MULTIPLE TARGET TRACKING FOR SURVEILLANCE

by

Songhwai Oh

Memorandum No. UCB/ERL M03/54

18 December 2003

MULTIPLE TARGET TRACKING FOR SURVEILLANCE

by

Songhwai Oh

Memorandum No. UCB/ERL M03/54

18 December 2003

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Multiple Target Tracking for Surveillance

by Songhwai Oh

sho@eecs.berkeley.edu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,
University of California at Berkeley,
in partial satisfaction of the requirements for the degree of

Master of Science, Plan II.

Approval for the Report and Comprehensive Examination:

Committee:

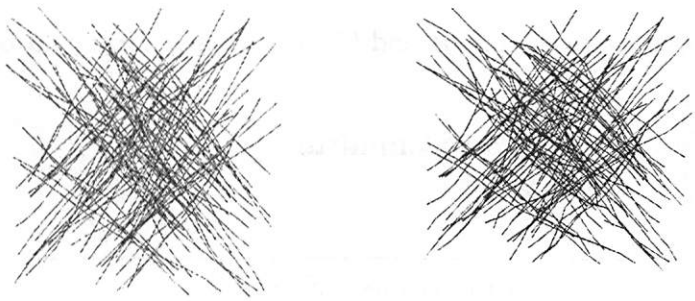
Prof. Shankar S. Sastry
Research Advisor

Date

* * * * * *

Prof. Stuart Russell
Second Reader

Date



To my parents

Contents

Acknowledgements

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

I also would like to thank Professor Stuart Russell for being a reader for this project. This project has benefited tremendously from many discussions with him. His expertise and ingenious questions and suggestions have been inspirational and kept me motivated.

I also would like to thank Dr. Jin Kim for introducing me to the multiple target tracking problem and many delightful discussions about this projects. I also thank Dr. Slobodan Simic for teaching me nonlinear systems and many wonderful discussions. I also thank David Blei and Professor Michael Jordan for discussions about the Bayesian modelling. I also would like to thank Professor Pravin Varaiya and Professor Laurent El Ghaoui.

Lastly, I would like to thank all the members in the Robotics and Intelligent Machines Laboratory at U.C. Berkeley. It was a great pleasure to work with so many extraordinary individuals and I am looking forward to continue working with them for many more years to come.

Chapter 1

Introduction

1.1 Multiple Target Tracking Algorithms

The multiple target tracking plays an important role in many areas of engineering such as surveillance, computer vision, and signal processing (Bar-Shalom & Fortmann, 1988; Cox, 1993). Under the most general setup, a varying number of targets are moving around in a region with continuous motions and the positions of moving targets are sampled at random intervals. The measurements about the positions are noisy, with detection probability less than one, and there is a noise background of spurious position reports (false alarms). Targets arise at random in space and time. Each target persists independently for a random length of time and ceases to exist. A track of a target is defined as a path in space-time traveled by the target. The seminal paper by Sittler (Sittler, 1964) introduced the major concepts about multiple target tracking and a method to evaluate tracks. He pointed out two major problems in multiple target tracking: data association and state estimation. The essence of the multiple target tracking problem is to find tracks from the noisy observations and it requires solutions to both data association and state estimation problems.

In (Sittler, 1964), the data association problem in multiple target tracking is described as a problem of finding a partition of observations such that each element of a partition is a collection of observations generated by a single target or clutter. However, due to the noises in state transition and observation, we cannot expect to find the exact solution. Hence, Sittler developed a probabilistic surveillance model and searched for a partition of observations on which the likelihood function defined by the surveillance model is maximized. This data-oriented view of data association has been applied and extended by many authors (Morefield, 1971; Stein & Blackman, 1975; Reid, 1979; Kurien, 1990; Cox & Hingorani, 1994; Poore, 1995). The most successful multiple target tracking algorithm based on this view is the multiple hypothesis tracker (MHT) (Reid, 1979). A different approach to the data association problem is the probabilistic data association filter (PDAF) (Bar-Shalom & Fortmann, 1988). Under the optimal Bayesian setting, given a fixed number of targets, all possible associations between the known targets and observations from the initial to the present time are weighted by their likelihoods. These weights are called the association weights or probabilities. For each combination, the state of a target is estimated by a filtering algorithm and this conditional expectation is weighted by the association weight. Then the state of each target

is estimated by summing over the weighted conditional expectations. However, the enumeration of all possible associations is not practical so the suboptimal single-stage data association approach, such as the joint probabilistic data association filter (JPDAF), has been applied in practice (Bar-Shalom & Fortmann, 1988). This Bayesian approach to data association has been applied by many authors to different applications (Bar-Shalom & Fortmann, 1988; Streit & Luginbuhl, 1994; Huang & Russell, 1997; Pasula et al., 1999; Schulz et al., 2001; Russell, 2001).

In MHT, each hypothesis associates past observations with a target such that an observation is not shared by more than one target. As a new set of observations arrives, a new set of hypotheses is formed from the previous hypotheses. The construction of new hypotheses requires the enumeration of all possibilities and the size of hypotheses grows exponentially. At each time step, each hypothesis is scored by the probability of having the hypothesis given the observations, i.e., the posterior of the hypothesis. The algorithm returns the hypothesis with the highest score as a solution. MHT is categorized as a deferred logic (Poore, 1995) in which the decision about forming a new track or removing an existing track is delayed until enough observations are collected. Hence, MHT is capable of initiating and terminating a varying number of targets and suitable for surveillance applications in which the tracker is required to initiate and terminate a varying number of tracks autonomously. However, the size of the hypotheses grows exponentially and the enumeration of all hypotheses is not practical. The initial implementation and later extensions proposed several heuristics, such as pruning, gating, clustering and N-scan-back logic, to reduce the complexity of the problem (Reid, 1979; Kurien, 1990). However, the hueristics are used at the expense of the optimality and the algorithm can still suffer in a dense environment. Furthermore, the running time at each step of the algorithm cannot be bounded easily, making it difficult to be deployed in a real-time surveillance. As a method of pruning, an efficient method of finding k-best hypothesis based on the algorithm by Murty (Murty, 1968) is developed in (Cox & Hingorani, 1994).

As opposed to finding the optimal association, JPDAF computes the weights of all the possible associations from the *latest* set of observations to the known tracks and clutter (Bar-Shalom & Fortmann, 1988). Given an association, the state of a target is estimated by a filtering algorithm and this conditional expectation of state is weighted by the association weight. Then the state of a target is estimated by summing over the weighted conditional expectations. JPDAF is a sequential tracker in which the associations between the known targets and the latest observations are made sequentially and the associations made in the past are not reversible (Poore, 1995). The sequential trackers are more efficient than deferred logic trackers such as MHT but they are prone to make erroneous associations (Poore, 1995). At each stage, the association between previously estimated states of targets and the latest observations is optimal from the Bayesian point of view. But the algorithm is suboptimal since the states estimated at each step of algorithm can be different from the states estimated from all observations from the initial to current time. In addition, it is inferred that the exact calculation at each stage is NP-hard (Collins & Uhlmann, 1992) since the related problem of finding the permanent of a 0-1 maxtix is #P-complete (Valiant, 1979). In (Huang & Russell, 1997), a single-stage data association problem is considered and a leave-one-out hueristic is developed to avoid the enumeration of all possible associations. Later, the approach is extended to a multi-stage data association problem using Markov chain Monte Carlo (Pasula et al., 1999).

Since only the current set of observations are considered in JPDAF, it cannot initiate or terminate tracks. Also JPDAF assumes a fixed number of targets and requires a good initial state for each

target. There are restricted extensions to JPDAF to allow the formation of a new track. See (Cox, 1993) for references. Instead of keeping a single Gaussian component for each target, the multisensor multitarget mixture reduction (MTMR) maintains a fixed number of Gaussian components for each target to prevent the loss of information when there are several significant and well spaced componenets (Pao, 1993). It has been shown that MTMR performs better than JPDAF but at the expense of computation (Pao, 1993). But MTMR also assumes a fixed number of targets and requires a good set of initial states. The probabilistic multi-hypothesis tracking (PMHT) uses probabilistic associations between observations and targets to avoid the maintenance of a hypothesis tree and the enumeration over all possible associations (Streit & Luginbuhl, 1994). PMHT allows an association between a single track and an arbitrary number of observations. This assumption may not represent the physical reality in many cases but reduces the complexity of the problem (Hue et al., 2002). However, a fixed known number of targets is assumed and the track initiation and termination are difficult under PMHT. A hypothesis test is presented in (Hue et al., 2002) as a method to allow a varying number of tracks but the paper also addresses the difficulty with estimating the initial states of new targets.

The data association problem of multiple target tracking formulated under the data-oriented view is also known to be NP-hard (Poore, 1995). It is a multidimensional assignment problem (Poore, 1995) and the multidimensional assignment problem is NP-hard since its special case 3-dimensional matching is NP-hard (Papadimitriou & Steiglitz, 1982). There is a polynomial time algorithm for the weighted bipartite matching problem, i.e., 2-dimensional matching problem, but the 3-dimensional matching problem is NP-hard. Hence, we can find an optimal solution to a single-stage data association problem in a polynomial time but cannot expect to find an optimal solution to a multiple-stage data association problem unless P = NP. The multiple target tracking problem was first formulated as a combinatorial optimization problem in (Sittler, 1964). Later, heuristics are added to speed up the algorithm (Reid, 1979; Kurien, 1990; Cox & Hingorani, 1994). An optimization approach to the data association has been applied as a 0-1 integer programming problem (Morefield, 1971) and as a multidimensional assignment problem (Poore, 1995). In both cases one needs to find a feasible set of tracks from all possible tracks, i.e., all possible partitions of observations, to prevent the exponential explosion and compute the cost of each feasible track, such as the negative log likelihood. Then the optimization routine finds a subset from the feasible tracks such that the combined costs are minimized while satisfying the constraints, i.e., each track has at most one observation at each time and no two tracks share the same observation. The gating method similar to the ones described in (Stein & Blackman, 1975; Reid, 1979) is used to find a feasible set of tracks. However, in a dense environment, the size of the feasible tracks can be very large and the complexity of the optimization routine increases dramatically, since the number of parameters in the optimization routine depends on the number of feasible tracks.

1.2 Project Summary

In this project, two new algorithms are developed to solve the multiple target tracking problems. The main objective of this project is to develop efficient algorithms for autonomous surveillance. So we take a more general approach toward the multiple target tracking problem. For example, we do not assume that the number of targets in the surveillance region is known in advance. Hence,

it is important for the algorithm to initiate and terminate tracks. The first algorithm described in Chapter 2 is a model-based approach and more flexible than the data-based approach shown in Chapter 3. We have found that the first approach is more accurate when we are more uncertain about the model but the second approach is more computationally efficient.

In Chapter 2, we propose a new probabilistic framework for nonparametric identification and estimation of dynamic systems. Under the parametric paradigm, a model of the system and a set of observations are given and the parameter space of the model is searched to optimize an objective function. However, if we are uncertain about the model, the parametric approach can easily overfit data and lead to risky decisions. In nonparametric estimation, the model uncertainty is introduced in a systematic manner to find both the model and associated parameters of the system. We consider a dynamic system consisting of a varing number of subsystems with noisy observations. The objective is to identify the subsystems at each time and estimate the associated parameters such that the observations are explained the best. We develop an efficient algorithm based on Markov chain Monte Carlo (MCMC) methods and apply our approach to multiple target tracking problems. We address the issues with the subsystem initiation and termination and initial state estimation. In simulation our algorithm shows an excellent performance for tracking a varying number of maneuvering targets with nonlinear dynamics. In some cases the algorithm outperforms any linear filtering algorithm with perfect associations.

In Chapter 3, a general purpose MCMC algorithm for data association is presented. We take the data-oriented, combinatorial optimization approach to the data association problems but avoid the enumeration of tracks by applying a sampling method called Markov chain Monte Carlo (MCMC). The MCMC data association algorithm is very flexible and easy to incorporate any domain specific knowledge to make it more efficient. Instead of searching over the whole state space, the MCMC algorithm randomly searches over the space where the posterior is concentrated. The simulation results show a remarkable performance of the MCMC tracker under the extreme conditions such as a large number of targets in a dense environment, low detection probabilities, and a large number of false alarms.

Chapter 2

A Sampling-Based Approach to Dynamic System Identification and Estimation for Multiple Target Tracking

2.1 Introduction

In parametric estimation problems, we are given a set of (noisy) observations from a known model and the goal is to estimate the parameters of the model such that some objective function is maximized (or minimized). For example, in linear regression, we minimize the sum of squared errors. In many practical problems the method of maximum likelihood provides a good solution for parametric estimation problems (Fisher, 1922). However, if there is uncertainty about the model, the parametric estimation methods, such as the maximum likelihood estimation, can easily overfit data and lead to risky decisions. The recognition of the limitations of parametric regression led to the surge of nonparametric methods such as an information criterion (Akaike, 1997), Bayesian information criterion (Schwarz, 1978), minimum description length (Hansen & Yu, 2001) and Bayes factors (Kass & Raftery, 1995). But difficulties in formulating the prior models and testing for different models have made them difficult to be applied to complex problems.

Solving complex problems by sampling methods such as Markov chain Monte Carlo (MCMC) has become more tractable, due the increased computational power. MCMC-based algorithms play a significant role in many fields such as physics, statistics, economics, and engineering (Beichl & Sullivan, 2000). In some cases MCMC is the only known general algorithm which finds a good solution in a polynomial time (Jerrum & Sinclair, 1996). For interested readers we refer to (Beichl & Sullivan, 2000), (Andrieu et al., 2003) and (Gilks et al., 1996). However, due to the measure theoretic issues, the standard MCMC is limited to problems with a fixed dimension, i.e., the parametric estimation problems, and cannot be easily extended to problems with varying dimensional parameters, i.e., the nonparametric estimation problems. Reversible jump MCMC (RJMCMC) solves this

issue by introducing reversible jumps between dimensions using deterministic dimension-matching transformations (Green, 1995). This method is different from other nonparametric methods, in that it allows the comparison among different models simultaneously while estimating parameters associated with each model. The different models are considered simultaneously based on the evidence so that the models not supported by the evidence are not considered as frequently as the models supported by the evidence. In addition, the method provides numerical values representing confidence in different models and bypasses the model choice structure of the other nonparametric methods, which requires prior modelling and testing for the different models (Robert et al., 2000).

In this chapter, we develop a probabilistic framework for nonparametric identification and estimation of dynamic systems. We consider a dynamic system with noisy observations, which consists of a varying number of subsystems. The goal is to identify the subsystems at each time step and estimate the associated parameters, including the dynamic of each subsystem, such that the observations are explained the best. Each subsystem can be either present or absent at each time following a Markov transition probability independently from the other subsystems. In addition the initial state of a subsystem is unknown when it appears or reappears. The main difficulty is the identification of subsystem initiation and termination times. We take the Bayesian hierarchical modelling approach (Richardson & Green, 1997) and formulate a general framework. Under our framework, observations are partitioned into the subsystems that they belong to, and thus, it becomes a trivial task to use the existing efficient parametric methods to estimate the parameters of subsystems' dynamics.

The multiple target tracking problem is a good example of the dynamic system we have described. Under the most general setup, a varying number of targets are moving around in a region with continuous motions and the positions of moving targets are sampled at random intervals. The measurements about the positions are noisy, with detection probability less than one and there is a noise background of spurious position reports (false alarms). Targets arise at random in space and time. Each target persists independently for a random length of time and ceases to exist. A track of a target is defined as a path in space-time traveled by the target. The seminal paper by Sittler (Sittler, 1964) introduced the major concepts about multiple target tracking and a method to evaluate tracks. He pointed out two major problems in multiple target tracking: data association and state estimation. However, in worst case, one has to search over all possible tracks which is equivalent to searching over the collection of all partitions of observations. This combinatorial optimization problem is NP-hard since it is a multidimensional assignment problem (Poore, 1995). Later, more computationally efficient algorithms were proposed using heuristics to reduce the size of the search space (Stein & Blackman, 1975) (Reid, 1979). The multiple hypothesis tracker (MHT) introduced in (Reid, 1979) used hueristics such as pruning, gating and N-scan-back logic but at the expense of accuracy.

As opposed to finding the optimal association, there is an alternative suboptimal approach for data association, called the joint-probabilistic data-association filter (JPDAF) (Bar-Shalom & Fortmann, 1988). The main limitations of JPDAF are that it assumes a fixed number of targets at all times and it cannot initiate or terminate tracks. On the other hand, the probabilistic multi-hypothesis tracking (PMHT) uses probabilistic associations between observations and targets to avoid the maintenance of a hypothesis tree (Streit & Luginbuhl, 1994). But PMHT also assumes a fixed number of targets and does not allow track initiation and termination. Our framework can be

considered as a full statistical extension of PMHT for tracking an unknown number of targets. With the development of particle filters, multitarget tracking algorithms have been extended to nonlinear dynamics. In (Hue et al., 2002), PMHT is applied with particle filters, but the method requires suboptimal estimation of the states of targets. JPDAF is used in (Schulz et al., 2001) with particle filters, but the described limitations of JPDAF remain the same. See (Cox, 1993) for more information about multitarget tracking algorithms.

