extremely ambiguous situations, in which there are many particles with small, roughly equal weights.

Algorithm 5.3: Particle reduction using Maximum Probability Mass

Given $P = \sum_{i=1}^{m} \alpha_i \mathcal{N}(\mu_i, P_i)$ where particles are sorted by their weights (i.e., $\alpha_j > \alpha_k$ iff $j < k$), there are three alternatives to pick particles with larger weights.

1. Eliminate particle i if $\alpha_i < k\alpha_1$ with k specified by user.
2. Eliminate particle with index $i > n$ with n specified by user.
3. Eliminate particle with index $i > j$ where j is smallest index such that $\sum_{k=1}^{j} \alpha_k \geq t$ with t specified by user

5.5.2 Adaptive KD-Tree Clustering

We propose using KD-trees to conduct hypothesis reduction, based on the following two motivations. First, although the Maximum Probability Mass approach provides a conceptually-clear approach to reduce the number of particles, it can not recover information in particles that it eliminates. Information in eliminated hypothesis are permanently lost. Second, the process of constructing KD-trees implicitly clusters input data points, because the closer the two points are in the same KD-tree the closer they are in the K-dimensional space. Although it is not the most precise clustering method, KD-tree provides an extremely efficient way to separate data points coarsely into different clusters.

We take a walk down the KD-tree starting from the root node. At each node, we calculate the KL-divergence between the two children nodes, and we stop at that node if the symmetrized KL-divergence between its children is smaller than a threshold specified by the user. We keep all the nodes at which this procedure stops. Effectively, we are making a multi-resolution cut through the tree (see Figure 5-2). The smaller the threshold is, the higher quality the output is. Because the number of particles in the output varies with the similarity between different components in the input mixture, this algorithm *adaptively* reduces the number of hypotheses.

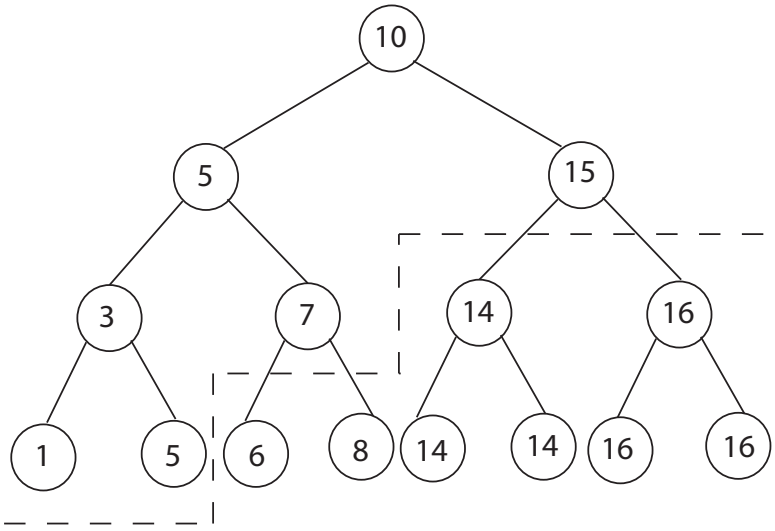


Figure 5-2: Our algorithm effectively makes a multi-resolution cut through the tree. In this one-dimensional example, we use the Euclidean distance rather than the KL-divergence. The stopping threshold is set to be 2; that is, if the distance between a node's two children is less than or equal to 2, then the algorithm cuts off the whole subtree under this node.

Because the KD-tree is a multi-scale data structure, aggregate information in child nodes are stored in parent nodes. Even though some children nodes are eliminated, their information is still represented by their parents.

Algorithm 5.4: Adaptive particle reduction using KD-trees

Given $P = \sum_{i=1}^{m} \alpha_i \mathcal{N}(\mu_i, P_i)$, we organize all particles in a KD-tree. Take a walk down the tree from the root and do the following.

1. At node i with children l_i and r_i, calculate symmetrized KL-divergence $D_{SKL}(l_i, r_i)$ using Equations 3.35 and 3.36
2. If $D_{KL}(l_i, r_i) \leq t$ with t specified by user, then eliminate the whole subtree under node i (keeping i). Otherwise, walk down one level and repeat step 1 and 2 recursively for both l_i and r_i.
3. When finished, output all leaf nodes.

5.5.3 k-means Clustering

Clustering has also been proposed in traditional MHT [5], where tracks that share at least one observations are clustered together. In doing so, MHT reduces a global data association problem into a localized problem. In our context of MTT using graphical models, there is no concept of "a track", because Belief Propagation does not keep track of which particle in the previous time point corresponds to a certain particle in the current time point. Therefore, clustering in graphical-model-based tracking takes a totally different meaning than its counterpart in MHT.

Here, we present a clustering algorithm modified from the conventional k-means algorithm (Section 3.5.2).[2] The key idea to modify the conventional k-means algorithm so that it can be used to cluster Gaussian mixtures is to replace the Euclidean distance with KL-divergence. In k means clustering, the first step is to construct a new partition by associating each point with the nearest centroid, which can be easily calculated with the definition of KL-divergence, according to Equation 3.36. The second step (calculating the centroid) is not trivial. The challenge is: given a Gaussian mixture distribution, how to define and to search for a centroid distribution that is a single Gaussian distribution? We define such a centroid distribution as one that minimizes the KL-divergence between the centroid distribution and the original mixture distribution. If we denote the original mixture as $p(x)$ and the centroid distribution as $q(x)$, then,

$$q(x) = \underset{q(x)\in\text{all Gaussian distribution}}{\operatorname{argmin}} D_{\mathrm{KL}}(p(x)\|q(x)) \tag{5.3}$$

Note that since KL-divergence is an *asymmetric* measure of dissimilarity of two probability distributions, $D_{\mathrm{KL}}(p(x)\|q(x))$ is not equal to $D_{\mathrm{KL}}(q(x)\|p(x))$.