In this chapter, we construct a new probabilistic framework for layered dynamic systems and develop an efficient RJMCMC based algorithm. We are able to construct an efficient sampler by taking the advantages of the structure of the problem and proposing targeted local moves. We apply this method to multitarget tracking problems, and demonstrate the robustness of the algorithm against outliers and its scalability. In some cases, when we apply our algorithm to nonlinear dynamics, it outperforms any linear filtering algorithm with perfect associations.

The remainder of this chapter is structured as follows. Section 2.2 presents a probabilistic framework for nonparametric identification and estimation of dynamic systems, called the dynamic Bayesian model selection. Section 2.3 describes an RJMCMC algorithm for solving the nonparametric identification and estimation problems. We also describe an online version of the algorithm. In Section 2.4, the algorithm is applied to track a varying number of targets moving with nonlinear dynamics, and the performance of the algorithm is analyzed.

2.2 Dynamic Bayesian Model Selection

Let us consider a discrete-time dynamic system $\mathcal{S}$ composed of subsystems $\mathcal{S}_i$ for $i = 1, \ldots, K$ where K is the total number of subsystems. Each subsystem can be either present or absent at any given time. When a subsystem is present over consecutive times, its state evolves according to unique dynamics independently from the other subsystems. Let $\mathbb{R}^n$ be the state space of each subsystem. Let $X_t^i \in \mathcal{R} \subset \mathbb{R}^n$ be the state of $\mathcal{S}_i$ for $t = 1, \ldots, T$. $\mathcal{R}$ has a finite volume $V_{\mathcal{R}}$. The dynamic of $\mathcal{S}_i$ is $F^i : \mathbb{R}^n \to \mathbb{R}^n$. Let $F = \{F^i : 1 \leq i \leq K\}$. The state transition is noisy such that

$$X_{t+1}^i = F^i(X_t^i) + V_t^i,$$

where $V_t^i \in \mathbb{R}^n$ is a noise process. When there is no confusion we will denote the subsystem $\mathcal{S}_i$ as the subsystem i. Let $M_t^i \in \{0, 1\}$ be a Markov chain denoting the status of the subsystem i, such that $M_t^i = 1$ if the subsystem i is present; otherwise $M_t^i = 0$. Let $p^i(j, k)$ for $j, k \in \{0, 1\}$ be the transition probability of M_t^i. Using the independence assumption, we can combine M_t^i into a single Markov chain $M_t \in \{0, 1\}^K$ on a product state space with the transition probability

$$P(M_{t+1} = m_{t+1} | M_t = m_t) = \prod_{i=1}^{K} p^i(m_t^i, m_{t+1}^i).$$

Let A be the transition matrix for M_t and the i-th row of A be distributed from the Dirichlet distribution $\mathcal{D}(\alpha_i^1, \ldots, \alpha_i^{2^K})$, where $\alpha \in [0, 1]^{2^K \times 2^K}$. Since we do not assume that the transition matrix A is known in advance, a prior model on A allows us to estimate A based on observations.

Let X_t be the state of $\mathcal{S}$ at time t such that $X_t = ({X_t^{i_1}}^T, \ldots, {X_t^{i_{k_t}}}^T)^T \in \mathbb{R}^{n \times k_t}$, where $k_t = \#\{i : M_t^i = 1, 1 \leq i \leq K\}$ and $M_t^{i_r} = 1$ for $r = 1, \ldots, k_t$. Let f^i be the conditional density of the next state of the subsystem i given its current state and control. Then the conditional probability of X_{t+1} is given by

$$P(X_{t+1} \in dx_{t+1} | X_t = x_t, M_t = m_t, M_{t+1} = m_{t+1}) = \prod_{i=1}^{K} \begin{cases} f^i(dx_{t+1}^i | x_t^i) & \text{if } m_{t+1}^i = 1, m_t^i = 1 \\ p_0^i(dx_{t+1}^i) & \text{if } m_{t+1}^i = 1, m_t^i = 0 \\ 1 & \text{otherwise,} \end{cases}$$

where $p_0^i(x)$ is a priori density of the initial state of the subsystem i. See Figure 2.1 for the graphical representation of the transition model. The values of the shaded nodes are fixed.

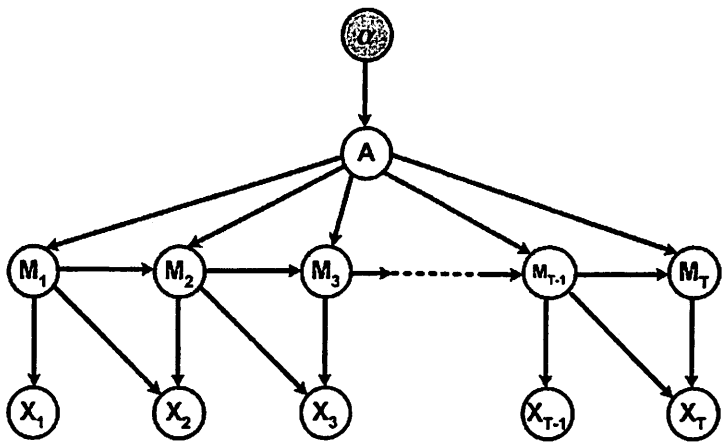


Figure 2.1: A graphical representation of the transition model

Let $Y_t^j \in \mathbb{R}^m$ be an observation vector for $j = 1, \ldots, n_t$. The observation model is

$$Y_t^j = H(X_t^i) + V_t'^j,$$

where $V_t'^j \in \mathbb{R}^m$ is a noise process. Let $h(y|x)$ be the density of the observation y given the state x. Now we assume that the observations $Y_t = \{Y_t^1, \ldots, Y_t^{n_t}\}$ are distributed according to the mixture model such that

$$Y_t^j \sim \sum_{r=0}^{k_t} w_t^{i_r} h(\cdot | x_t^{i_r}) \qquad \text{for } j = 1, 2, \ldots, n_t,$$

where $w_t^{i_r}$ is the weight of the subsystem i_r and $\sum_{r=0}^{k_t} w_t^{i_r} = 1$. The weight w_t^i can be interpreted as the chance that a randomly chosen observation is generated from the subsystem i. Let W_t be r.v.'s associated with the weights w_t. We assume W_t are distributed from the Dirichlet distribution $\mathcal{D}(\delta_t^0, \ldots, \delta_t^{k_t})$ and independent over time, i.e., W_t is independent of W_s for $t \neq s$.

As in (Richardson & Green, 1997), we define allocation variables Z_t such that $Z_t^j = i_r$ if j-th observation comes from the subsystem i_r and Z_t^j are drawn from the distributions $P(Z_t^j = i_r) = w_t^{i_r}$ for $r = 0, 1, \ldots, k_t$. We associate all observations that are not generated from the known subsystems with clutter and let w_t^0 be its weight. The observations from clutter are uniformly distributed over

$\mathcal{R}^m \subset \mathcal{R}$. Let $V_{\mathcal{R}^m}$ be the volume of $\mathcal{R}^m$. We consider these observations as outliers. If the j-th observation is generated from clutter, we set $Z_t^j = 0$. Hence, given the allocation variable Z_t, the observations are distributed as

$$P(Y_t^j \in dy | Z_t = z, X_t = x) = \begin{cases} h(dy|x^{z^j}), & z^j \neq 0 \\ dy/V_{\mathcal{R}^m}, & z^j = 0, \end{cases}$$

independently for $j = 1, \ldots, n_t$. We note that this mixture model formulation is similar to the probabilistic associations of PMHT (Streit & Luginbuhl, 1994).

We assume V_t^i is an i.i.d. white Gaussian process with zero mean and a covariance matrix Σ^i. Σ^i have the scaled inverse χ^2 distribution, i.e., $\Sigma^i_{(l,l)} \sim \text{Inv-}\chi^2(\nu_{il}, {\sigma_{0_{il}}}^2)$ for $l = 1, \ldots, n$ (Gelman et al., 1995). The hyperparameter ν_{il} is called the degree of freedom and $\sigma_{0_{il}}$ is the scale parameter. We also assume $V_t'^j$ is an i.i.d. white Gaussian process with zero mean and a covariance matrix Σ^y, and the hyperparameters are assigned similarly. See Figure 2.2 for the graphical representation of the model at time t.

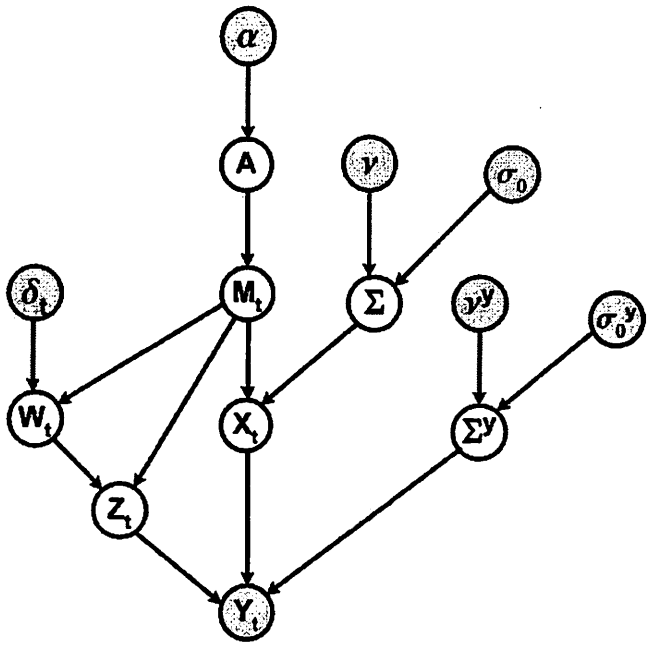


Figure 2.2: A graphical representation of the model at time t

Now to make an inference on the described model, we need to estimate the dimensionality k_t and a set of parameters associated with k_t for every t. However, since at each t explicit associations between parameters at $t-1$, t and $t+1$ are required for the state transitions of subsystems, we consider each instance of the subsystem status as a model. So there are 2^K possible models at each t. The model labelling is explicit in our formulation and we work with the union space $C = \bigcup_{k \in \mathcal{K}} C_k$, where C_k is a parameter subspace associated with the k-th model and $\mathcal{K} = \{1, \ldots, 2^K\}$. In next sections we show that this highly complex model selection problem is efficiently solved by the reversible jump MCMC. Lastly, the joint distribution of all the variables (except those for the noise

models) can be expressed as

$$\begin{aligned} P(M_{1:T}, X_{1:T}, Y_{1:T}, Z_{1:T}, W_{1:T}, \delta_{1:T}, A, \alpha, F) &= P(F)P(\alpha)P(A|\alpha)P(X_1, M_1) \\ &\times \prod_{t=1}^{T-1} P(M_{t+1}, X_{t+1}|M_t, X_t, F, A) \\ &\times \prod_{t=1}^{T} P(Y_t|Z_t, X_t)P(Z_t|W_t, M_t)P(W_t|\delta_t, M_t)P(\delta_t), \end{aligned}$$

where $x_{1:T} = \{x_1, \ldots, x_T\}$. Hence, our main objective is to estimate $P(M_{1:T}|Y_{1:T})$, $P(X_{1:T}|M_{1:T}, Y_{1:T})$, $P(A|Y_{1:T})$, $F|Y_{1:T}$ and so on.

2.3 Algorithm

2.3.1 Markov Chain Monte Carlo

Many practical problems are involved with high-dimensional, high-complexity probability distributions. In order to make an inference or prediction, one must integrate over these complex distributions but there is rarely a closed-form analytical expression for the high-dimensional integrals. Markov chain Monte Carlo (MCMC) is a family of stochastic algorithms that uses Markov chains to estimate the integrals which have no closed-form analytical expressions.

In MCMC, an irreducible and aperiodic Markov chain is constructed such that its stationary distribution is the posterior $\pi(\theta)$ where θ is the set of parameters of interests. Then given the current state θ, the sampler proposes a candidate state θ' from a proposal distribution $q(\theta, \theta')$. Then the proposal is accepted with probability

$$\min\left\{1, \frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')}\right\}$$

so that the detailed balance of the Markov chain is preserved. Then by the ergodic theorem, one can estimate $E_{\pi(\theta)}[f(\theta)]$ of a bounded function $f(\cdot)$ by

$$\bar{f} = \frac{1}{n-m}\sum_{i=m+1}^{n} f(\theta_i)$$

where θ_i is the state of the Markov chain at the i-th iteration (Gilks et al., 1996). The first m samples are burn-ins and discarded for the estimate calculations.

2.3.2 Reversible Jump Markov Chain Monte Carlo

However the standard MCMC described above is limited to problems with a fixed dimension and cannot be easily extended to problems with varying dimension parameters due to the measure theoretic issues. The reversible jump MCMC (RJMCMC) resolves this issue by introducing reversible

jumps between dimensions using deterministic dimension-matching transformations (Green, 1995). In RJMCMC, different types of moves are considered to traverse across the combined parameter space $\mathcal{C}$. Recall that $\mathcal{C} = \bigcup_{k\in\mathcal{K}} \mathcal{C}_k$, where $\mathcal{C}_k$ is a parameter subspace associated with the k-th model and $\mathcal{K} = \{1, \ldots, 2^K\}$. Let $\pi(\theta)$ be the posterior where θ is the set of parameters of interests. If the MCMC sampler proposes a move type m and a new state θ' from θ, the move is accepted with probability

$$\min\left\{1, \frac{\pi(\theta')j_m(\theta')q_{m2}(u')}{\pi(\theta)j_m(\theta)q_{m1}(u)} \cdot \left|\frac{\partial g_m(\theta,u)}{\partial(\theta,u)}\right|\right\},$$

where $j_m(\theta)$ is the probability of choosing the move type m when in state θ; $q_{m1}(u)$ and $q_{m2}(u')$ are the proposal densities of u and u', respectively; and g_m is the deterministic dimension-matching bijection such that $g_m(\theta,u) = (\theta',u')$ and $g_m^{-1}(\theta',u') = (\theta,u)$. The Jacobian arises from the change of variables from (θ,u) to (θ',u') (Richardson & Green, 1997).

2.3.3 DBMS RJMCMC

In this section we present the RJMCMC algorithm for solving DBMS problems. Let

$$\theta = (M_{1:T}, X_{1:T}, Z_{1:T}, W_{1:T}, A, \Sigma^1, \ldots, \Sigma^K, \Sigma^y, F)$$

be the set of the unknowns of which we are interested in finding the posterior $\pi(\theta)$ given $y_{1:T}$. In construction of the algorithm we assume that $\delta = \delta_t^0 = \cdots = \delta_t^{k_t}$ for all t and both δ and α are held fixed.

Each MCMC step consists of following moves:

(a) track split, track merge or track update move on randomly selected type i and time t,

(b) weight update move for W_t,

(c) allocation update move for Z_t,

(d) transition matrix update move for A,

(e) covariance matrices update move,

(f) dynamics update move.

The track split and merge moves of (a) are dimension-varying moves while the other moves are within the same dimension. The moves (b) and (c) are the Gibbs moves on W_t and Z_t for t selected in (a). In moves (a)-(c), we proposes a new state θ' which differs from θ only at some time t. Since only a small neighborhood of t is affected by this move, the acceptance probability of the move can be computed efficiently. In addition, each dimension-varying move takes time linear in the number of observations n_t. Since it is required to consider parameters at all times for the moves (d)-(f), we choose the moves (d)-(f) with some small probability p_{all} while the moves (a)-(c) are chosen with probability $1 - p_{all}$.

We choose a track split move with probability b_{n_0} and a track merge move with probability d_{n_1} such that $b_{n_0} + d_{n_1} < 1$ where $n_1 = \#\{i : M_t^i = 1, 1 \le i \le K, 1 \le t \le T\}$ and $n_0 = KT - n_1$. A track update move is chosen with probability $1 - b_{n_0} - d_{n_1}$. For the convenience of our discussion below, we define submove types. We say the move at (i,t) has:

- submove type (i) if $M_{t-1}^i = 0$ and $M_{t+1}^i = 0$,
- submove type (ii) if $M_{t-1}^i = 0$ and $M_{t+1}^i = 1$,
- submove type (iii) if $M_{t-1}^i = 1$ and $M_{t+1}^i = 0$,
- submove type (iv) if $M_{t-1}^i = 1$ and $M_{t+1}^i = 1$.

2.3.3.1 Gibbs moves

To update the weights, we draw W_t from the posterior given all other variables, $P(W_t|\cdots)$. Since the posterior remains as Dirichlet, we draw W_t as

$$W_t|\cdots \sim \mathcal{D}(\delta + d_0, \delta + d_{i_1}, \ldots, \delta + d_{i_{k_t}}),$$

where $d_{i_p} = \#\{j : Z_t^j = i_p\}$, where i_p is defined such that $M_t^{i_p} = 1$ for $1 \le p \le k_t$ and $i_0 = 0$.