It turns out that $q(x)$ can be readily calculated from $p(x)$, as presented in the following theorem.

Theorem 1. *Given a Gaussian mixture distribution* $p(x) = \sum_{i=1}^{M} \alpha_i \mathcal{N}(\mu_i, \Sigma_i)$, *the*

[2] In fact, with the message-passing structure, we can take advantages of many other modern clustering techniques developed in the statistical machine learning, for instance, the EM algorithm and Affinity Propagation. However, in this thesis, we only consider the k-means clustering.

parameters of the centroid $q(x)$, defined as a single Gaussian distribution that has the smallest KL-divergence with $p(x)$ (i.e., $D_{KL}(p(x)\|q(x))$), is given by the weighted average of all modes in $p(x)$. In other words, the mean and the covariance matrix of $q(x)$ is given by:[3]

$$\hat{\mu} = \sum_{i=1}^{M} \alpha_i \mu_i \tag{5.4}$$

$$\hat{\Sigma} = \sum_{i=1}^{M} \alpha_i (\mu_i \mu_i^T + \Sigma_i) - \hat{\mu}\hat{\mu}^T \tag{5.5}$$

Proof. Given $p(x) = \sum_{i=1}^{M} \alpha_i \mathcal{N}(\mu_i, \Sigma_i)$ and $q(x) = \mathcal{N}(\hat{\mu}, \hat{\Sigma})$, their KL-divergence is given by

$$\begin{aligned} D_{\mathrm{KL}}(p(x)\|q(x)) &= \int p(x) \log \frac{p(x)}{q(x)} dx \\ &= \int p(x) \left(\log p(x) + \frac{1}{2}(x-\hat{\mu})^T \hat{\Sigma}^{-1}(x-\hat{\mu}) + \frac{1}{2} n \log 2\pi + \frac{1}{2} \log \det \hat{\Sigma} \right) dx \end{aligned}$$

By setting its derivative against $\hat{\mu}$ to zero, we have[4]:

$$\frac{\partial D_{\mathrm{KL}}(p(x)\|q(x))}{\partial \hat{\mu}} = 0 \quad = \quad -\int p(x) \hat{\Sigma}^{-1}(x-\hat{\mu}) dx \tag{5.6}$$

$$= \hat{\Sigma}^{-1} \left(\hat{\mu} - \int p(x) x dx \right) \tag{5.7}$$

Thus,

$$\hat{\mu} = \int p(x) x dx = \sum_{i=1}^{M} \alpha_i \mu_i \tag{5.8}$$

Similarly, by setting its derivative against $\hat{\Sigma}$ to zero, we have[5]:

$$\begin{aligned} \frac{\partial D_{\mathrm{KL}}(p(x)\|q(x))}{\partial \hat{\Sigma}} = 0 \quad &= \quad \int p(x) \left(-\frac{1}{2} \hat{\Sigma}^{-1}(x-\hat{\mu})(x-\hat{\mu})^T \hat{\Sigma}^{-1} + \frac{1}{2} \hat{\Sigma}^{-1} \right) dx \\ &= -\frac{1}{2} \hat{\Sigma}^{-1} \mathrm{E}_{p(x)} \left[(x-\hat{\mu})(x-\hat{\mu})^T \right] \hat{\Sigma}^{-1} + \frac{1}{2} \hat{\Sigma}^{-1} \end{aligned} \tag{5.9}$$

[3] In the MTT context, approximating several Gaussian distributions with one single Gaussian distribution is similar to the Probability Data Association (PDA) approach. See Section 2.2.2

[4] We make use of the matrix identity $\frac{\partial}{\partial s}(x-s)^T W (x-s) = -2W(x-s)$, as long as W is symmetric.

[5] We make use of the matrix identities $\frac{\partial a^T X^{-1} b}{\partial X} = -X^{-T} a b^T X^{-T}$ and $\frac{\partial \log |\det(X)|}{\partial X} = (X^{-1})^T$

Thus,

$$\hat{\Sigma} = \mathrm{E}_{p(x)}\left[(x-\hat{\mu})(x-\hat{\mu})^T\right] = \sum_{i=1}^{M} \alpha_i(\mu_i\mu_i^T + \Sigma_i) - \hat{\mu}\hat{\mu}^T \tag{5.10}$$

Hence the proof is complete. □

Algorithm 5.5: k-means Clustering for Gaussian Mixtures

Given $P = \sum_{i=1}^{m} \mathcal{K}(J_i, h_i, \gamma_i)$, and $n \leq m$, approximate P with $Q = \sum_{j=1}^{n} \mathcal{K}(J_j, h_j, \gamma_j)$

1. Initialization: randomly assign each mode in the original mixture to one of n clusters.
2. Calculate the centroid for each cluster j using Theorem 1.
3. Calculate symmtrised KL-divergence between every pair of modes in P and Q, $D_{ij}(P_i \| Q_j)$ using Equation 3.36.
4. Assign mode i in P to cluster with index $= \underset{j}{\operatorname{argmin}}\, D_{ij}$.
5. Iterate steps 1 thru 4 until convergence. If algorithm fails to converge then initialize again and try again. Typically to avoid being trapped in local minimum, run algorithm multiple times with different initialization parameters.

5.5.4 Discussion

In terms of running time, the Maximum Probability Mass approach is the fastest, KD-tree clustering is the second fastest, and k-means clustering is the slowest. In terms of approximation quality, on a conceptual level, KD-tree clustering and k-means clustering method have the advantage that even though the number of particles is reduced, the original information is better preserved.[6] In contrast, in the maximum probability mass method (as well as traditional N-Scan method), when a hypothesis is eliminated,

[6]We here only offer a conceptual comparison, because the performance of a certain method depends on the particular scenario. For this reason, it is difficult to come up with a fair basis on which to compare all these methods.