The allocation variables Z_t are drawn from the posterior

$$P(Z_t^j = i_p|\cdots) \propto \begin{cases} w_t^{i_p} \cdot h(y_t^j|x_t^{i_p}), & i_p \neq 0 \\ w_t^0 \cdot \frac{1}{V_{\mathcal{R}^m}}, & i_p = 0. \end{cases}$$

To update the covariance matrix Σ^i for $1 \le i \le K$, we first let

$$s_i = \sum_{t=1}^{T-1} \mathbb{I}(M_t^i = 1, M_{t+1}^i = 1)$$

and

$$v_{ij} = \frac{1}{s_i} \sum_{t=1}^{T-1} \mathbb{I}(M_t^i = 1, M_{t+1}^i = 1)(X_{t+1}^i - F^i(X_t^i))_j^2$$

where $\mathbb{I}(\cdot)$ is an indicator function and $(x^i)_j$ returns the j-th component of x^i. The j-th diagonal element of Σ^i is drawn from its posterior

$$\Sigma_{(j,j)}^i|\cdots \sim \text{Inv-}\chi^2\left(\nu_{ij} + s_i, \frac{\nu_{ij}\sigma_{0_{ij}}^2 + s_i v_{ij}}{\nu_{ij} + s_i}\right),$$

for $j = 1, \ldots, n$. We apply the similar method to update the covariance Σ^y.

The update of A is similar to the update of W_t. The i-th row of A is drawn from

$$(A)_{\text{row-}i}|\cdots \sim \mathcal{D}(\alpha_i^1 + b_{i1}, \ldots, \alpha_i^{2^K} + b_{i2^K}),$$

where $b_{ij} = \#\{j : M_t = i, M_{t+1} = j, 1 \le t \le T-1\}$.

2.3.3.2 Track Update Moves

We choose (i,t) randomly from

$$G_1 = \{(i,t) : M_t^i = 1, 1 \le i \le K, 1 \le t \le T\}.$$

For submove type (i), we choose the placement update; for submove (ii), we choose the backward update; for submove (iii), we choose the forward update; and for submove (iv), we choose the forward update with probability 1/2 and the backward update with probability 1/2. The forward and backward moves can be thought of as filtering and smoothing, respectively. Each move is accepted with probability $\min(1,R)$ where the acceptance ratio R is described below for each move.

Placement update
The placement update move proposes a new state $X_t'^i \sim \frac{1}{n_t}\sum_{j=1}^{n_t} \mathcal{N}(\cdot|y_t^j, \Sigma^y)$. The move is accepted with probability $\min(1,R)$ where

$$\begin{aligned} R &= \frac{\pi(\theta')}{\pi(\theta)}\frac{q(X_t^i)}{q(X_t'^i)} \\ &= \frac{P(y_t|Z_t, X_t')p_0^i(X_t'^i)}{P(y_t|Z_t, X_t)p_0^i(X_t^i)}\frac{q(X_t^i)}{q(X_t'^i)} \end{aligned}$$

where $q(x)$ is the proposal density of x and $X_t' = \{X_t \setminus X_t^i\} \cup \{X_t'^i\}$.

Forward update
The forward update move proposes a new state $X_t'^i = F^i(X_{t-1}^i) + \eta$ where $\eta \sim \mathcal{N}(\cdot|0,\Sigma^i)$ and if the submove type is (iii), the acceptance ratio is

$$R = \frac{P(y_t|Z_t, X_t')f^i(X_t'^i|X_{t-1}^i)}{P(y_t|Z_t, X_t)f^i(X_t^i|X_{t-1}^i)}\frac{q(X_t^i)}{q(X_t'^i)} = \frac{P(y_t|Z_t, X_t')}{P(y_t|Z_t, X_t)}.$$

Notice that the acceptance ratio calculation is simplified by our choice of proposal distribution. If the submove type is (iv), the acceptance ratio is

$$\begin{aligned} R &= \frac{P(y_t|Z_t, X_t')f^i(X_t'^i|X_{t-1}^i)f^i(X_{t+1}^i|X_t'^i)}{P(y_t|Z_t, X_t)f^i(X_t^i|X_{t-1}^i)f^i(X_{t+1}^i|X_t^i)}\frac{q(X_t^i)}{q(X_t'^i)} \\ &= \frac{P(y_t|Z_t, X_t')f^i(X_{t+1}^i|X_t'^i)}{P(y_t|Z_t, X_t)f^i(X_{t+1}^i|X_t^i)}. \end{aligned}$$

Backward update
The backward update move is available when $(F^i)^{-1}$ exists and is differentiable. To propose a backward update, we first sample $\eta \sim \mathcal{N}(\cdot|0,\Sigma^i)$ and set $X_t'^i = (F^i)^{-1}(X_{t+1}^i - \eta)$. Let g be the function of this mapping. Since many variables are unchanged and the Jacobian is identity for them, we only display the variables involved in this transformation.

$$g(X_t^i, X_{t+1}^i, \eta) = (X_t'^i, X_{t+1}'^i, \eta') = \left((F^i)^{-1}(X_{t+1}^i - \eta), X_{t+1}^i, X_t^i\right)$$

and its Jacobian is

$$\left|\frac{\partial g}{\partial X_t \partial \eta}\right| = \left|\frac{\partial (F^i)^{-1}}{\partial \eta}(X_{t+1}^i - \eta)\right|.$$

Now for submove type (ii), the acceptance ratio is

$$\begin{aligned} R &= \frac{P(y_t|Z_t, X_t')f^i(X_{t+1}^i|X_t'^i)}{P(y_t|Z_t, X_t)f^i(X_{t+1}^i|X_t^i)}\frac{q(X_t^i)}{q(X_t'^i)}\left|\frac{\partial (F^i)^{-1}}{\partial \eta}(X_{t+1}^i - \eta)\right| \\ &= \frac{P(y_t|Z_t, X_t')}{P(y_t|Z_t, X_t)}\left|\frac{\partial (F^i)^{-1}}{\partial \eta}(X_{t+1}^i - \eta)\right| \end{aligned}$$

and for submove type (iv), the acceptance ratio is

$$\begin{aligned} R &= \frac{P(y_t|Z_t, X_t')f^i(X_t'^i|X_{t-1}^i)f^i(X_{t+1}^i|X_t'^i)}{P(y_t|Z_t, X_t)f^i(X_t^i|X_{t-1}^i)f^i(X_{t+1}^i|X_t^i)}\frac{q(X_t^i)}{q(X_t'^i)}\left|\frac{\partial (F^i)^{-1}}{\partial \eta}(X_{t+1}^i - \eta)\right| \\ &= \frac{P(y_t|Z_t, X_t')f^i(X_t'^i|X_{t-1}^i)}{P(y_t|Z_t, X_t)f^i(X_t^i|X_{t-1}^i)}\left|\frac{\partial (F^i)^{-1}}{\partial \eta}(X_{t+1}^i - \eta)\right|. \end{aligned}$$

2.3.3.3 Track Split and Merge Moves

For a split move, we select (i,t) with probability $p_b(\theta,(i,t))$, where $p_b(\theta,(i,t)) > 0$ on G_1^C and $p_b(\theta,(i,t)) = 0$ on G_1, and propose a new component at (i,t) by splitting the weight of clutter W_t^0 into $W_t'^0$ and $W_t'^i$. The move is rejected if $W_t^0 = 0$ or $|n_t^0| = 0$ where $n_t^0 = \{j : Z_t^j = 0\}$ since such move is not reversible. For a merge move, we select (i,t) with probability $p_d(\theta,(i,t))$, where $p_d(\theta,(i,t)) > 0$ on G_1 and $p_d(\theta,(i,t)) = 0$ on G_1^C, and merge the component at (i,t) into clutter.

In a track split move, we take the following steps to propose θ':

1. Propose η_1 from Beta distribution, $\eta_1 \sim Be(2,2)$;
2. Set $W_t'^0 = W_t^0\eta_1$ and $W_t'^i = W_t^0(1-\eta_1)$;
3. Set $M_t'^i = 1$ and $M_t'^r = M_t^r$ for $r \neq i$;
4. Propose η_2 for $X_t'^i$. The proposal of η_2 is different for each submove type and it is described below;
5. Propose the allocation variables $Z_t'^j \in \{0,i\}$ for $j \in n_t^0$ using the posterior of Z_t given the other variables, i.e.,
$$P(Z_t'^j = r|\cdots) \propto \begin{cases} w_t'^i \cdot h(y_t^j|x_t^i), & r = i \\ w_t'^0 \cdot \frac{1}{V_{\mathcal{R}^m}}, & r = 0, \end{cases}$$
for $j \in n_t^0$.
6. The remaining variables are unchanged.

There are again three different split (merge) moves: placement split (merge), forward split (merge), and backward split (merge). For submove (i), we choose the placement split; for submove (iii) and (iv), we choose the forward split; and for submove (ii), we choose the backward split. In general the acceptance probability is $\min(1, R)$ with an acceptance ratio R. A similar merge move is applied for each submove type and a merge move is accepted with probability $\min(1, R^{-1})$ with appropriate substitutions.

The following prior ratio is common for all the moves described below

$$\frac{P(Z'_t, W'_t|M'_t, \delta)}{P(Z_t, W_t|M_t, \delta)} = \frac{(W'^0_t)^{\delta-1+l_0}(W'^i_t)^{\delta-1+l_i}}{(W^0_t)^{\delta-1+l_0+l_i}B((k_t+1)\delta, \delta)}$$

where $l_0 = \#\{j : Z'^j_t = 0, j \in n^0_t\}$ and $l_i = \#\{j : Z'^j_t = i, j \in n^0_t\}$ and $B(\cdot)$ is the Beta function.

Placement split

A new state X'^i_t is proposed as

$$X'^i_t \sim \frac{1}{n^0_t} \sum_{j \in n^0_t} \mathcal{N}(\cdot|y^j_t, \Sigma^y).$$

The acceptance ratio becomes

$$\begin{aligned} R &= \frac{\pi(\theta')}{\pi(\theta)} \cdot \frac{j_m(\theta')}{j_m(\theta)} \cdot \frac{1}{q(\eta_1)q(X'^i_t)q(Z'_t)} \cdot \left|\frac{\partial g_m(\theta, \eta)}{\partial(\theta, \eta)}\right| \\ &= \frac{P(y_t|Z'_t, X'_t)A_{M_{t-1}M'_t}A_{M'_tM_{t+1}}}{P(y_t|Z_t, X_t)A_{M_{t-1}M_t}A_{M_tM_{t+1}}} \frac{P(Z'_t, W'_t|M'_t, \delta)}{P(Z_t, W_t|M_t, \delta)} \frac{p^i_0(X'^i_t)}{1} \\ &\times \frac{d_{n_1+1}p_d(\theta', (i,t))}{b_{n_0}p_b(\theta, (i,t))} \cdot \frac{1}{q(\eta_1)q(X'^i_t)q(Z'_t)} \cdot W^0_t. \end{aligned}$$

Forward split

For the forward split, a new state X'^i_t is proposed from $F^i(X^i_{t-1}) + \eta_2$ where $\eta_2 \sim \mathcal{N}(\cdot|0, \Sigma^i)$. For submove type (iii),

$$\begin{aligned} R &= \frac{P(y_t|Z'_t, X'_t)A_{M_{t-1}M'_t}A_{M'_tM_{t+1}}}{P(y_t|Z_t, X_t)A_{M_{t-1}M_t}A_{M_tM_{t+1}}} \frac{P(Z'_t, W'_t|M'_t, \delta)}{P(Z_t, W_t|M_t, \delta)} \frac{f^i(X'^i_t|X^i_{t-1})}{1} \\ &\times \frac{d_{n_1+1}p_d(\theta', (i,t))}{b_{n_0}p_b(\theta, (i,t))} \frac{1}{q(\eta_1)q(X'^i_t)q(Z'_t)} W^0_t \\ &= \frac{P(y_t|Z'_t, X'_t)A_{M_{t-1}M'_t}A_{M'_tM_{t+1}}}{P(y_t|Z_t, X_t)A_{M_{t-1}M_t}A_{M_tM_{t+1}}} \frac{P(Z'_t, W'_t|M'_t, \delta)}{P(Z_t, W_t|M_t, \delta)} \\ &\times \frac{d_{n_1+1}p_d(\theta', (i,t))}{b_{n_0}p_b(\theta, (i,t))} \frac{1}{q(\eta_1)q(Z'_t)} W^0_t \end{aligned}$$

and for submove type (iv),

$$\begin{aligned} R &= \frac{P(y_t|Z'_t, X'_t)A_{M_{t-1}M'_t}A_{M'_tM_{t+1}}}{P(y_t|Z_t, X_t)A_{M_{t-1}M_t}A_{M_tM_{t+1}}} \frac{P(Z'_t, W'_t|M'_t, \delta)}{P(Z_t, W_t|M_t, \delta)} \frac{f^i(X^i_{t+1}|X'^i_t)}{p^i_0(X^i_{t+1})} \\ &\times \frac{d_{n_1+1}p_d(\theta', (i,t))}{b_{n_0}p_b(\theta, (i,t))} \frac{1}{q(\eta_1)q(Z'_t)} W^0_t. \end{aligned}$$

Backward split
As done in the track update, we sample $\eta_2 \sim \mathcal{N}(\cdot|0, \Sigma^i)$ and set $X_t'^i = (F^i)^{-1}(X_{t+1}^i - \eta_2)$. So the acceptance ratio is

$$\begin{aligned} R &= \frac{P(y_t|Z_t', X_t')A_{M_{t-1}M_t'}A_{M_t'M_{t+1}}}{P(y_t|Z_t, X_t)A_{M_{t-1}M_t}A_{M_tM_{t+1}}} \frac{P(Z_t', W_t'|M_t', \delta)}{P(Z_t, W_t|M_t, \delta)} \frac{p_0^i(X_t'^i)}{p_0^i(X_{t+1}^i)} \\ &\times \frac{d_{n_1+1}p_d(\theta', (i,t))}{b_{n_0}p_b(\theta, (i,t))} \frac{1}{q(\eta_1)q(Z_t')} \left| \frac{\partial (F^i)^{-1}}{\partial \eta_2}(X_{t+1}^i - \eta_2) \right| W_t^0 \end{aligned}$$

2.3.4 Online DBMS RJMCMC

The algorithm described in previous section is a batch algorithm. A suboptimal online version of the algorithm can be easily implemented by sliding a window of length T_w. Suppose that the window is forwarded by s time steps for each online step. For $k = 0, 1, 2, \ldots$, we run the algorithm on observations $y_{sk+1:sk+T_w}$ as soon as they become available. At k-th online step, the algorithm returns a set of estimates $(\hat{\theta}_{sk+1:sk+T_w})$ along with posterior probabilities. We choose $\hat{M}_{sk+1:sk+T_w}$ which maximizes the posterior $P(M_{sk+1:sk+T_w}|y_{sk+1:sk+T_w})$ and estimate $\hat{X}_{sk+1:sk+T_w}$ given $\hat{M}_{sk+1:sk+T_w}$. Then for the next $(k+1)$-th online step, the sampler is initialized with $\hat{M}_{s(k+1)+1:sk+T_w}$ and $\hat{X}_{s(k+1)+1:sk+T_w}$. The major benefit of our algorithm is that the estimation errors made in earlier stages do not propagate over time. Since the framework allows the formation of a new track at any time, new tracks will be formed instead of extending erroneous tracks which are not supported by observations. In next section, the performance of the online version is measured for different window sizes.

2.4 Simulation Results

The unicycle dynamic model is used for our simulations below. The state vector is $x = (x, y, \theta)^T$ where (x, y) is a position of a vehicle in a $\mathbb{R}^2$ plane and θ is a heading of a vehicle. The continuous state equation is

$$\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \varphi \cos\theta \\ \varphi \sin\theta \\ \omega \end{pmatrix}$$

where φ is a directional velocity and ω is an angular velocity. φ and ω are control inputs. If φ and ω are constant over the sampling period T_s, its discretized state equation (with noise) becomes

$$\begin{aligned} x_{t+1} &= x_t + \frac{2}{\omega_t}\varphi_t \sin\left(\frac{\omega_t T_s}{2}\right) \cos\left(\theta_t + \frac{\omega_t T_s}{2}\right) + v_{t,1} \\ y_{t+1} &= y_t + \frac{2}{\omega_t}\varphi_t \sin\left(\frac{\omega_t T_s}{2}\right) \cos\left(\theta_t + \frac{\omega_t T_s}{2}\right) + v_{t,2} \\ \theta_{t+1} &= \theta_t + \omega_t T_s + v_{t,3}, \end{aligned}$$

where $V_t = (v_{t,1}, v_{t,2}, v_{t,3})^T$ are white Gaussian noises.

Now suppose there are more than one objects all moving with the dynamic above and the type of an object can be classified by its directional velocity. So the state equation is

$$X_{t+1}^i = F^i \left(X_t^i = \begin{pmatrix} x_t^i \\ y_t^i \\ \theta_t^i \end{pmatrix}, \varphi^i, \omega_t^i \right) + U_t^i$$

and each φ^i is constant. Let $\varphi = (\varphi^1, \ldots, \varphi^K)$. But we let ω_t^i be a r.v. uniformly distributed over $[-\pi/16, \pi/16]$. So the targets are free to change their heading at any time as they desire and the tracking algorithm is required to distinguish them. A slight modification is required to handle ω_t^i's and it is described in Section 2.4.1. Notice that the transition density function f^i now becomes $f^i(dx_{t+1}|x_t, \varpi_t) = P(X_{t+1}^i \in dx_{t+1}|X_t^i = x_t, \omega_t^i = \varpi_t)$.

We assume that the initial state of a target is uniformly distributed over $\mathcal{R} = [0, L]^2 \times [-\pi, \pi]$. Hence, $p_0^i(x) = \frac{1}{2\pi L^2}$ for each i. We use $b_{n_0} = \frac{n_0}{KT}$ and $d_{n_1} = \frac{5n_1}{KT}$ as the probabilities for split and merge moves. $p_b(\theta, (i,t))$ is assigned such that (i,t) positioned before or after an established track is weighted three times higher than the remaining $(i,t) \in G_1^C$.

We consider a linear observation model

$$Y_t^j = HX_t^i + V_t'^j$$

where

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

for the case when j-th observation comes from an object of type i.

The covariance matrix for V_t^i is

$$\Sigma^i = \begin{bmatrix} (.05T_s\varphi^i)^2 & 0 & 0 \\ 0 & (.05T_s\varphi^i)^2 & 0 \\ 0 & 0 & (.2\frac{\pi}{8})^2 \end{bmatrix}$$

and the covariance matrix for $V_t'^j$ is

$$\Sigma^y = \begin{bmatrix} (.1T_s)^2 & 0 \\ 0 & (.1T_s)^2 \end{bmatrix}$$

and they are held fixed. We have also fixed A such that $A_{ij} \propto 10^{K-d_1(i,j)}$ where d_1 is the Manhattan distance between two states i and j in their binary representations. Hence, $p_{all} = 0$. The false alarms are uniformly distributed over the region $\mathcal{R}^2 = [0, L]^2$ and the number of false alarms has a Poisson distribution with parameter $\lambda V_{\mathcal{R}^2}$. The Dirichlet prior on weights is $\delta = 1$. The detection probability is .95.

Assessing the convergence of MCMC algorithms is usually a difficult task. It is especially challenging in our case since the dimension of the parameter space changes. (Brooks & Giudici, 1998) descibes a method to assess convergence of RJMCMC algorithms by a real-valued function which summerizes the parameters. However such a function is not obvious in our case. Instead, we measure the distance between the true tracks and the estimated tracks since we are working in a simulation

environment. Let $M^*_{1:T}$ and $X^*_{1:T}$ be the parameters of true tracks. Let $\hat{M}_{1:T}$ be the estimate with maximum posterior and let $\hat{X}_{1:T}$ be the estimate of states given $\hat{M}_{1:T}$. We use two metrics:

$$d_m(M^*_{1:T}, \hat{M}_{1:T}) = \frac{1}{KT}\sum_{t=1}^{T}\sum_{i=1}^{K}\mathbb{I}(M_t^{*i} \neq \hat{M}_t^i)$$

to measure the distance between models and

$$d_x(X_{1:T}, M_{1:T}, \hat{X}_{1:T}, \hat{M}_{1:T}) = 1 - \frac{1}{KT}\Big(\sum_{t=1}^{T}\sum_{i=1}^{K}\mathbb{I}(M_t^i = \hat{M}_t^i = 0) + \mathbb{I}(M_t^i = \hat{M}_t^i = 1)e^{-\frac{1}{6}(X_t^i - \hat{X}_t^i)^T\Sigma^{i-1}(X_t^i - \hat{X}_t^i)}\Big)$$

to measure the distance between the models and states simultaneously. Here $\mathbb{I}(\cdot)$ is an indicator function. We used the first two components of X_t^i to evaluate d_x in experiments below.

2.4.1 DBMS RJMCMC for Discrete-Time Unicycle Dynamics

In order to estimate the input control ω_t^i, we perform the following move in every MCMC step

(f') update ω_t^i at randomly selected (i,t).

The move is performed along with moves (a)-(c).

For move (f'), we first choose (i,t) randomly from G_2 where

$$G_2 = \{(i,t) : M_t^i = 1, M_{t+1}^i = 1, 1 \leq i \leq K, 1 \leq t \leq T-1\}$$

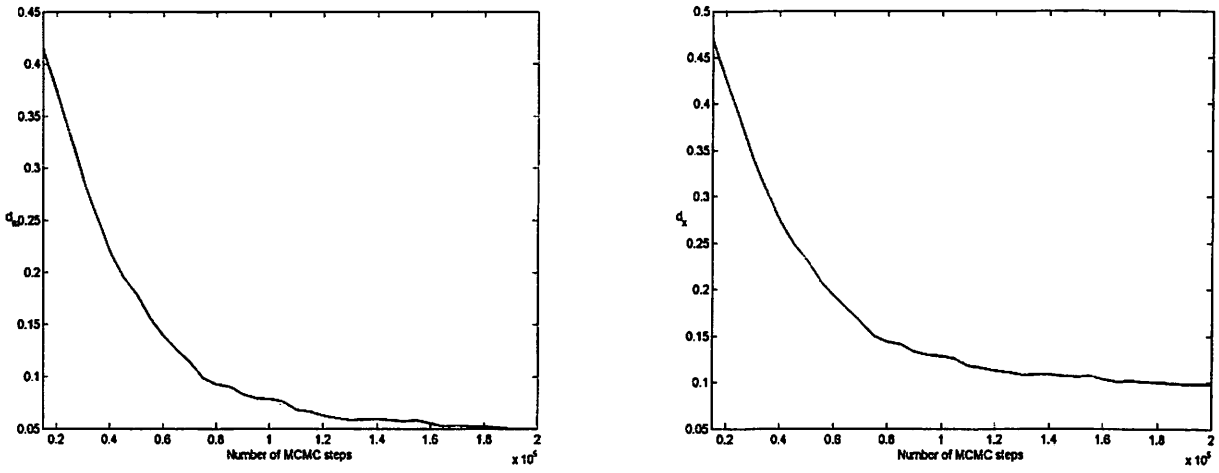
and propose $\omega_t'^i \sim \mathcal{U}[-\pi/16, \pi/16]$. Note that $\mathcal{U}[a,b]$ is the uniform distribution over $[a,b]$. Then we accept the proposal with probability $\min(1,R)$ where

$$R = \frac{f^i(X_{t+1}|X_t, \omega_t'^i)}{f^i(X_{t+1}|X_t, \omega_t^i)}.$$

The track update moves are unchanged except the change in the transition density function. In track split move, (i,t) is chosen as described in Section 2.3.3.3 and the proposals of η_1 and Z_t' are the same. The proposal of $X_t'^i$ is the same in forward split and backward split moves. In placement split, the first two components of $X_t'^i$ are proposed as done in Section 2.3.3.3 and the third component of $X_t'^i$ is proposed from $\mathcal{U}[-\pi,\pi]$. Lastly, we propose the control $\omega_t'^i \sim \mathcal{U}[-\pi/16, \pi/16]$. The acceptance ratio calculation is the same as described in Section 2.3.3.3 but a special care is required to make sure that the third components of X_t^i are kept inside $[-\pi,\pi]$ for correct estimate calculations.

2.4.2 Experiment I (Convergence)

We set $K = 3$, $T = 20$, $L = 100$, $T_s = 1$, $\lambda V_{\mathcal{R}^2} = 1$ and $\varphi = (2,4,6)$. We generated ten random scenarios then ran each scenario ten times (first 10,000 samples are used as burn-ins). All observations are assigned to clutter at the initial state of the sampler. Figure 2.3 shows d_m and d_x averaged over 100 runs against the number of MCMC samples. Any tracks with length less than three time steps are discarded from the estimates. We note that the metric d_x is very conservative and visual inspection shows that any distance less than .05 is almost a perfect match.



(a) (b)

Figure 2.3: Experiment I (Convergence): (a) d_m vs. number of MCMC samples; (b) d_x vs. number of MCMC samples

2.4.3 Experiment II - DBMS vs. Optimal Linear Filter

We have compared the performance of our algorithm against the optimal linear filter using Kalman filters on the same scenarios used in Experiment I. The multiple tracking algorithm such as MHT uses Kalman filters, hence, if the observations are correctly partitioned, i.e., when the data association is correct, MHT will give the optimal estimates according to the linear dynamics. For each scenario, we first partitioned the observations into the original tracks from which the scenario was generated. We then run Kalman filters on each track and compared d_x estimated from Kalman filters against the estimates from DBMS. For the Kalman filters, we use the usual linear model for tracking (Li & Jilkov, 2000). The state vector is $x = [x, y, \dot{x}, \dot{y}]^T$ where (x, y) is a position on a plane and $(\dot{x}, \dot{y})$ is a velocity vector. Let δ be the sampling interval. Then the dynamic and measurement models are

$$\begin{aligned} x_{t+1} &= Ax_t + Gw_t \\ y_t &= Cx_t + v_t, \end{aligned}$$

where

$$A = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix}$$

and

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Here w_t and v_t are Gaussian noise process with zero mean and convariance Q and R, respectively. We used $Q = \alpha m_x \times \text{diag}(1,1,1,1)$ and $R = \alpha m_y \times \text{diag}(1,1)$, where $m_x = \max_{i,j,k}(\Sigma^i_{j,k})$ and $m_y = \max_{j,k}(\Sigma^y_{j,k})$. We vary α to check if the estimates are influenced by the changes in covariance.

Table 2.1: DBMS vs. Optimal Linear Filter

Test Case	DBMS d_x	Optimal Linear Filter d_x ($\alpha = 1$)	Optimal Linear Filter d_x ($\alpha = 5$)
1	0.0937	0.0832	0.0836
2	**0.1260**	0.1564	0.1569
3	0.1010	0.0728	0.0740
4	0.0977	0.0690	0.0683
5	0.0870	0.0828	0.0832
6	**0.1069**	0.1336	0.1337
7	**0.0912**	0.1202	0.1198
8	**0.0430**	0.0432	0.0445
9	**0.0638**	0.0875	0.0823
10	0.1333	0.0989	0.0966

Table 2.1 summarizes the results. Note that the observations are not partitioned for DBMS and the estimates for DBMS are averaged over ten repeated runs. The estimates from the optimal linear filter are not influenced by the changes in covariance. In some cases, DBMS outperforms the optimal linear filter (shown in boldface, Table 2.1). This means, in these cases, DBMS outperforms any linear flitering algorithm even if the linear filtering algorithm is given perfect information about data association. Since we can directly apply the nonlinear dynamics with DBMS, it does not suffer from the approximation error of the linear filters. Figure 2.4 shows an example of such estimation error. In addition, MHT is unlikely to achieve the values listed in Table 2.1, since it is not possible to have perfect associations in all cases due to hueristics required for MHT such as gating, N-scan-back and pruning.

2.4.4 Experiment III (Model Complexity)

We measure the performance of the algorithm against the model complexity. The setup is the same as Experiment I except we vary K. We generated one scenario for each $K = 3,4,5,6,7$, where $\varphi = (2,3,4,5,6)$ and $\Sigma^i = \text{diag}((.1T_s)^2, (.1T_s)^2, (.2\frac{\pi}{8})^2)$. One of the scenarios and the estimated tracks are shown in Figure 2.6. Each scenario is run for five times (first 20,000 samples are used as burn-ins). The average d_x and d_m are shown in Figure 2.5. The convergence rate for a higher order model is slower than a lower order model but the figures also suggest that the algorithm scales well with complex models. The curves are not smooth as Figure 2.3 since a single scenario is used for each K.

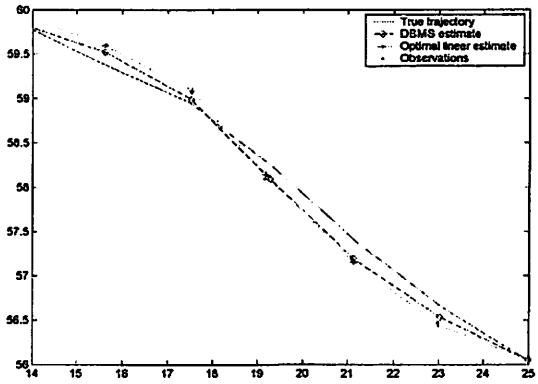
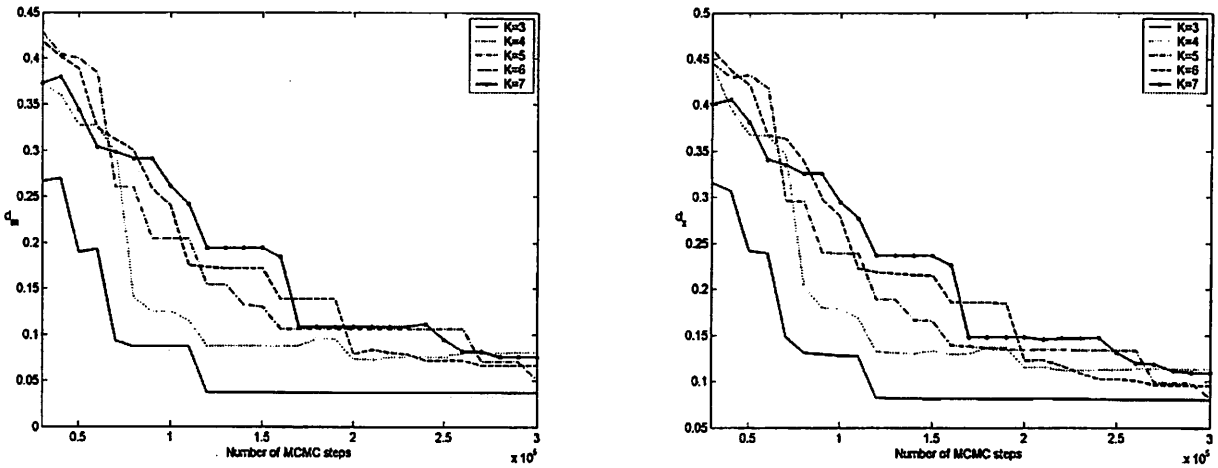


Figure 2.4: An example of the inaccuracy of the optimal linear filter

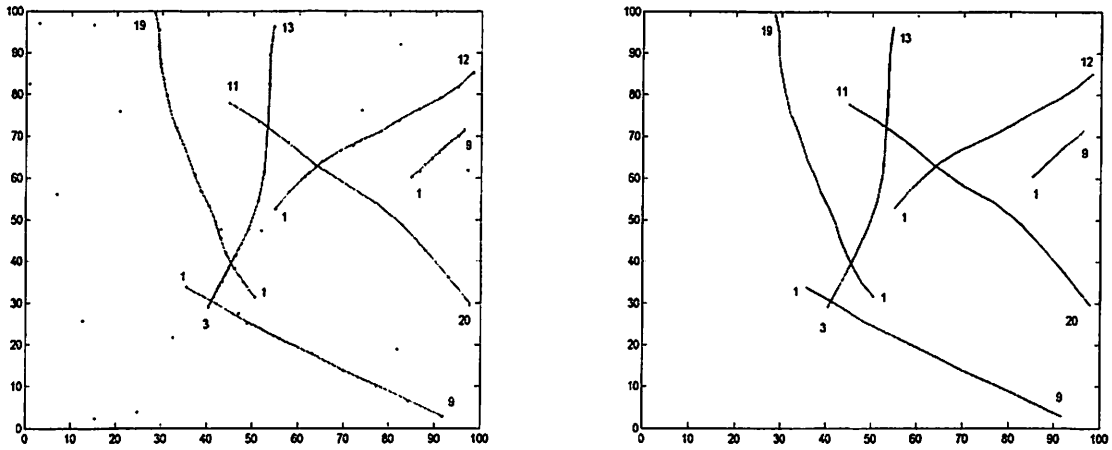


(a) (b)

Figure 2.5: Experiment II (Model Complexity): (a) d_m vs. number of MCMC samples; (b) d_x vs. number of MCMC samples

2.4.5 Experiment IV (False Alarms)

We apply different levels of false alarm rates to assess the robustness of the algorithm against outliers. The setup is the same as Experiment I but we vary $\lambda V_{\mathcal{R}^2}$ from 1 to 10. A set of random tracks was generated first and then different false alarm rates were applied on this set of tracks to generate ten different scenarios. See Figure 2.7. We ran each scenario 10 times (first 20,000 samples are used as burn-ins). Figure 2.8 shows that our algorithm is extemely robust against false alarms (results for $\lambda V_{\mathcal{R}^2} = 2, 4, 6, 10$ are shown).



(a) (b)

Figure 2.6: A scenario used in Experiment III (Model Complexity) $K = 7$: (a) True tracks (solid line) and observations (dots) - a pair of numbers next to each track indicates the appearance and disappearance time of the track; (b) An example of estimated tracks for (a)

2.4.6 Experiment IV (Online DBMS RJMCMC)

The setting is the same as Experiment (I) except $T = 500$, $L = 250$ and $\varphi = (4, 6, 8)$. A single scenario was randomly generated (the accumulated observations and true tracks are shown in Figure 2.9 along with an example of estimated tracks). Three different window sizes are considered $T_w = 10, 20, 30$ and windows are forwarded by a single time step. At each online step, the algorithm is run for a fixed number of MCMC samples. For each window size, we have tried 3 different numbers of MCMC samples, 2000, 4000 and 8000. So there are 9 cases and we ran each case 4 times. The average performance is shown in Table 2.2 [1]. The performance gets better as we increase the window size and number of MCMC samples but with expense of increasing execution time. It shows that the algorithm is well suited for a real-time surveillance system.