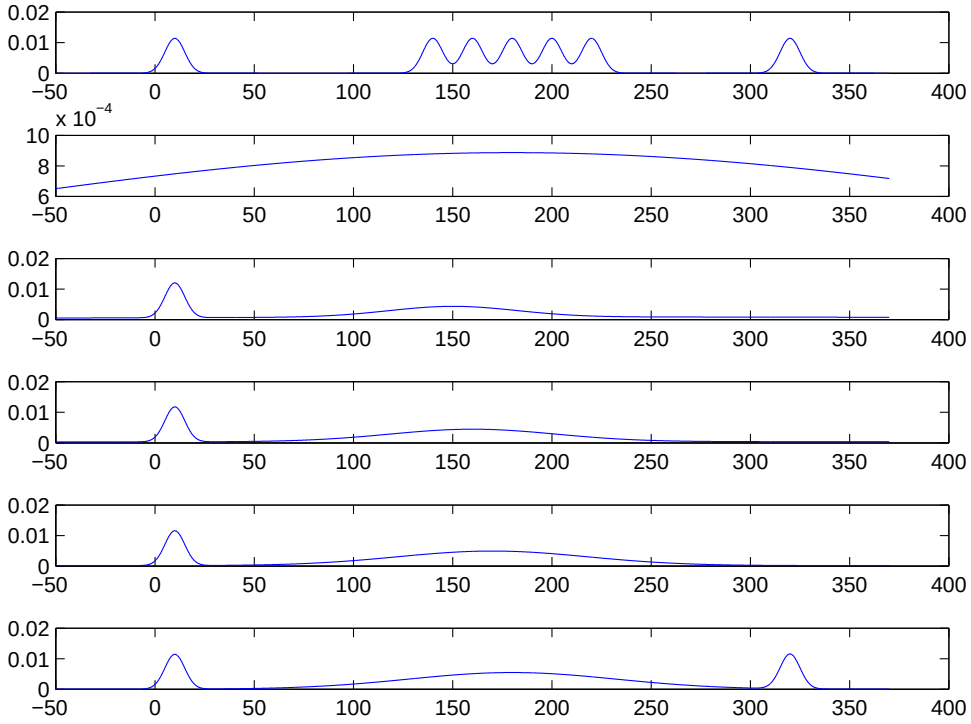


Figure 5-3: The modified k-means clustering algorithm approximates a Gaussian mixture distribution with 7 modes, using 3 modes. The procedure converges in 5 iterations. The first plot is the original mixture, and the ones below are outputs of the k-means clustering algorithm at each iteration.

information in the hypothesis is permanently lost. Therefore, we believe the KD-tree method offers the best tradeoff between running time and tracking accuracy among these three methods.

5.6 Experimental Results

In this section, we present some experimental results of using our message passing algorithms to conduct MTT. In 5.6.1, we present the setup and assumptions under which we operate our experiments. In 5.6.2, we provide the evidence to our claim that the complexity of our message passing algorithm is roughly linear in the duration of the tracking window. In 5.6.3, we show that the graphical model structure allows

us to handle special situations, such as track stitching and late data arrival, without difficulty.

5.6.1 Basic Setup

In the basic setup of our experiments, multiple targets move in a 2-D surveillance area. The number of targets is known a-priori. Thus, we do not consider track initialization and termination here. The movement of each target is modeled by the following state-space model:

$$x_{t+1} = Fx_t + w_t \tag{5.11}$$

where x_t is the 2-D kinetic state vector, including position, velocity and acceleration, namely,

$$x_t = \begin{pmatrix} x & \dot{x} & \ddot{x} & y & \dot{y} & \ddot{y} \end{pmatrix}^T \tag{5.12}$$

The transition matrix F is given by

$$F = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \epsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \epsilon \end{pmatrix} \tag{5.13}$$

where ϵ is inserted to avoid matrix singularity, and can be interpreted as the correlation in acceleration from time to time In our experiment, we set $\epsilon = 0.001$. It is worthwhile to point out that Equation 5.11 is the single target version of Equation 4.1, which describes the state transitions of all targets. If there are M targets, then the A matrix is a block diagonal matrix, consisting of M F matrices.

The noise w_t is drawn from distribution $\mathcal{N}(0, \Lambda)$. The value of Λ regulates how erratically targets travel. In discrete time, because noise accumulates on all components of the state vector, Λ should be invertible, which happens to help us keep our

equations simple. Because we want the noise to mainly drive the acceleration, we define the Λ matrix as follows:

$$\Lambda = \begin{pmatrix} .001 & 0 & 0 & 0 & 0 & 0 \\ 0 & .001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & .001 & 0 & 0 \\ 0 & 0 & 0 & 0 & .001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.14}$$

There are three (3) types of sensors monitoring the surveillance area. The first and the second types are bearing-only sensors that are located very far from the surveillance such that they only give one-dimensional linear measurements of the full kinetic vector. We choose to locate sensors far away from the surveillance area to preserve the linearity in the tracking problem. The third type sensors give 2-D, near-field measurements of positions only. In order words, we have the following three types of sensors:

- Type I: measures $x, \dot{x}$ only.
- Type II: measures $y, \dot{y}$ only.
- Type III: measures x, y only.