2.5 Conclusions

We have presented a dynamic Bayesian model selection framework for layered dynamic systems in which the model selection problems are solved sequentially. An efficient algorithm based on reversible jump Markov chain Monte Carlo is described and extended to online computations. The multitarget tracking problem is formulated as an instance of the dynamic Bayesian model selection problem. We have shown that the algorithm is very robust against outliers and scales well with model complexity in simulation. In some cases, when we apply DBMS to nonlinear dynamics, DBMS outperforms any linear filtering algorithm with perfect associations. This framework can

[1] The algorithm is written in Matlab and run on PC with a 2-GHz Intel Pentium 4 processor.

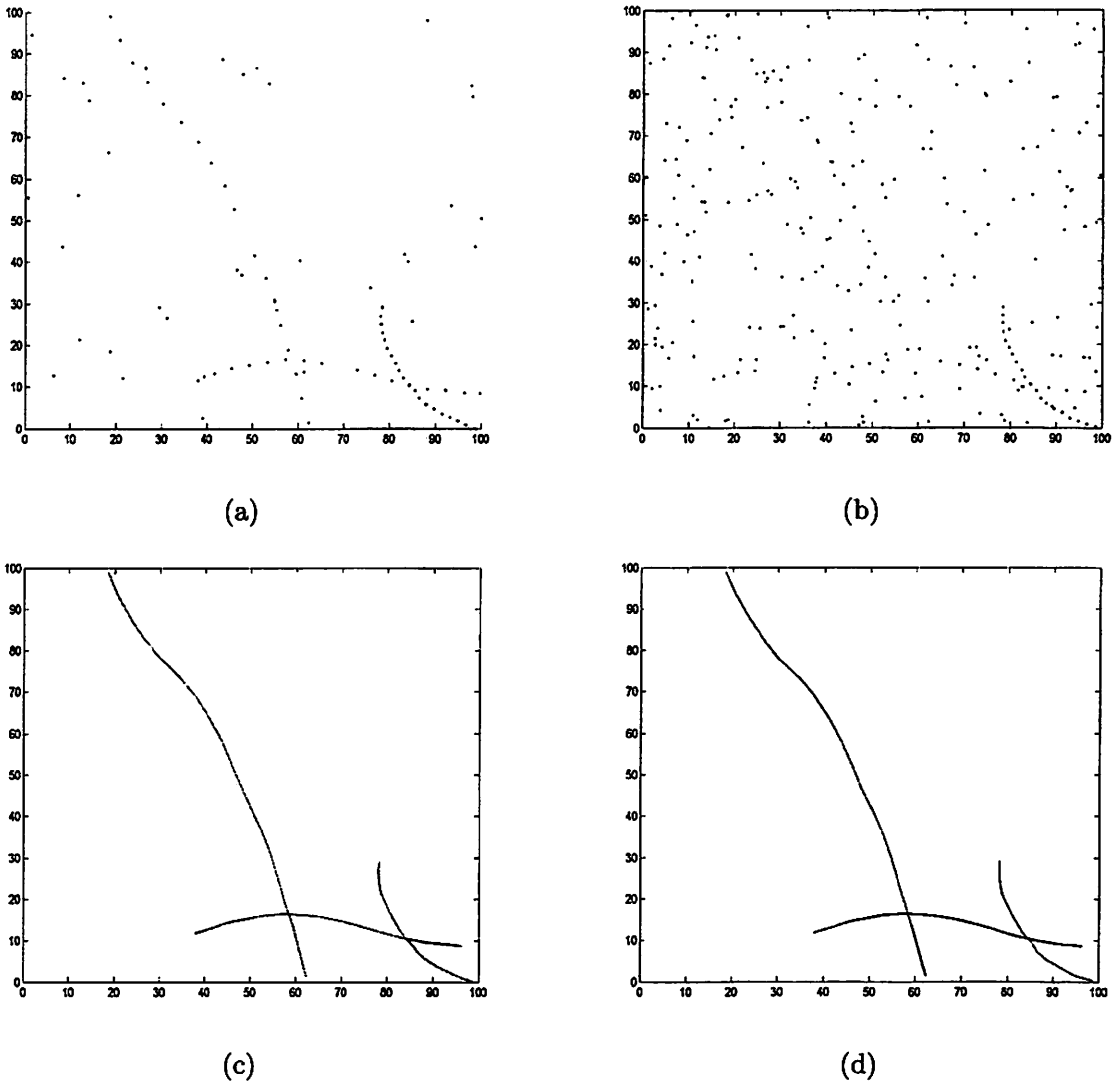
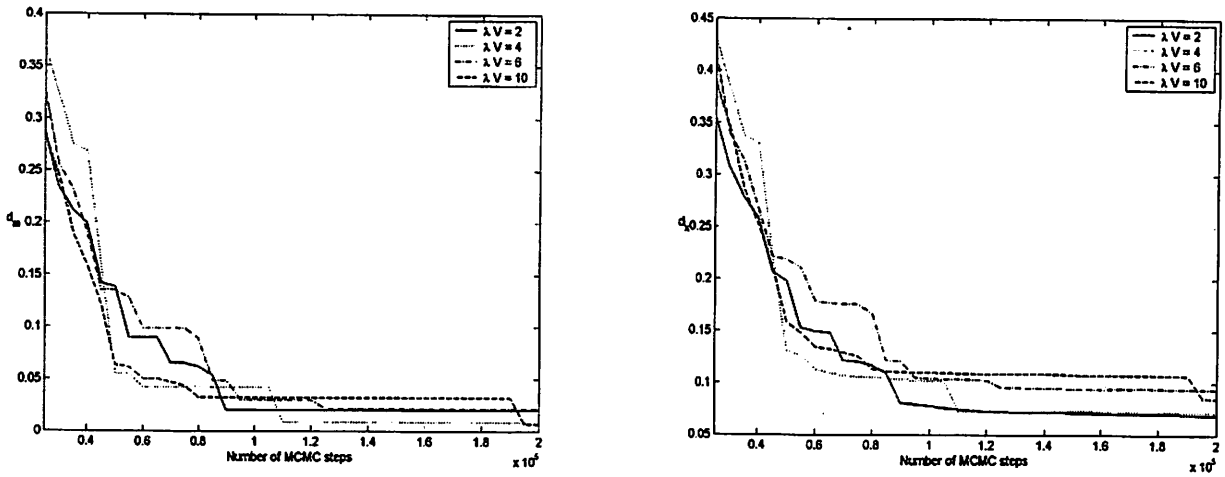


Figure 2.7: Experiment III (False Alarms): (a) Accumulated observations with $\lambda V_{\mathcal{R}} = 2$; (b) Accumulated observations with $\lambda V_{\mathcal{R}} = 10$; (c) True tracks; (d) An example of estimated tracks for $\lambda V_{\mathcal{R}} = 10$

be easily generalized to other applications such as signal processing and computer vision by using it as a general dynamic pattern recognizer.



(a) (b)

Figure 2.8: Experiment III (False Alarms): (a) d_m vs. number of MCMC samples; (b) d_x vs. number of MCMC samples

Table 2.2: Performance of Online DBMS RJMCMC

T_w	MCMC samples (burn-ins)	d_x	d_m	sec / time step
10	2000 (1000)	0.0544	0.0472	6.30
	4000 (2000)	0.0442	0.0372	12.47
	8000 (4000)	0.0332	0.0263	24.61
20	2000 (1000)	0.0658	0.0582	6.84
	4000 (2000)	0.0362	0.0267	13.53
	8000 (4000)	0.0338	0.0258	26.98
30	2000 (1000)	0.0823	0.0727	7.37
	4000 (2000)	0.0347	0.0250	14.62
	8000 (4000)	0.0241	0.0152	28.94

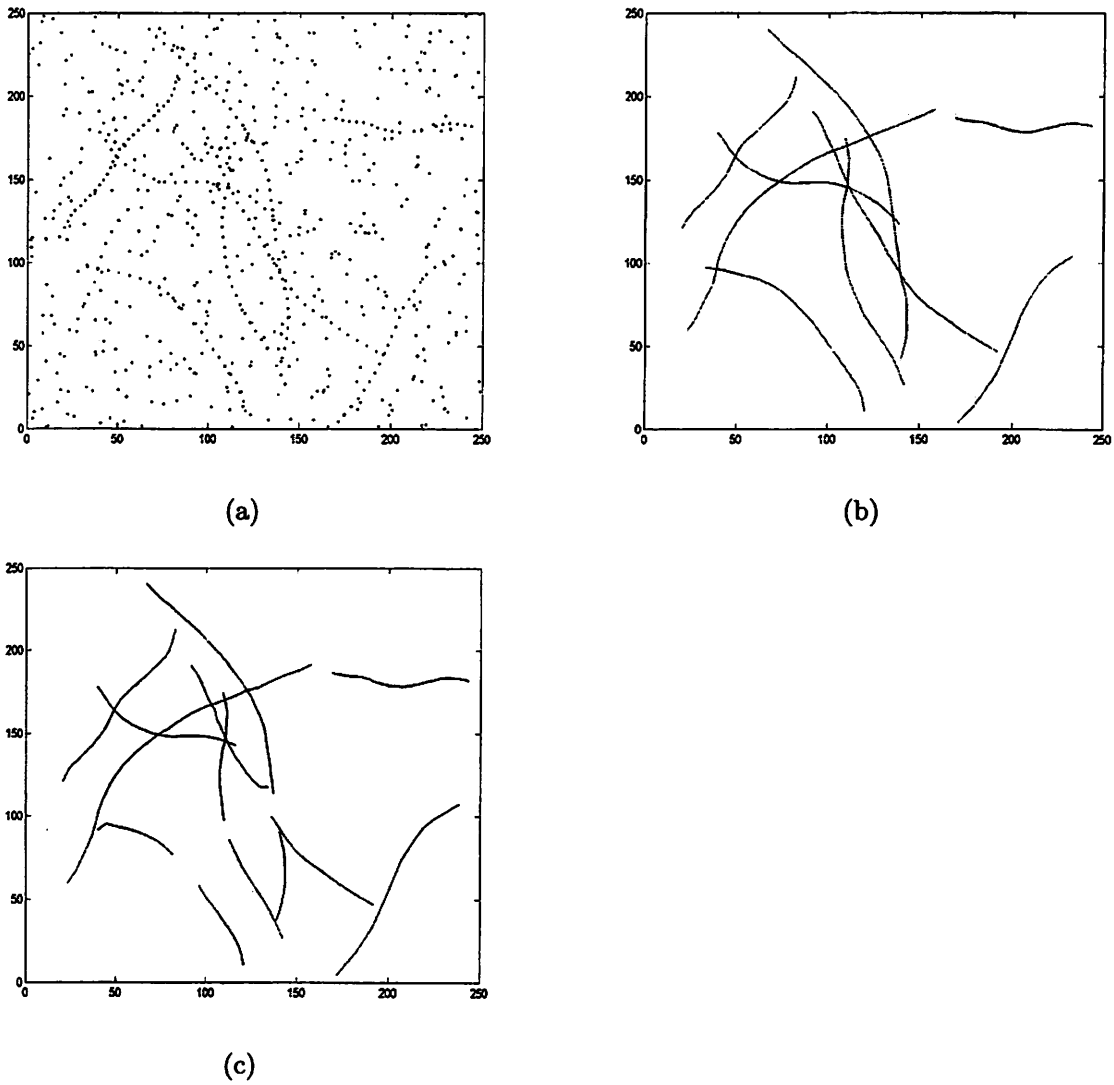


Figure 2.9: (a) Accumulated observations; (b) True tracks; (c) Estimated tracks

Chapter 3

MCMC Data Association for General Multiple Target Tracking Problems

3.1 Introduction

The multiple target tracking plays an important role in many areas of engineering such as surveillance, computer vision, and signal processing (Bar-Shalom & Fortmann, 1988; Cox, 1993). Under the most general setup, a varying number of targets are moving around in a region with continuous motions and the positions of moving targets are sampled at random intervals. The measurements about the positions are noisy, with detection probability less than one, and there is a noise background of spurious position reports (false alarms). Targets arise at random in space and time. Each target persists independently for a random length of time and ceases to exist. A track of a target is defined as a path in space-time traveled by the target. The seminal paper by Sittler (Sittler, 1964) introduced the major concepts about multiple target tracking and a method to evaluate tracks. He pointed out two major problems in multiple target tracking: data association and state estimation. The essence of the multiple target tracking problem is to find tracks from the noisy observations and it requires solutions to both data association and state estimation problems.

In (Sittler, 1964), the data association problem in multiple target tracking is described as a problem of finding a partition of observations such that each element of a partition is a collection of observations generated by a single target or clutter. However, due to the noises in state transition and observation, we cannot expect to find the exact solution. Hence, Sittler developed a probabilistic surveillance model and searched for a partition of observations on which the likelihood function defined by the surveillance model is maximized. This data-oriented view of data association has been applied and extended by many authors (Morefield, 1971; Stein & Blackman, 1975; Reid, 1979; Kurien, 1990; Cox & Hingorani, 1994; Poore, 1995). The most successful multiple target tracking algorithm based on this view is the multiple hypothesis tracker (MHT) (Reid, 1979). A different approach to the data association problem is the probabilistic data association filter (PDAF) (Bar-Shalom & Fortmann, 1988). Under the optimal Bayesian setting, given a fixed number of targets, all possible associations between the known targets and observations from the initial to the present time are weighted by their likelihoods. These weights are called the association weights or

probabilities. For each combination, the state of a target is estimated by a filtering algorithm and this conditional expectation is weighted by the association weight. Then the state of each target is estimated by summing over the weighted conditional expectations. However, the enumeration of all possible associations is not practical so the suboptimal single-stage data association approach, such as the joint probabilistic data association filter (JPDAF), has been applied in practice (Bar-Shalom & Fortmann, 1988). This Bayesian approach to data association has been applied by many authors to different applications (Bar-Shalom & Fortmann, 1988; Streit & Luginbuhl, 1994; Huang & Russell, 1997; Pasula et al., 1999; Schulz et al., 2001; Russell, 2001).

In MHT, each hypothesis associates past observations with a target such that an observation is not shared by more than one target. As a new set of observations arrives, a new set of hypotheses is formed from the previous hypotheses. The construction of new hypotheses requires the enumeration of all possibilities and the size of hypotheses grows exponentially. At each time step, each hypothesis is scored by the probability of having the hypothesis given the observations, i.e., the posterior of the hypothesis. The algorithm returns the hypothesis with the highest score as a solution. MHT is categorized as a deferred logic (Poore, 1995) in which the decision about forming a new track or removing an existing track is delayed until enough observations are collected. Hence, MHT is capable of initiating and terminating a varying number of targets and suitable for surveillance applications in which the tracker is required to initiate and terminate a varying number of tracks autonomously. However, the size of the hypotheses grows exponentially and the enumeration of all hypotheses is not practical. The initial implementation and later extensions proposed several heuristics, such as pruning, gating, clustering and N-scan-back logic, to reduce the complexity of the problem (Reid, 1979; Kurien, 1990). However, the hueristics are used at the expense of the optimality and the algorithm can still suffer in a dense environment. Furthermore, the running time at each step of the algorithm cannot be bounded easily, making it difficult to be deployed in a real-time surveillance. As a method of pruning, an efficient method of finding k-best hypothesis based on the algorithm by Murty (Murty, 1968) is developed in (Cox & Hingorani, 1994).

As opposed to finding the optimal association, JPDAF computes the weights of all the possible associations from the *latest* set of observations to the known tracks and clutter (Bar-Shalom & Fortmann, 1988). Given an association, the state of a target is estimated by a filtering algorithm and this conditional expectation of state is weighted by the association weight. Then the state of a target is estimated by summing over the weighted conditional expectations. JPDAF is a sequential tracker in which the associations between the known targets and the latest observations are made sequentially and the associations made in the past are not reversible (Poore, 1995). The sequential trackers are more efficient than deferred logic trackers such as MHT but they are prone to make erroneous associations (Poore, 1995). At each stage, the association between previously estimated states of targets and the latest observations is optimal from the Bayesian point of view. But the algorithm is suboptimal since the states estimated at each step of algorithm can be different from the states estimated from all observations from the initial to current time. In addition, it is inferred that the exact calculation at each stage is NP-hard (Collins & Uhlmann, 1992) since the related problem of finding the permanent of a 0-1 maxtix is #P-complete (Valiant, 1979). In (Huang & Russell, 1997), a single-stage data association problem is considered and a leave-one-out hueristic is developed to avoid the enumeration of all possible associations. Later, the approach is extended to a multi-stage data association problem using Markov chain Monte Carlo (Pasula et al., 1999).

Since only the current set of observations are considered in JPDAF, it cannot initiate or terminate tracks. Also JPDAF assumes a fixed number of targets and requires a good initial state for each target. There are restricted extensions to JPDAF to allow the formation of a new track. See (Cox, 1993) for references. Instead of keeping a single Gaussian component for each target, the multisensor multitarget mixture reduction (MTMR) maintains a fixed number of Gaussian components for each target to prevent the loss of information when there are several significant and well spaced componenets (Pao, 1993). It has been shown that MTMR performs better than JPDAF but at the expense of computation (Pao, 1993). But MTMR also assumes a fixed number of targets and requires a good set of initial states. The probabilistic multi-hypothesis tracking (PMHT) uses probabilistic associations between observations and targets to avoid the maintenance of a hypothesis tree and the enumeration over all possible associations (Streit & Luginbuhl, 1994). PMHT allows an association between a single track and an arbitrary number of observations. This assumption may not represent the physical reality in many cases but reduces the complexity of the problem (Hue et al., 2002). However, a fixed known number of targets is assumed and the track initiation and termination are difficult under PMHT. A hypothesis test is presented in (Hue et al., 2002) as a method to allow a varying number of tracks but the paper also addresses the difficulty with estimating the initial states of new targets.