In addition to Type I and II sensors, Type III sensors help resolve some ambiguity in data associations. If we want to make the scenario more difficult for the tracker, we can remove Type III sensors. All observations are generated according to the following observation model:

$$y_t = Hx_t + v_t \tag{5.15}$$

where the observation noise is modeled by $\mathcal{N}(0, .1I)$. Since different types of sensors observe different components of the kinematic state vector, the dimensions of the H matrix and the noise covariance change with the type of the sensor. Equation 5.15 does not contain the data assignment variable as in Equation 4.2. When sensors

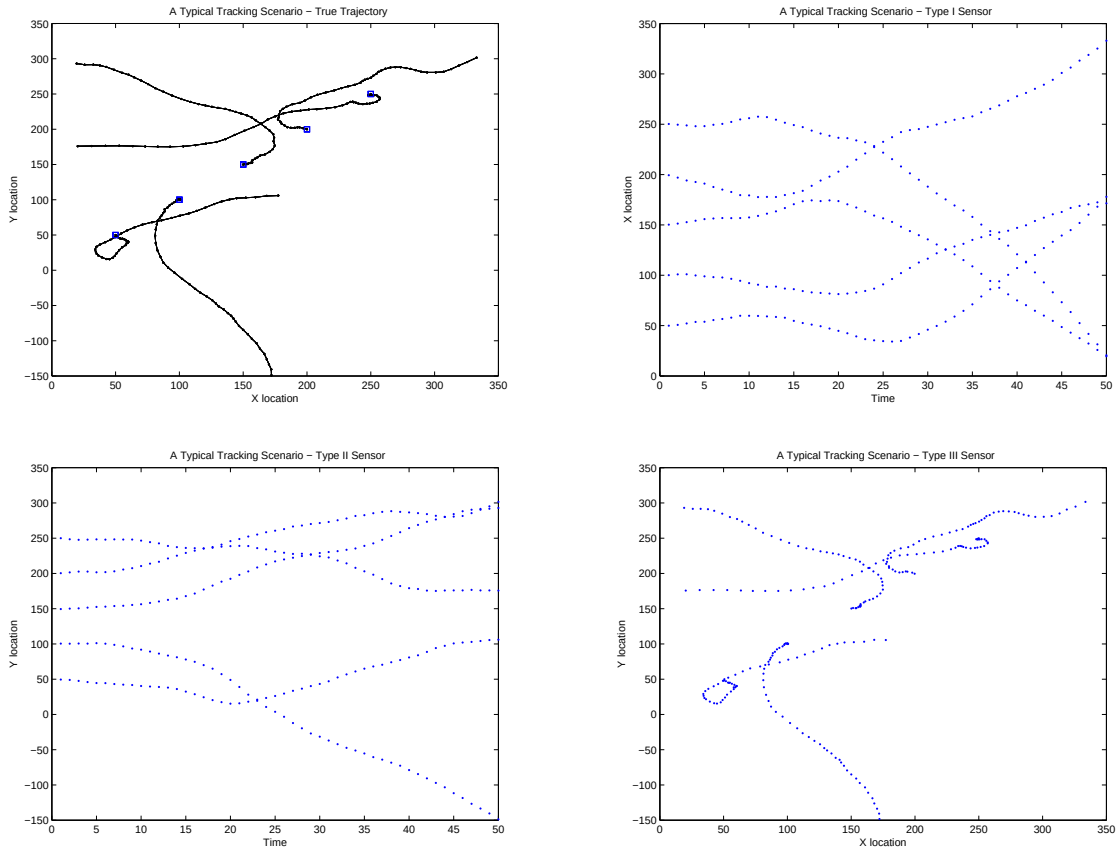


Figure 5-4: The true trajectory and observations from all three types of sensors of a typical tracking scenario with 5 targets and 50 time frames. Although Type I and II sensors also observe the velocity as well, we here are not showing the velocity observations.

receive observations, they do not know the correspondence between observations and targets. Also, in our experiments, we included false alarms and missed detections, the probabilities of which are set to be 0.05.

Targets' initial kinematic states in our experiments are generated randomly, and subsequent states are generated according to the dynamic model and observation model given above. We used up to 10 targets, 3 sensors (one from each type), and the time horizon spans from 20 frames to 100 frames. A typical tracking scenario is given in Figure 5-4.

When we show the results of our tracking algorithm in Section 5.6.3, we are showing the particle representation of the estimated target positions. For each target

(differentiated by a unique color), there is a set of particles with different weights (indicated by the density of the color). If there is a predominant track of particles (i.e., less ambiguity), then only that track is shown while others are invisible on the plots. If there are multiple sets of particles with high weights (i.e., more ambiguity), then more than one estimated track for one or more targets is shown on the plots (e.g., ghost).

5.6.2 Complexity of N-Scan Algorithm

If we can limit the number of particles in messages passed between different time points, then the complexity of our N-Scan message-passing algorithm would be linear with respect to N, or the duration of the tracking window. The first level to implement such a limit is to rigidly put a cap on the number of particles to keep. The second and more advanced level is to somehow adaptively reduce the number of particles while maintaining tracking quality. Here, we show the evidence that, by using adaptive KD-tree clustering as the hypothesis reduction method, while maintaining acceptable tracking accuracy, the message-passing algorithm achieves linear complexity with respect to the duration of the tracking window.

In Figure 5-5, we show some sample tracking results (N=5). These are a subset of the 100 scenarios we used. In all of them, there are 5 targets, 3 sensors (one from each type), and the duration is 50 time frames. Also, in our experiments, we included false alarms and missed detections, the probabilities of which are set to be 0.05. We show this to establish that message-passing MTT algorithm, using KD-tree clustering as the hypothesis reduction method, produces acceptable tracking accuracy.

In Figures 5-6 and 5-7, we show the relationship between running time and the duration of the tracking window (N). Experiments in Figure 5-6 are done using Type I and II sensors only, while those in Figure 5-7 are done using sensors of all three types. It turns out that, although the running time of the message-passing N-Scan MTT algorithm is linear with respect to N for each tracking scenario, as shown in Figures 5-6(a) and 5-7(a), different scenarios have different levels of stress (difficulty).

	Mean	Standard Deviation
Using Type I and II sensors only	69.5	12.2
Using sensors of all three types	52.1	4.9

Table 5.1: Statistics for K_i across 100 scenarios

We can further model such a linear running time relationship with N as

$$T_i = K_i N \tag{5.16}$$

where i is the index of the scenario. The more ambiguous the tracking scenario is, the larger the value of K_i is. We estimated the mean and variance of the slopes in Equation 5.16, as shown in Table 5.1. Due to such a variation in the slope, the standard deviations of running time across different scenarios increase with the value of N, as shown by the error bars in Figures 5-6(b) and 5-7(b). A notable observation is that the running time increases much faster in experiments in which the Type III sensor is not used, because the tracing situation becomes much more ambiguous without Type III sensors (Table 5.1).