The data association problem of multiple target tracking formulated under the data-oriented view is also known to be NP-hard (Poore, 1995). It is a multidimensional assignment problem (Poore, 1995) and the multidimensional assignment problem is NP-hard since its special case 3-dimensional matching is NP-hard (Papadimitriou & Steiglitz, 1982). There is a polynomial time algorithm for the weighted bipartite matching problem, i.e., 2-dimensional matching problem, but the 3-dimensional matching problem is NP-hard. Hence, we can find an optimal solution to a single-stage data association problem in a polynomial time but cannot expect to find an optimal solution to a multiple-stage data association problem unless P = NP. The multiple target tracking problem was first formulated as a combinatorial optimization problem in (Sittler, 1964). Later, heuristics are added to speed up the algorithm (Reid, 1979; Kurien, 1990; Cox & Hingorani, 1994). An optimization approach to the data association has been applied as a 0-1 integer programming problem (Morefield, 1971) and as a multidimensional assignment problem (Poore, 1995). In both cases one needs to find a feasible set of tracks from all possible tracks, i.e., all possible partitions of observations, to prevent the exponential explosion and compute the cost of each feasible track, such as the negative log likelihood. Then the optimization routine finds a subset from the feasible tracks such that the combined costs are minimized while satisfying the constraints, i.e., each track has at most one observation at each time and no two tracks share the same observation. The gating method similar to the ones described in (Stein & Blackman, 1975; Reid, 1979) is used to find a feasible set of tracks. However, in a dense environment, the size of the feasible tracks can be very large and the complexity of the optimization routine increases dramatically, since the number of parameters in the optimization routine depends on the number of feasible tracks.

Our main objective in this chapter is to develop an efficient real-time algorithm that solves the data association problem and is capable of initiating and terminating a varying number of tracks. We take the data-oriented, combinatorial optimization approach to the data association problems but avoid the enumeration of tracks by applying a sampling method called Markov chain Monte Carlo (MCMC). The immediate benefit of using MCMC is the low memory requirement as discussed below. The MCMC data association algorithm shows a remarkable performance under the extreme

conditions such as a large number of targets in a dense environment, low detection probabilities, and a large number of false alarms. The MCMC data association algorithm can be considered as a deferred logic since its decision about forming a track is based on the current and past observations. But, at the same time, it can be considered as an approximation to the optimal Bayesian filter if it is used to approximate the association probabilities or expectations such as the average link travel time as done in (Pasula et al., 1999).

The remainder of this chapter is structured as follows. We formally state the (discrete-time) general multiple target tracking problem in Section 3.2. In Section 3.3, we develop a general purpose MCMC data association algorithm for multiple target tracking. The algorithm is applied in simulations to very extreme situations and its performance is compared with the greedy algorithm and MHT in Section 3.5.

3.2 Problem Formulation

Problem 3.1 (General Multiple Target Tracking Problem (Discrete-Time)) Let $T \in \mathbb{Z}^+$ be the duration of surveillance. Let K be the unknown number of objects moving around the surveillance region $\mathcal{R}$ for some duration $[t_i^k, t_f^k] \subset [1, T]$ for $k = 1, \ldots, K$. Let V be the volume of $\mathcal{R}$. Each object arises at a random position in $\mathcal{R}$ at t_i^k, moves independently around $\mathcal{R}$ until t_f^k and disappears. The number of objects arising at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $(\lambda_b V)$ where λ_b is the birth rate of new objects per unit volume for the duration of surveillance. The initial position of a new object is uniformly distributed over $\mathcal{R}$.

Let $F^k : \mathbb{R}^d \to \mathbb{R}^d$ be the discrete-time dynamics of the object k, where d is the dimension of the state variable, and let $x_t^k \in \mathbb{R}^d$ be the state of the object k at time t for $k = 1, 2, \ldots, K$. The object k moves according to

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

where $w_t^k \in \mathbb{R}^d$ are white noise processes. Note that F^k can be the same for all or some of the objects. The noisy observation about the state of the object is measured with the detection probability p_d which is less than unity. There are also false alarms and the number of false alarms has a Poisson distribution with a parameter $(\lambda_f V)$ where λ_f is the false alarm rate per unit time, per unit volume. Let n_t be the number of observations at time t which includes both noisy observations and false alarms. Let $y_t^j \in \mathbb{R}^m$ be the j-th observation at time t for $j = 1, \ldots, n_t$, where m is the dimensionality of each observation vector. Each object generates a unique observation at each sampling time if it is detected. Let $H^j : \mathbb{R}^d \to \mathbb{R}^m$ be the observation model. Then the observations are generated as follows:

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

where $v_t^j \in \mathbb{R}^m$ are white noise processes and $u_t \sim \text{Unif}(\mathcal{R})$ is a noise process for false alarms. Notice the with probability $1 - p_d$ the object is not detected and we call this a missing observation.

Under the data-oriented approach, the multiple target tracking problem is to partition the observations such that the posterior is maximized. The posterior function is described in (3.12). Under

the Bayesian approach, if we are given a function defined on Ω, the collection of all partitions of observations (see below for its definition), we seek the expected value of the function given the observations. ■

Remark 3.1 In the description of Problem 3.1, we assume that the detection probability is independent of states of targets and sensors in order to simplify the posterior function (3.12). This may not model the physical world accurately but it has been generally accepted and applied in practice. However, it is easy to extend such that the detection probability depends on the type of object being tracked (Cox, 1993).

Remark 3.2 Under the data-oriented approach, we seek for a single partition of observations which maximizes the posterior, i.e., the maximum a posteriori (MAP) estimate. This MAP solution may not be robust in the Bayesian sense. But it is convenient if we want to estimate parameters whose dimension is dependent on the number of tracks, such as the states of targets. Since the size of $x_t = ({x_t^1}^T, \ldots, {x_t^K}^T)^T$ depends on the number of tracks K, the estimation of x_t without fixing the number of tracks is not meaningful. Hence, under the Bayesian approach, if a single set of state estimation is required, we might first estimate the most likely number of targets and then estimate the expected values of states given the number of targets. So when a single set of state estimation is required, it is sometimes more convenient to use the data-oriented approach. However, we can easily estimate the association probabilities, average number of targets, and average travel time using the proposed algorithm.

Remark 3.3 In multiple target tracking, it is generally assumed that targets are indistinguishable and we take the same assumption in this paper. However, if observations include target type or attribute information, they can be easily incorporated into our observation model. But we may required to use a Bayesian filtering algorithm if we cannot apply Kalman filters.

Let us first specify the dynamic and measurement models. Here we use the usual linear system model but the method can be easily extended to non-linear models coupled with a non-linear regression. If an object is observed k times at $t_1, t_2, \ldots, t_k$, its dynamic and measurement models can be expressed as:

$$\begin{aligned} x_{t_{i+1}} &= A(t_{i+1} - t_i)x_{t_i} + G(t_{i+1} - t_i)w_{t_i} \qquad \text{for } i = 1, \ldots, k-1 \\ y_{t_i} &= Cx_{t_i} + v_{t_i} \qquad \text{for } i = 1, \ldots, k, \end{aligned} \tag{3.3}$$

where w_{t_i} and v_{t_i} are white Gaussian noises with zero mean and covariance Q and R, respectively. $A(\cdot)$, $G(\cdot)$, and C are matrices with appropriate sizes. The entries of the matrix $A(t_{i+1} - t_i)$ and $G(t_{i+1} - t_i)$ are determined by the sampling interval $t_{i+1} - t_i$ for each i. For clarity, the subsequence notation for the time index is suppressed for now. Let us define

$$\begin{aligned} \bar{x}_{t+1} &= \mathbb{E}\left(x_{t+1} | y_1, \ldots, y_t\right), \\ \bar{P}_{t+1} &= \mathbb{E}\left((x_{t+1} - \bar{x}_{t+1})(x_{t+1} - \bar{x}_{t+1})^T | y_1, \ldots, y_t\right), \\ \hat{x}_t &= \mathbb{E}\left(x_t | y_1, \ldots, y_t\right), \quad \text{and} \\ \hat{P}_t &= \mathbb{E}\left((x_t - \hat{x}_t)(x_t - \hat{x}_t)^T | y_1, \ldots, y_t\right). \end{aligned}$$

Then applying the Kalman filter, the update equations are

$$\begin{aligned} \bar{x}_{t+1} &= A\hat{x}_t \\ \bar{P}_{t+1} &= A\hat{P}_t A^T + GQG^T \end{aligned} \tag{3.4}$$

and the measurement update equations are

$$\begin{aligned} \hat{x}_t &= \bar{x}_t + K_t(y_t - C\bar{x}_t) \\ \hat{P}_t &= \bar{P}_t - \bar{P}_t C^T B_t^{-1} C\bar{P}_t, \end{aligned} \tag{3.5}$$

where $K_t = \hat{P}_t C^T R^{-1}$ is the Kalman gain matrix and $B_t = C\bar{P}_t C^T + R$.

Let $y_t = \{y_t^j : j = 1, \ldots, n_t\}$ and $Y = \bigcup_{t\in\{1,\ldots,T\}} y_t$. Let Ω be a collection of partitions of Y such that, for $\omega \in \Omega$,

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

As defined in Problem 3.1, K is the number of tracks for the given partition $\omega \in \Omega$. We call τ_k a track when there is no confusion although the actual track is the set of estimated states from the observations τ_k. However, we assume there is a deterministic function that returns a set of estimated states given the set of observation, so no distinction is required. We denote by $\tau_k(t)$ the observation at time t that is assigned to the track τ_k. Notice that $\tau_k(t)$ can be empty if it is a missing observation or if t is before the appearance time or after the disappearance time of the associated target. The fourth requirement says that a track can have at most one observation at each time, but, in the case of multiple sensors, we can easily relax this requirement to allow multiple observations per track. A track is assumed to contain at least two observations since we cannot distinguish a track with a single observation from a false alarm. Let $n_d(t)$ be the number of detected objects at time t, i.e., $n_d(t) = |\{\tau_k(t) : 1 \leq k \leq K\}|$, and let $n_d = \sum_{t=1}^{T} n_d(t)$ be the total number of detections. Similarly, let $n_f(t)$ be the number of false alarms at time t, i.e., $n_f(t) = |\tau_0(t)|$, and let $n_f = \sum_{t=1}^{T} n_f(t)$ be the total number of false alarms.

In (Pasula et al., 1999), the exchangeability of $\omega \in \Omega$ is assumed and a uniform prior on ω is used. However, it is no longer true when there are a varying number of objects with false alarms and missing observations. We instead assume the exchangeability among ω's with the same number of tracks, false alarms and missing observations as done in the identity uncertainty problems (Russell, 2001; Pasula et al., 2003). First, we can expand the prior $P(\omega)$ as

$$\begin{aligned} P(\omega) &= P(\omega, n_d, n_f, K) \\ &= P(\omega | n_d, n_f, K) P(n_d, n_f | K) P(K), \end{aligned} \tag{3.6}$$

since n_d, n_f, and K are completely known given $\omega \in \Omega$. Recall that the number of objects has a Poisson distribution with a parameter $(\lambda_b V)$ so

$$P(K) = \frac{(\lambda_b V)^K}{K!} e^{-\lambda_b V}. \tag{3.7}$$

One the other hand, since false alarms are independent from the tracks,

$$P(n_d, n_f | K) = \prod_{t=1}^{T} \binom{K}{n_d(t)} p_d^{n_d(t)} (1-p_d)^{K-n_d(t)} \frac{(\lambda_f V)^{n_f(t)}}{n_f(t)!} e^{-\lambda_f V}. \tag{3.8}$$

By the exchangeability, we also have

$$P(\omega | n_d, n_f, K) = \prod_{t=1}^{T} \left[\binom{n_t}{n_d(t)} \frac{K!}{(K-n_d(t))!} \right]^{-1}. \tag{3.9}$$

Hence, after plugging (3.7),(3.8), and (3.9) into (3.6), we have

$$P(\omega) = \left(\prod_{t=1}^{T} \frac{1}{n_t!} p_d^{n_d(t)} (1-p_d)^{K-n_d(t)} (\lambda_f V)^{n_f(t)} \right) \frac{(\lambda_b V)^K}{K!} e^{-V(\lambda_f T + \lambda_b)}. \tag{3.10}$$

Once a partition $\omega \in \Omega$ is chosen, the tracks $\tau_1, \ldots, \tau_K \in \omega$ and a set of false alarms $\tau_0 \in \omega$ are completely determined. Hence, for each track, we can estimate the states of an object independently, since each object moves independently from the other objects. For each track $\tau \in \omega$, we apply the Kalman filter (3.4), (3.5) to estimate the states $\hat{x}_t(\tau)$ and covariances $B_t(\tau)$, where $\hat{x}_t(\tau)$ is the estimated state of the track τ at time t and $B_t(\tau) = C\bar{P}_{t_i}(\tau)C^T + R$ is the conditioned observation covariance for the track τ. So the likelihood becomes

$$P(Y|\omega) = \left(\frac{1}{V}\right)^{n_f} \prod_{\tau \in \omega \setminus \{\tau_0\}} \left(\frac{1}{V} \prod_{i=1}^{|\tau|-1} \mathcal{N}(\tau(t_{i+1}) - A(t_{i+1} - t_i)\hat{x}_{t_i}(\tau), B_{t_{i+1}}(\tau)) \right), \tag{3.11}$$

where $\mathcal{N}(\mu, \Sigma)$ is the Gaussian density function with mean μ and covariance matrix Σ and t_i in $\tau(t_i)$ is the time at which the object associated with the track τ is observed i times.

Let $n_u = \sum_{t=1}^{T} (K - n_d(t))$ be the total number of missing observations. Then the posterior of ω becomes

$$\begin{aligned} P(\omega|Y) &= \frac{1}{Z} \frac{1}{K!} p_d^{n_d} (1-p_d)^{n_u} \lambda_f^{n_f} \lambda_b^K \\ &\times \prod_{\tau \in \omega \setminus \{\tau_0\}} \prod_{i=1}^{|\tau|-1} \mathcal{N}(\tau(t_{i+1}) - A(t_{i+1} - t_i)\hat{x}_{t_i}(\tau), B_{t_{i+1}}(\tau)), \end{aligned} \tag{3.12}$$

where Z is a normalizing constant. Now under the data-oriented, combinatorial optimization approach, the goal of Problem 3.1 is to find a partition of observations such that $P(\omega|Y)$ is maximized.

3.3 MCMC Data Association Algorithm

In this section we develop an MCMC sampler to solve Problem 3.1. Solving complex problems by sampling methods such as Markov chain Monte Carlo (MCMC) has become more tractable, due the increased computational power. MCMC-based algorithms play a significant role in many fields such as physics, statistics, economics, and engineering (Beichl & Sullivan, 2000). In some cases, MCMC is the only known general algorithm which finds a good solution to a complex problem in a polynomial time (Jerrum & Sinclair, 1996). MCMC techniques have been applied to the complex probability distribution integration problems, counting problems such as #P-complete problems, and combinatorial optimization problems (Jerrum & Sinclair, 1996; Beichl & Sullivan, 2000). The MCMC approach applied to the combinatorial optimization problems is generally known as simulated annealing.

The set Ω becomes a state space of the MCMC sampler and we sample from Ω such that its stationary distribution is $P(\omega|Y)$. If we are at state $\omega \in \Omega$, we propose $\omega' \in \Omega$ following the proposal distribution $q(\omega, \omega')$. The move is accepted with an acceptance probability $A(\omega, \omega')$ where

$$A(\omega, \omega') = \min\left(1, \frac{P(\omega'|Y)q(\omega', \omega)}{P(\omega|Y)q(\omega, \omega')}\right), \tag{3.13}$$

otherwise the sampler stays at ω, so that the detailed balance is satisfied. If we make sure that the chain is irreducible and aperiodic, then the chain converges to its stationary distribution. The sampler consists of five types of moves. They are

1. a birth/death move pair;
2. a split/merge move pair;
3. an extension/reduction move pair;
4. a track update move; and
5. a track switch move.

So there are a total of 8 moves. The paired move types are used to guarantee the reversibility of the Markov chain. Let m be the move and it is chosen randomly from the distribution $\xi_K(m)$ where K is the number of tracks of the current partition ω. Assume that we index each move by an integer such that $m = 1$ for a birth move, $m = 2$ for a death move and so on. When there is no track, we can only propose a birth move, so we set $\xi_0(m = 1) = 1$ and 0 for all other moves. When there is only a single target, we cannot propose a merge or track switch move, so $\xi_1(m = 4) = \xi_1(m = 7) = 0$. If it is desired, although not required, we can limit K and assign zero probability of proposing a birth or split move when we are at the limit of K. The MCMC data association algorithm for multiple target tracking is described in Figure 3.1. The inputs are the set of all observations Y, the number of samples n_{mc}, and the initial state ω_{init}. At each step of the algorithm, ω is the current state of the Markov chain. The acceptance probability $A(\omega, \omega')$ is defined in (3.13) where the posterior (3.12) is used.

Algorithm 3.1 (MCMC Data Association) :
Input: $Y, n_{mc}, \omega_{\text{init}}$
Output: $\hat{\omega} = \arg\max_{n=1}^{n_{mc}} p(\omega(n)|Y)$

$\omega \leftarrow \omega_{\text{init}}$
for $n = 1$ to n_{mc}
  choose a move m according to the distribution ξ_K
  propose ω' based on the move m and current state ω (described below)
  sample U from Unif[0, 1]
  if $U < A(\omega, \omega')$
    $\omega \leftarrow \omega'$
  end
  $\omega(n) \leftarrow \omega$
end

Figure 3.1: MCMC Data Association Algorithm

Remark 3.4 In Algorithm 3.1, we use MCMC to find a solution to a combinatorial optimization problem. So it can also be considered as a simulated annealing at a constant temperature. No burn-in samples are used since we are simply looking for a partition which maximizes the posterior. In addition, the memory requirement of the algorithm is at its bare minimum. Instead of keeping all $\{\omega(n)\}_{n=1}^{n_{mc}}$, we can simply keep the partition with the maximum posterior. If the algorithm is used to estimate $\mathbb{E}_{P(\omega|Y)} f(\omega)$ for some bounded function f, we will need burn-in samples and need to maintain the sufficient statistics for the desired expectation.

In order to make the algorithm more efficient, we make two additional assumptions: (1) the maximum directional spead of any target in $\mathcal{R}$ is less than $\bar{v}$; and (2) the number of consecutive missing observations of any track is less than $\bar{d}$. The first assumption is reasonable in a surveillance scenario since, in many cases, the maximal speed of a vehicle is generally known based on its type and terrain conditions. The second assumption is a user defined parameter and it can be used as one of the criteria to distinguish an event of a new object's appearance from an event of a continuation of an existing object. We will now assume that these two new conditions are added to the definition of Ω so each element $\omega \in \Omega$ satisfies two assumptions above as well as the five conditions described earlier.

We now introduce a data structure which is used to propose a new partition ω' in Algorithm 3.1. We define a neighborhood tree of observations as

$$L_d(y_t^j) = \{y_{t+d}^k \in y_{t+d} : \|y_t^j - y_{t+d}^k\| \leq d \cdot \bar{v}\} \tag{3.14}$$

for $d = 1, \ldots, \bar{d}$, $j = 1, \ldots, n_t$ and $t = 1, \ldots, T - d$. Here $\|\cdot\|$ is the usual Euclidean distance. This neighborhood tree groups temporally separated observations based on their distances and the algorithm proposes a new state ω' based on this neighborhood tree. Notice that we cannot simply

consider pairs of consecutive observations since there could be missing observations in between. The parameter d allows missing observations between observations. The use of this neighborhood tree makes the algorithm more scalable since distant observations will be considered separately. It is similar to the clustering technique used in MHT but $L_d(\cdot)$ is fixed for a given set of observations Y.

We now describe each move of the sampler in detail. First, let $\zeta(d)$ be a distribution of a random variable d taking values from $\{1, 2, \ldots, \bar{d}\}$. We assume the current state of the chain is $\omega = \omega^0 \cup \omega^1 \in \Omega$ and the number of tracks is $K = |\omega| - 1$. The proposed partition is denoted by $\omega' = \omega'^0 \cup \omega'^1 \in \Omega$. Note the abuse of notation below with indexing of time, i.e., when we say $\tau(t_i)$, t_i means the time at which the corresponding object is observed i times. So t_1 in $\tau(t_1)$ is the first time τ is observed and $t_{|\tau|}$ in $\tau(t_{|\tau|})$ is the last time τ is observed.

3.3.1 Birth and Death Moves

For a birth move, we increase the number of tracks from K to $K' = K + 1$ and select t_1 randomly from $\{1, \ldots, T-1\}$ as an appearance time of a new track. Let $\tau_{K'}$ be the track of this new object. Then we choose d_1 from the distribution ζ. Let $L^1_{d_1} = \{y^j_{t_1} : L_{d_1}(y^j_{t_1}) \neq \emptyset, y^j_{t_1} \notin \tau_k(t_1), j = 1, \ldots, n_{t_1}, k = 1, \ldots, K\}$. $L^1_{d_1}$ is a set of observations at t_1 such that, for any $y \in L^1_{d_1}$, y does not belong to other tracks and y has a descendant in $L_{d_1}(y)$. We choose $\tau_{K'}(t_1)$ randomly from $L^1_{d_1}$. If $L^1_{d_1}$ is empty, the move is rejected since the move is not reversible. Once the initial observation is chosen, we then choose the subsequent observations for the track $\tau_{K'}$. For $i = 2, 3, \ldots$, we choose d_i from ζ and then choose $\tau_{K'}(t_i)$ randomly from $L_{d_i}(\tau_{K'}(t_{i-1})) \setminus \{\tau_k(t_{i-1} + d_i) : k = 1, \ldots, K\}$ until this set is empty. We then propose this modified partition where $\omega'^1 = \omega^1 \cup \{\tau_{K'}\}$ and $\omega'^0 = \{\omega^0 \setminus \tau_{K'}\}$.

For a death move, we simply choose k randomly from $\{1, \ldots, K\}$ and delete the k-th track and propose a new partition where $\omega'^1 = \omega^1 \setminus \{\tau_k\}$ and $\omega'^0 = \{\omega^0 \cup \tau_k\}$.

3.3.2 Split and Merge Moves

For a split move, we select $\tau_s(t_r)$ randomly from $\{\tau_k(t_i) : i = 2, \ldots, |\tau_k| - 2, k = 1, \ldots, K\}$. Then we split the the track τ_s into τ_{s_1} and τ_{s_2} such that $\tau_{s_1} = \{\tau_s(t_i) : i = 1, \ldots, r\}$ and $\tau_{s_2} = \{\tau_s(t_i) : i = r+1, \ldots, |\tau_s|\}$. The modified track partition becomes $\omega'^1 = (\omega^1 \setminus \{\tau_s\}) \cup \{\tau_{s_1}\} \cup \{\tau_{s_2}\}$ and the false alarm partition ω'^0 is updated accordingly.

For a merge move, we consider the following set

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

Here we assume that $L_d(\cdot) = \emptyset$ for $d \leq 0$ and $d > \bar{d}$. We select a pair $(\tau_{s_1}(t_f), \tau_{s_2}(t_1))$ randomly from M. The tracks are combined into a single track $\tau_s = \tau_{s_1} \cup \tau_{s_2}$. Then we propose a new partition where $\omega'^1 = (\omega^1 \setminus (\{\tau_{s_1}\} \cup \{\tau_{s_2}\})) \cup \{\tau_s\}$ and ω'^0 with appropriate rearrangements.

3.3.3 Extension and Reduction Moves

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

3.3.4 Track Update Move

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

3.3.5 Track Switch Move

For a track switch move, we select a pair of observations $(\tau_{k_1}(t_r), \tau_{k_2}(t_r))$ from two different tracks such that, for some $d \leq \bar{d}$, $\tau_{k_1}(t_r + d) \in L_d(\tau_{k_2}(t_r))$ and $\tau_{k_2}(t_r + d) \in L_d(\tau_{k_1}(t_r))$. Then we let

$$\tau_{k_1} = \{\tau_{k_1}(t_1), \ldots, \tau_{k_1}(t_r), \tau_{k_2}(t_{r+1}), \ldots, \tau_{k_2}(t_{|\tau_{k_2}|})\}$$

and

$$\tau_{k_2} = \{\tau_{k_2}(t_1), \ldots, \tau_{k_2}(t_r), \tau_{k_1}(t_{r+1}), \ldots, \tau_{k_1}(t_{|\tau_{k_1}|})\}.$$

Remark 3.5 Since, in the track update move, there is always a positive probability of staying at the current state, the chain is aperiodic. The chain is also irreducible since it has a positive probability of visiting every element of Ω. Since the transitions described in Algorithm 3.1 satisfy the detailed balance, by the ergodic theorem, the chain converges to its stationary distribution.

3.4 A Greedy Algorithm for Multiple Target Tracking

In order to measure the performance of our MCMC tracker, we develop a greedy algorithm which is similar to the nearest-neighbor algorithm. The greedy multiple target tracking algorithm is shown in Figure 3.2. As shown in the next section, the algorithm works very well for simple problems and extremely fast compared to MCMC or MHT.

The inputs to Algorithm 3.2 are a set of observations $Y = \{y_t^j : 1 \leq j \leq n_t, 1 \leq t \leq T\}$ and a threshold σ. $L_d(\cdot)$ is defined in (3.14) and $\bar{d}$ is a maximum allowed number of consecutive missing observations. The algorithm selects a most probable track from observations that are not yet assigned and combines it to the partition ω^1 until there is no track whose posterior is greater than the threshold σ. In Algorithm 3.2, the linear system (3.3) is used and $\bar{x}_{t+d}(\tau_t)$ is the estimated state of the track τ_t computed by the time update equation of the Kalman filter (3.4). The worst-case running time of Algorithm 3.2 is $O(T^2 \bar{n}^{2\bar{d}+1})$ where $\bar{n} = \max_{t=1}^{T} n_t$.

Algorithm 3.2 (Greedy Multiple Target Tracking) :

Input: Y, σ

Output: $\omega = \omega^0 \cup \omega^1$

$\omega^1 \leftarrow \emptyset$
for $t = 1$ to T
  repeat
    $\omega_t \leftarrow \emptyset$
    foreach $(y_t^{j_t}, y_s^{j_s})$ in $\{(y_t^{j_t}, y_s^{j_s}) : 1 \leq s - t \leq \bar{d}$ and $y_t^{j_t}, y_s^{j_s} \notin \tau, \tau \in \omega^1\}$
      $\tau_t \leftarrow \emptyset$
      estimate the initial position and velocities from $(y_t^{j_t}, y_s^{j_s})$
      $\tau_t(t_1) \leftarrow y_t^{j_t}$ and $\tau_t(t_2) \leftarrow y_s^{j_s}$
      $r \leftarrow s$
      while $r < T$
        for $d = 1$ to $\bar{d}$
          if $L_d(\tau_t(r)) \neq \emptyset$
            $\tau_t(r + d) \leftarrow \arg\min_{y \in L_d(\tau_t(r))} \|y - \bar{x}_{r+d}(\tau_t)\|$
            break
          end
        end
        $r \leftarrow r + d$
      end
      if $p(\tau_t|Y) \geq \sigma$
        $\omega_t \leftarrow \omega_t \cup \{\tau_t\}$
      end
    end
    $\omega^1 \leftarrow \omega^1 \cup \{\arg\max_{\tau \in \omega_t} p(\tau|Y)\}$
  until $\omega_t \neq \emptyset$
end
$\omega^0 \leftarrow Y \setminus (\cup_{\tau \in \omega^1} \tau)$

Figure 3.2: Greedy Multiple Target Tracking Algorithm

3.5 Simulation Results

For the simulations we consider the surveillance over a rectangular region on a plane, $\mathcal{R} = [0, L] \times [0, L] \subset \mathbb{R}^2$. The state vector is $x = [x, y, \dot{x}, \dot{y}]^T$ where (x, y) is a position on $\mathcal{R}$ along the usual x and y axes and $(\dot{x}, \dot{y})$ is the velocity vector. The linear system model (3.3) is used (Li & Jilkov, 2000) where δ is an interval between observations and

$$A(\delta) = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad G(\delta) = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The covariance matrices are

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \quad \text{and} \quad R = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}.$$

The complexity of the multiple target tracking problems Problem 3.1 can be measured by several metrics: (1) the intensity of the false alarm rate λ_f; (2) the detection probability p_d; (3) the number of tracks K; and (4) the density of tracks. The problem gets more challenging with increasing λ_f, decreasing p_d, increasing K, and increasing density of tracks. The number of tracks itself may not make the problem more difficult if they are scattered apart. But the difficulty arises when there are many tracks that are moving closely and crossing each other. This is when the ambiguity of data association is greater. So we only consider situations in which the tracks move very closely so we can control the density of tracks by the number of tracks. Hence, we study the performance of the MCMC tracker against the greedy tracker Algorithm 3.2 and Multiple Hypothesis Tracker (MHT) by varing the parameters listed above.

Based on our model described above and in Problem 3.1, we have generated different scenarios. In particular, in all cases, except for the online tracking, a half of new objects appears from the left bottom quadrant of $\mathcal{R}$ and the other half of new objects appears from the the right bottom quadrant. The actual initial positions are chosen randomly from each quadrant. They all move diagonally so each group of tracks crosses the other group in the middle of $\mathcal{R}$. Also targets move very close to each other and there are also crossovers within each group. For an example, see Figure 3.5-3.14. The situations we have used for simulations below include very extreme cases and, in our opinion, such complex situations have not appeared in the multiple target tracking literatures.

We measure the performance of each algorithm by the estimated number of tracks, running time, and the log posterior $p(\omega|Y)$ relative to the original partition ω^* with which the test case is generated. Although the original partition may not maximize the posterior $p(\omega|Y)$, due to the noise and false alarms, it will give a value that is very close to the optimal value. Hence, we can measure the performance of each algorithm by comparing the posterior of the partition estimated by the algorithm with the posterior of the original partition. Let $\hat{\omega}$ be a partition of observations estimated by an algorithm. Then the relative log posterior (or log posterior difference) of $\hat{\omega}$ with respect to ω^* is defined as $RLP = \log p(\hat{\omega}|Y) - \log p(\omega^*|Y)$.

The algorithms are all written in Matlab and run on a PC with a 2-GHz Intel processor. The implementation of MHT algorithm is based on (Reid, 1979) with data structures and pruning techniques suggested in (Kurien, 1990; Cox, 1993). The implementation features 3-scan-back and 5-sigma validation region for gating. Also, at each step, we prune any hypothesis whose weight is less than 0.01.

3.5.1 Experiment I (Number of Tracks)

In this experiment, we separated the test cases into two groups: the first group consists of test cases with less than 10 tracks and the second group contains test cases with more than 10 tracks (up to 100). We call the tests in the first group as "simple" scenarios and the tests in the second group as "complex" scenarios. The main reason for this split is due to the lengthy running time of the MHT algorithm for tests with more than 10 tracks. So for the complex scenarios we have only run the greedy algorithm and MCMC. While we vary K, the other parameters are held fixed: $\mathcal{R} = [0, 1000] \times [0, 1000]$, $T = 10$, $\lambda_f V = 1$, $\bar{v} = 100$ unit length per unit time. The main focus of this experiment was to test the performance of data association of MCMC against other algorithms so the tracks are detected at all time, however, we have set $p_d = .9$ for the prior calculation. We have also set $\bar{d} = 1$. Since all tracks are observed, the number of observations increases as the number of tracks increases.

3.5.1.1 Simple Scenarios

For simple scenarios, we varied the number of tracks K from 2 to 9 and ran the three different tracking algorithms: Greedy (Algorithm 3.2), MHT and MCMC (Algorithm 3.1). For each value of K, we generated 5 tests randomly with restrictions described earlier so there are a total of 40 test cases. The results are summerized in Table 3.1. In Table 3.1, the results for MCMC are the average values over 10 repeated runs. For each test case, the table lists the number tracks (under the label K) and the log posterior $\log p(\omega|Y)$ (under the label log-p) of the original scenario and number of tracks and log posterior estimated by three different algorithms. It also lists the running time (under the label 'time', in seconds) of each algorithm. Note that the values of the log posterior is not normalized and this is not a problem since the normalizing constant Z in (3.12) is a constant for each test case.

We note that the initial state of MCMC is initialized with the greedy algorithm and the running time of MCMC includes both. Also we have used 10,000 samples for the MCMC algorithm. The relative log posterior of simple tests are depicted in Figure 3.3. The x-axis of the graph corresponds to the test case ID given in the Table 3.1. It shows a good performance from all three algorithms. In Experiment III (detection probability), we will see that the performance of MHT gets worse as the detection probability decreases while MCMC maintains high performance at low detection probability. The average running time of the algorithms at different number of tracks is shown in Figure 3.4. The running times of the greedy and MCMC are almost constants for the different number of tracks while the running time of MHT grows exponential with an increasing number of tracks.