5.6.3 Special Situations

As mentioned in Section 2.3.4, the conventional MHT algorithm, with its hypothesis tree structure, does not handle well special situations, such as late data arrival and track stitching. Here, we provide two examples in which we show that our algorithm based on graphical models is capable and quite natural to handle these situations.

In Figure 5-8, we compare tracking results between a message-passing N-Scan algorithm with $N = 3$ and a N-Scan with $N = 15$, in a scenario in which observations from $t = 8$ to $t = 15$ arrive late at $t = 19$. If the tracking window is small ($N = 3$ as in Figure 5-8(a)), then when late data arrive, the tracker is not able to incorporate those late data as the tracking window has already moved passed the range with late data. As a result, the tracker mistakenly confuses the two targets, each of which is marked by a different color. If the tracking window is long enough ($N = 15$ as in Figure 5-8(b)), then to incorporate the late data when they arrive, the tracker just needs

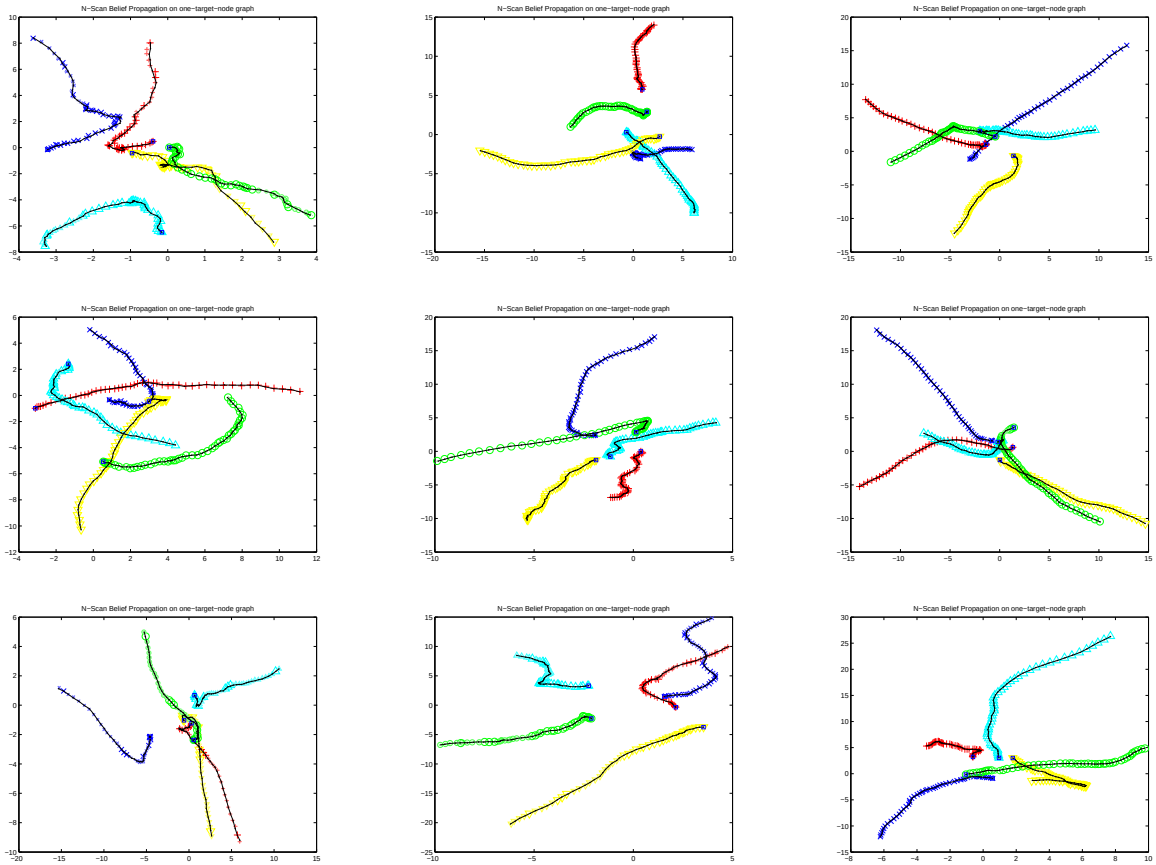
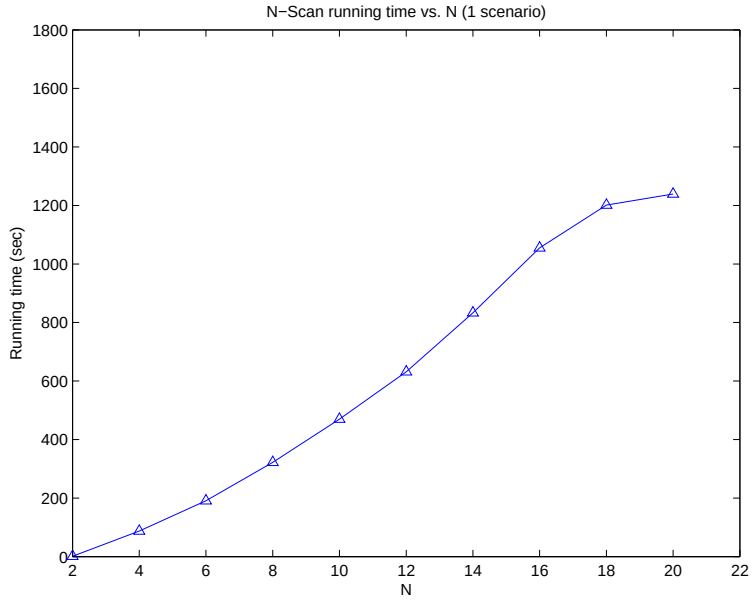


Figure 5-5: We show some sample tracking results when N=5. These are a subset of the 100 scenarios we used. In all of them, there are 5 targets, 3 sensors (one from each type), and the duration is 50 time frames. Also, the tracker has built in the capability to account for false alarms and missed detections, the probabilities of which are set to .05. We show this to establish that message-passing MTT algorithm, using KD-tree clustering as the hypothesis reduction method, produces acceptable tracking accuracy.

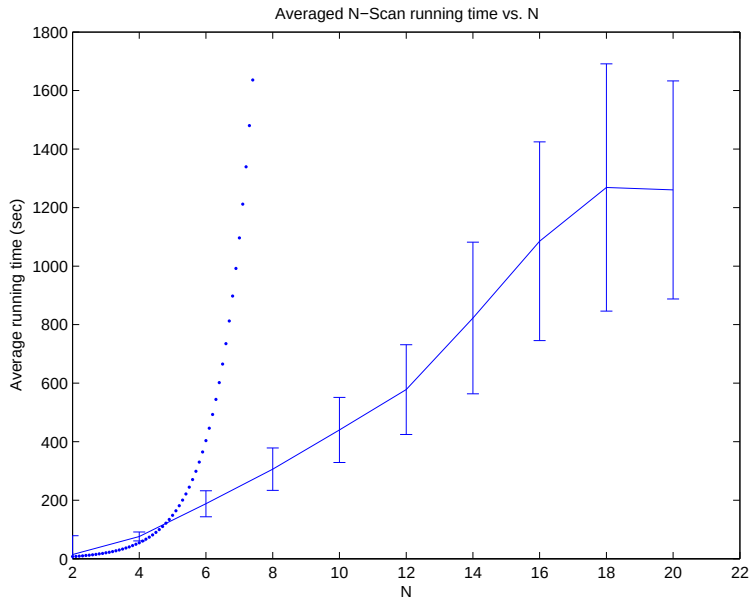
to conduct a regular backward-forward message-passing within its tracking window. As showed in Figure 5-8(a), the tracker now correctly tracks the two targets. In this scenario, there 3 sensors (one from each type), and the duration is 50 time frames. Target movements are simulated with noise covariance $Q = 0.001I_{6\times 6}$. Also, we included false alarms and missed detections, the probabilities of which are set to .05.

In Figure 5-9, we show an example of track-stitching, using a message-passing N-Scan algorithm with $N = 30$. In this scenario with 50 time frames, observations are missing for time points from $t = 5$ to $t = 25$. Each target is marked by an unique

color. Because we only use one Type I sensor and one Type II sensors, there would be two ghost tracks, although they may have smaller weights (likelihoods). We initialize the tracker with relatively strong certainty of where the true targets are such that ghost tracks do not appear before the window of missing data. However, after the window, without track stitching, the tracker cannot distinguish between real tracks and ghost tracks, hence the two ghost tracks in Figure 5-9(a). If we make available some extra information to make clear the correspondence between tracks before and after missing data, then the tracker can easily "stitch" the tracks together (Figure 5-9(b), because the window length $N = 30$ spans across the period of missing data. Also, we included false alarms and missed detections, the probabilities of which are set to .05.

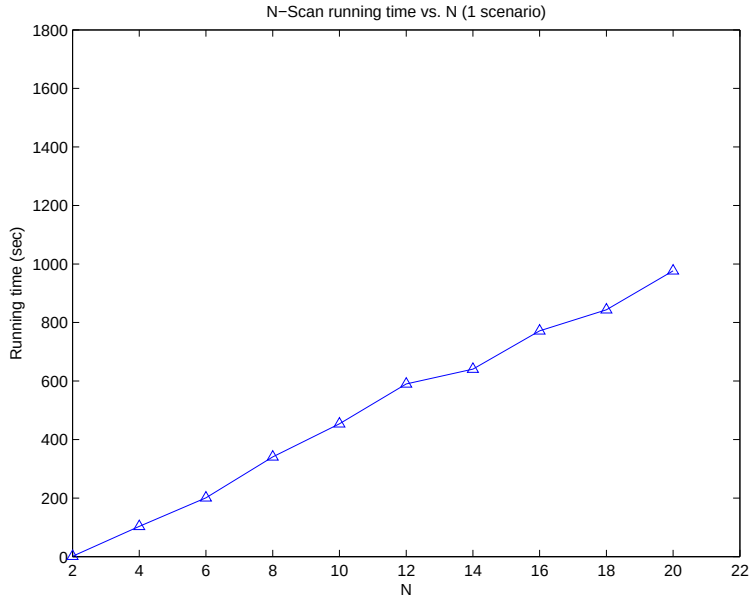


(a) Running time as a function of N, for a particular tracking scenario, using only Type I and II sensors

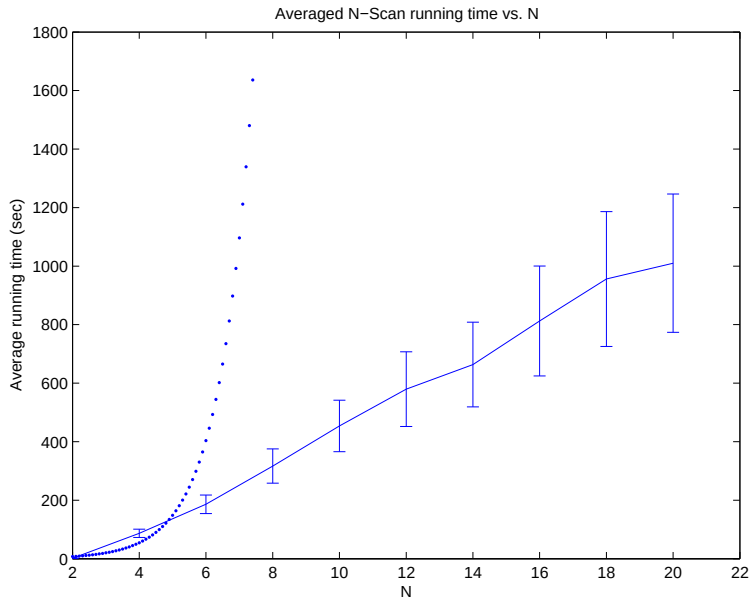


(b) Averaged running time as a function of N for 100 scenarios, using only Type I and II sensors

Figure 5-6: Using Type I and II sensors only, we show the relationship between running time and N. This is done for scenarios in which there are 5 targets, and 50 time frames. In (b), the error bars indicate the standard deviations of the running times at a particular N across different scenarios. In order to highlight the difference between linear complexity and exponential complexity, we also plot the corresponding relationship of a hypothetical MHT tracker.

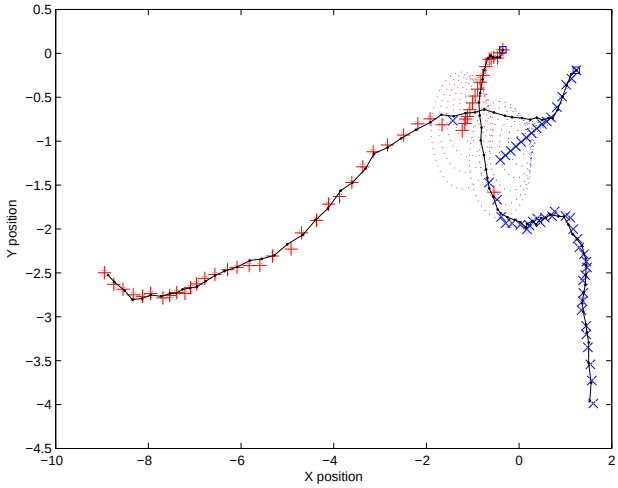


(a) Running time relationship with N for a particular tracking scenario, using sensors of all three types

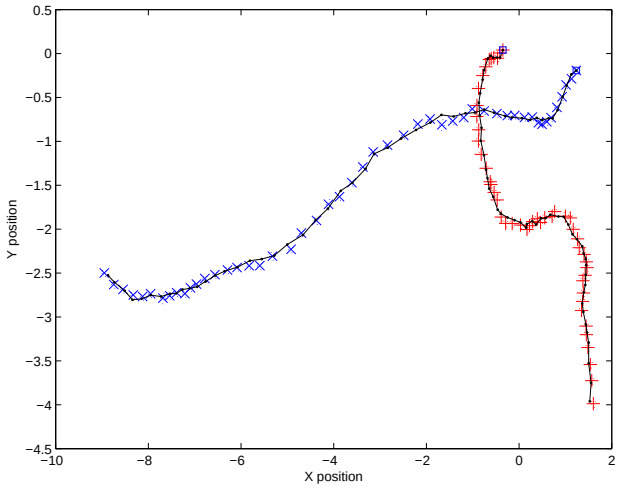


(b) Averaged running time relationship with N for 100 scenarios, using sensors of all three types

Figure 5-7: Using sensors of all three types, we show the relationship between running time and N. This is done for scenarios in which there are 5 targets, and 50 time frames. In (b), the error bars indicate the standard deviations of the running times at a particular N across different scenarios. In order to highlight the difference between linear complexity and exponential complexity, we also plot the corresponding relationship of a hypothetical MHT tracker.

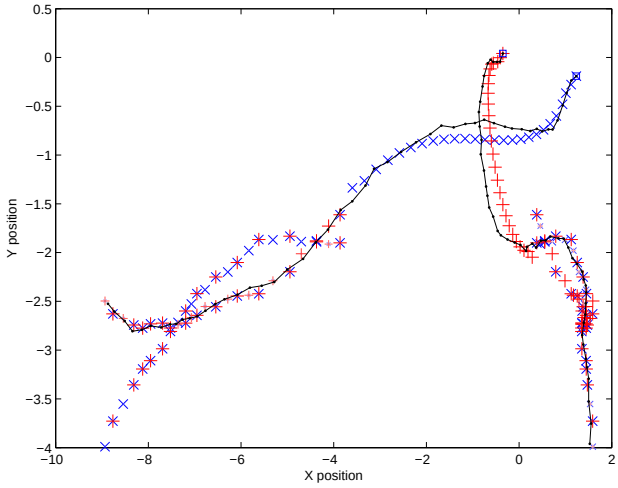


(a) 3-Scan tracking with late data from $t = 8$ to $t = 15$ arriving at $t = 19$

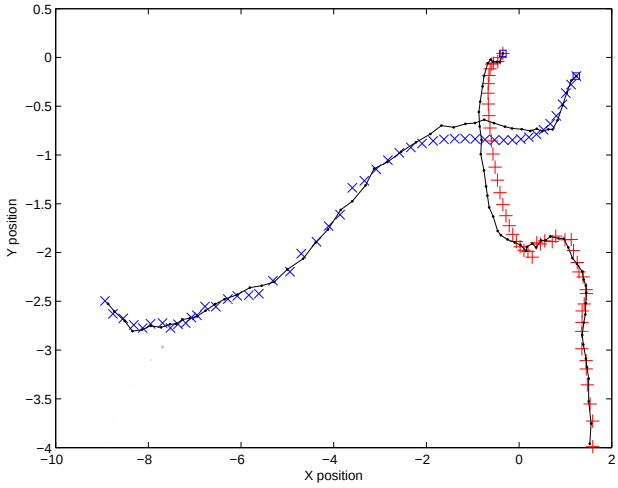


(b) 20-Scan tracking with late data from $t = 8$ to $t = 15$ arriving at $t = 19$

Figure 5-8: We compare tracking results between a message-passing N-Scan algorithm with $N = 3$ and another with $N = 15$, in a scenario where observations from $t = 8$ to $t = 15$ arrive late at $t = 20$. Each target is marked by an unique color. With the shorter window, the tracker is not able to make use of the late data, and thus it confuses the two targets. With a much longer window, the tracker is able to incorporate late data within its tracking window and thus gives correct tracking results. In this scenario, there 3 sensors (one from each type), and the duration is 50 time frames. Also, we included false alarms and missed detections, the probabilities of which are set to .05.



(a) 3-Scan tracking without track-stitching



(b) 30-Scan tracking with track-stitching

Figure 5-9: In this scenario with 50 time frames, observations are missing from $t = 5$ to $t = 25$. Each target is marked by an unique color. Because we only use one Type I sensor and one Type II sensor, there would be two ghost tracks. We initialize the tracker with relative strong certainty of where the true targets are, such that ghost tracks do not appear before the window of missing data. However, after the window, without track stitching, the tracker cannot distinguish between real tracks and ghost tracks, hence the two ghost tracks in Figure 5-9(a). If we make available some extra information to make clear the correspondence between tracks before and after missing data, then the tracker can easily "stitch" the tracks together (Figure 5-9(b), because the window length $N = 30$ spans across the period of missing data. Also, we included false alarms and missed detections, the probabilities of which are set to .05.

Chapter 6

Conclusion and Recommendation

In this thesis, we present a framework to use graphical models and message passing algorithms to solve the Multi-Target Tracking (MTT) problem. In this chapter, we summarize the major contributions of this thesis and suggest several ideas for further research.

6.1 Contribution

6.1.1 Use of The Graphical Model Formalism

We cast the MTT problem into the graphical model formalism and show that the graphical model structure offers enormous flexibility to overcome several limitations faced by existing MTT algorithms. The graphical structures constructed for MTT are capable of handling basic events, such as false alarms and missed detections, and special events, such as out-of-sequence observations and track stitching.

6.1.2 Efficient Message-Passing Algorithm for MTT

Existing exact inference algorithms, when applied to solve the tracking problem, lead to exponential complexity in the duration of the tracking window, because there is a mix of both continuous and discrete random variables on the graph. Thus, we develop several efficient algorithms to do approximate tracking on MTT graphs. In

particular, one hypothesis reduction method that is based on KD-trees can adaptively manage the size of messages passed on the graph. Using this method, we show that the complexity can be made to be as low as linear with respect to the duration of the tracking window.

6.2 Suggestion for Further Research

Target tracking is an area of special interests to the defense and surveillance industry. Despite its long history, MTT remains a very challenging problem. Not to mention the many technical subtleties required to turn theory into practical systems, there are still many theoretical questions to address. This is even more-so with our new graphical model framework for MTT. In this section, we suggest two ideas for further theoretical research in this area.

6.2.1 Nonlinear Tracking Using Sampling

As a first attempt to use graphical models and message passing algorithms to solve the MTT problem, we assume linear Gaussian models in our developments. However, we believe that our framework, with appropriate modifications, should be powerful enough to handle more general cases, such as nonlinear observation models, which arise when sensors give range-only or bearing-only measurements. We imagine that a nonlinear observation model would require a different set of potential functions (on the graph structure) that are approximated by particle-based representations. In this case, since the underlying probability distribution is not linear-Gaussian anymore, it may make sense to use sampling to reduce the number of particles in messages, rather than using the deterministic hypothesis reduction methods presented in this thesis.

6.2.2 Distributed Multi-Target Tracking

The need to do tracking in a decentralized manner is necessitated by the emergence of distributed sensor networks. Indeed, there have been works in using graphical

models to do distributed data association [6–8]. The MTT message-passing algorithm presented in this thesis is a centralized algorithm that requires a powerful central processing center. At this point, although the graphical model formalism is naturally well-posed to model sensor networks, it is unclear how to map the MTT problem into distributed sensor networks.

6.3 Concluding Remarks

We demonstrated the viability of using graphical models and message passing algorithms to solve the Multi-Target Tracking (MTT) problem. This new approach, as an alternative to the existing Multi-Hypothesis Tracking algorithm, is worth further research efforts as it offers several promising properties including being very efficient and being able to overcome some inherent limitations faced by existing algorithm.

Bibliography

[1] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear non-Gaussian Bayesian tracking. *IEEE Trans. Signal Processing*, 50:174–188, February 2002.

[2] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, Orlando, 1988.

[3] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, II:451–460, 1975.

[4] S. S. Blackman. *Multiple-Target Tracking with Radar Applications*. Artech House, Dedham, MA, 1986.

[5] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19:5–18, Jan 2004.

[6] L. Chen, M. Cetin, and A. S. Willsky. Distributed data association for multi-target tracking in sensor networks. In *Proc. Int. Conf. Information Fusion*, July 2005.

[7] L. Chen, M. Wainwright, M. Cetin, and A. S. Willsky. Multitarget-multisensor data association using the tree-reweighted max-product algorithm. In *Proc. at the SPIE Aerosense conference*, March 2003.

[8] L. Chen, M. J. Wainwright, M. Cetin, and A. S. Willsky. Data association based on optimization in graphical models with application to sensor networks. *Mathematical and Computer Modeling*, 43(9-10):1114–1135, May 2006.

[9] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, New York, 1991.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[11] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[12] J. A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[13] Alexander T. Ihler. *Inference in Sensor Networks: Graphical Models and Particle Methods*. PhD thesis, MIT, 2005.

[14] S. Kullback. *Information Theory and Statistics*. John Wiley, 1959.

[15] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50:157–224, 1988.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.

[17] K. Plarre and P. Kumar. Extended message passing algorithm for inference in loopy gaussian graphical models. In *Ad Hoc Networks*, 2004.

[18] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, AC-24:843–854, 1979.

[19] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition (CVPR)*, 2003.

[20] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper)trees: message-passing and linear programming approaches. In *Proc. Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2002.