Table 3.1: Experiment I (Number of Tracks, Simple Scenarios)

Test	True		Greedy			MHT			MCMC		
ID	K	log-p	K	log-p	time	K	log-p	time	K	log-p	time
1	2	-360.74	2	-360.74	0.50	2	-360.74	0.99	2	-360.74	23.61
2	2	-363.57	2	-372.87	0.03	2	-363.57	1.16	2	-363.57	23.62
3	2	-333.39	2	-333.39	0.03	2	-333.39	1.20	2	-333.39	23.84
4	2	-352.01	2	-352.01	0.03	2	-352.01	0.92	2	-352.01	24.52
5	2	-250.83	2	-250.83	0.02	2	-250.83	0.64	2	-250.83	23.63
6	3	-400.11	3	-400.11	0.05	3	-400.11	1.05	3	-400.11	23.60
7	3	-449.72	3	-449.72	0.05	3	-449.72	3.56	3	-449.72	25.34
8	3	-417.32	3	-417.32	0.05	3	-417.32	1.91	3	-417.32	23.83
9	3	-406.10	3	-406.10	0.06	3	-406.10	4.50	3	-406.10	25.00
10	3	-433.92	3	-433.92	0.05	3	-433.92	3.76	3	-433.92	24.93
11	4	-552.75	4	-562.46	0.09	4	-552.75	4.95	4	-552.75	24.30
12	4	-519.24	4	-524.46	0.11	4	-519.24	5.50	4	-519.24	24.93
13	4	-467.85	4	-467.85	0.09	4	-467.85	6.69	4	-467.85	24.71
14	4	-515.39	4	-515.39	0.09	4	-515.39	5.03	4	-515.39	24.73
15	4	-518.17	4	-521.95	0.08	4	-518.17	7.34	4	-518.55	24.61
16	5	-603.07	5	-603.07	0.13	5	-603.07	16.44	5	-603.07	24.76
17	5	-726.05	5	-726.05	0.16	5	-726.05	30.89	5	-726.05	25.11
18	5	-610.27	5	-610.27	0.42	5	-610.27	9.50	5	-610.27	24.78
19	5	-612.18	5	-614.57	0.13	5	-612.18	35.28	5	-612.18	25.62
20	5	-616.94	5	-619.44	0.13	5	-616.94	18.64	5	-616.94	25.01
21	6	-757.19	6	-757.19	0.25	6	-757.19	43.88	6	-757.19	26.14
22	6	-664.90	6	-664.90	0.17	6	-664.90	136.97	6	-664.90	24.61
23	6	-793.65	6	-793.65	0.22	6	-793.65	53.84	6	-793.65	25.80
24	6	-728.19	6	-728.19	0.16	6	-728.19	18.30	6	-728.19	24.74
25	6	-664.24	6	-664.24	0.20	6	-664.24	23.51	6	-664.24	25.42
26	7	-826.22	7	-826.22	0.30	7	-826.22	94.45	7	-826.22	25.05
27	7	-933.06	7	-933.06	0.53	7	-933.06	163.45	7	-933.06	25.70
28	7	-978.50	7	-978.50	0.27	8	-991.35	122.36	7	-978.50	25.25
29	7	-885.05	7	-887.12	0.33	7	-885.05	160.34	7	-885.05	26.57
30	7	-872.71	7	-872.71	0.28	7	-872.71	54.94	7	-872.71	25.66
31	8	-971.25	8	-971.25	0.56	8	-971.25	317.59	8	-971.25	26.24
32	8	-925.28	8	-925.41	0.34	8	-925.28	291.14	8	-925.37	25.55
33	8	-922.32	8	-922.32	0.55	9	-933.04	658.13	8	-922.32	27.98
34	8	-958.33	8	-958.33	0.31	8	-958.33	136.11	8	-958.33	25.81
35	8	-887.30	8	-887.30	0.59	8	-887.30	504.02	8	-887.30	25.96
36	9	-995.72	9	-995.72	0.33	9	-995.72	116.74	9	-995.72	24.91
37	9	-986.77	9	-986.77	0.44	9	-986.77	1048.14	9	-986.77	25.55
38	9	-1096.24	9	-1095.63	0.45	9	-1095.63	533.92	9	-1095.63	26.14
39	9	-1060.32	9	-1090.55	0.78	9	-1060.02	1662.08	9	-1084.65	25.95
40	9	-1108.82	9	-1108.82	0.50	9	-1108.82	1029.44	9	-1108.82	26.39

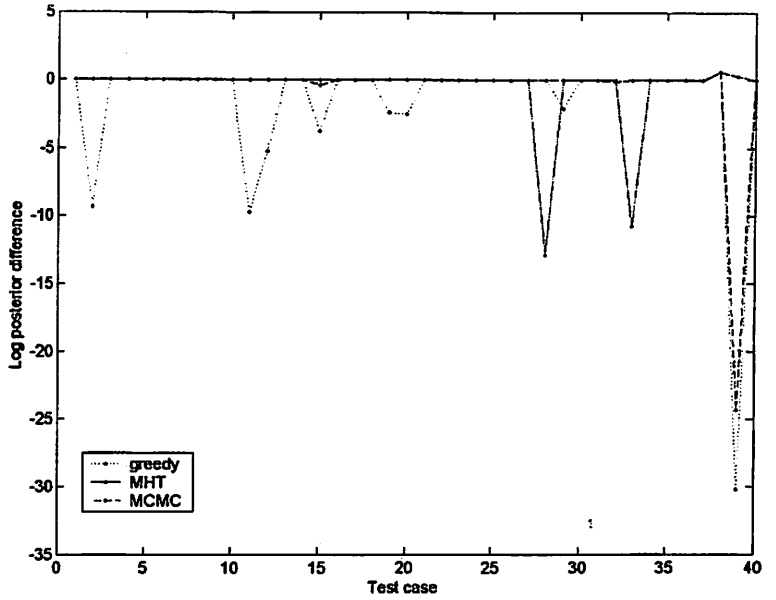


Figure 3.3: Relative log posterior: Experiment I (number of tracks, simple scenarios)

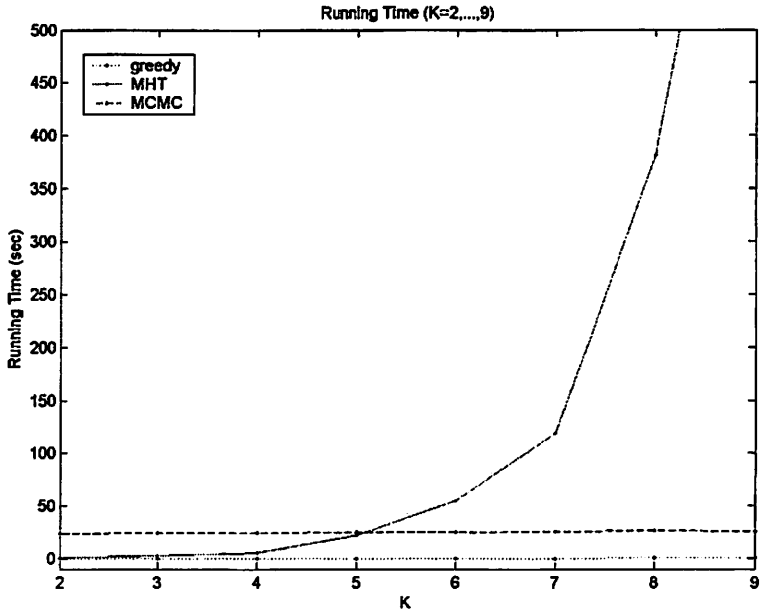


Figure 3.4: Average running times of greedy, MHT and MCMC algorithms vs. number of targets: Experiment I (number of tracks, simple scenarios)

3.5.1.2 Complex Scenarios

The excellence of MCMC can be seen when there are a large number of tracks. In the second part of the experiment, we vary K from 10 to 100 while the other parameters are held fixed as the first part (the actual values of K are 10, 20, 30, 40, 50, 75 and 100). As before, for each value of K, we have generated 5 random scenarios. In Table 3.2 we show the results from the greedy algorithm and MCMC. The results under MCMC1 is collected by running MCMC with 10,000 samples while MCMC2 is run with 100,000 samples. The results for both MCMC1 and MCMC2 are the average values over 10 repeated runs. The relative log posterior is drawn in Figure 3.15 in which the x-axis is indexed by test case IDs from Table 3.2.

An example is shown in Figure 3.5-3.14 with observations, true tracks and the estimated tracks at each time. This is the test case 35 in which there are 100 tracks moving closely to each other and the estimated tracks are from MCMC with 100,000 samples. Around $t = 5$ and $t = 6$, we can see that there are many crossovers between tracks and the difficulty of this scenario.

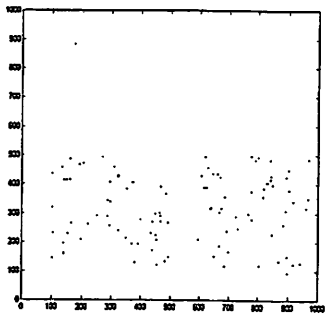
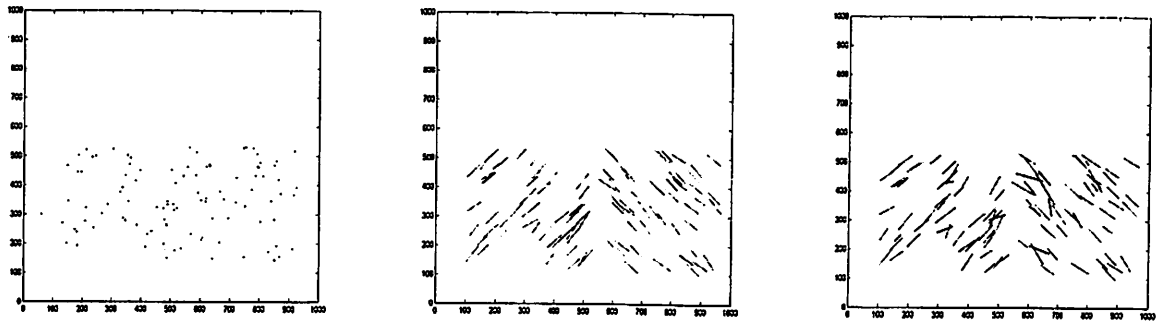
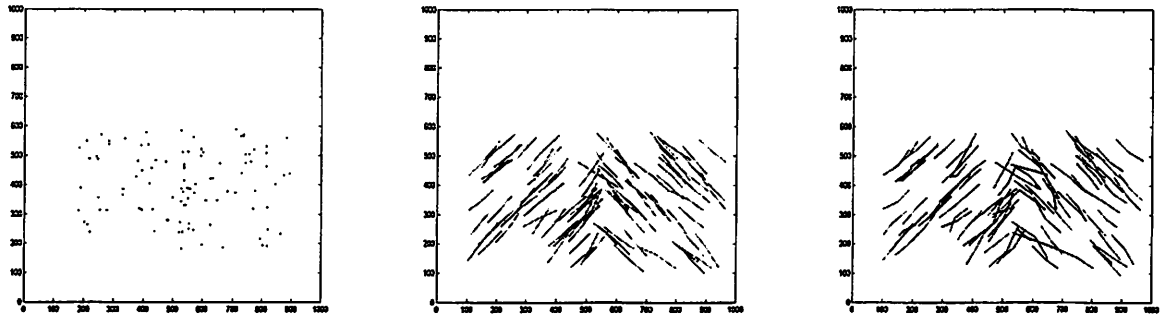


Figure 3.5: Test case 35 ($t = 1, K = 100$): observations



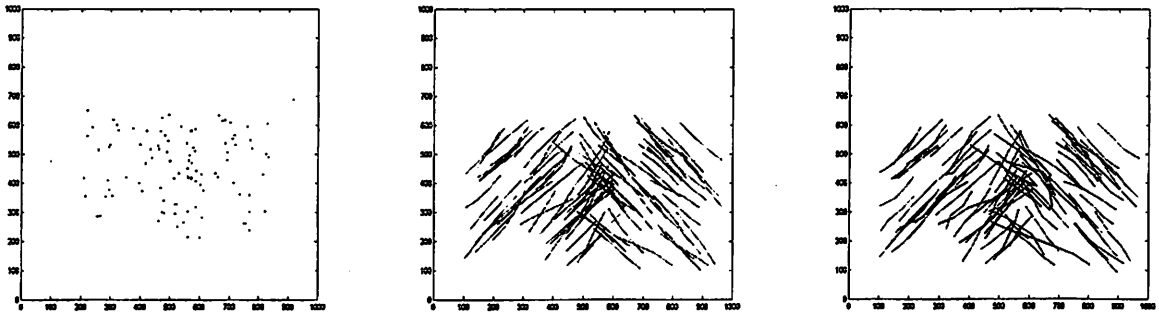
(a) (b) (c)

Figure 3.6: Test case 35 ($t = 2, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



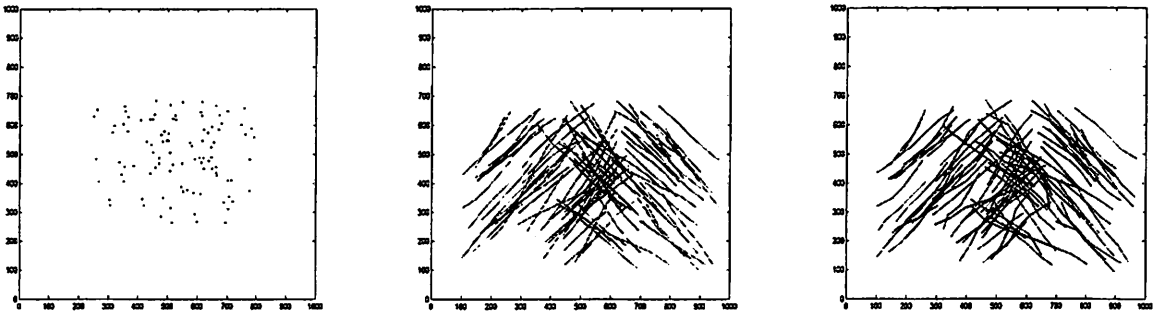
(a) (b) (c)

Figure 3.7: Test case 35 ($t = 3, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



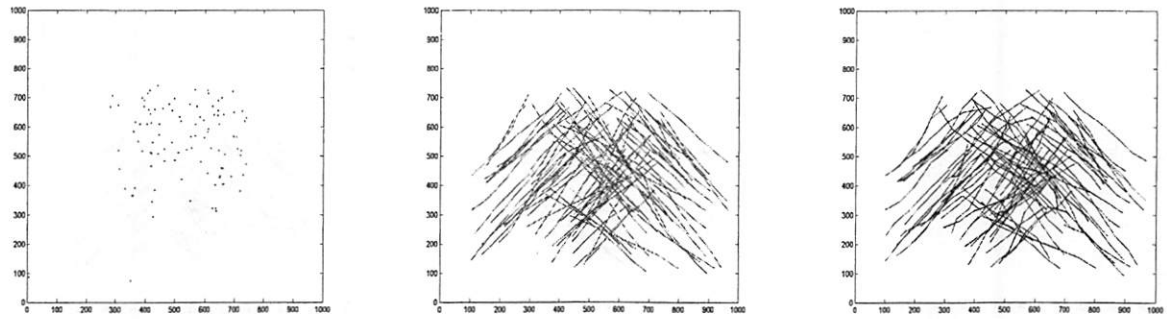
(a) (b) (c)

Figure 3.8: Test case 35 ($t = 4, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



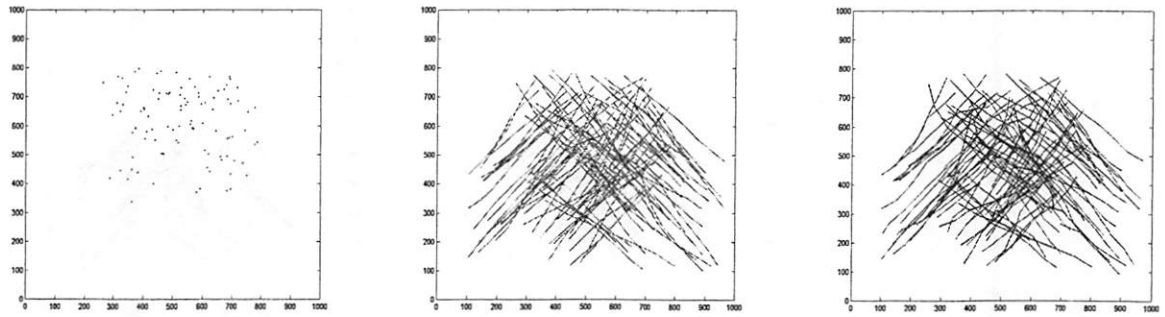
(a) (b) (c)

Figure 3.9: Test case 35 ($t = 5, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



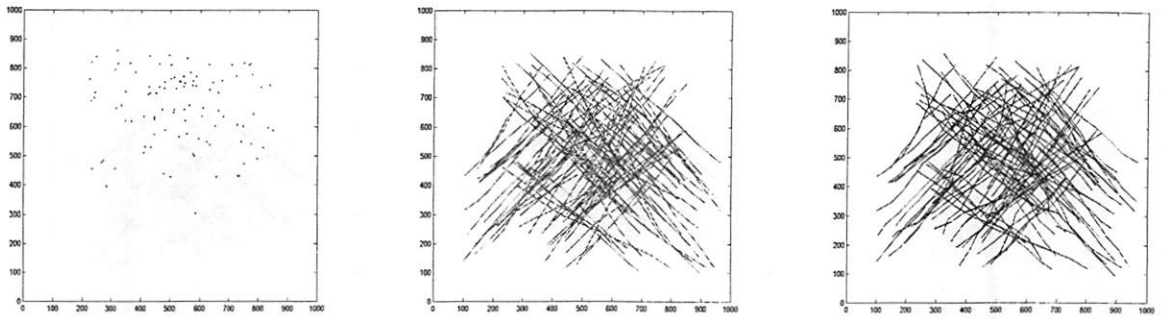
(a) (b) (c)

Figure 3.10: Test case 35 ($t = 6, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



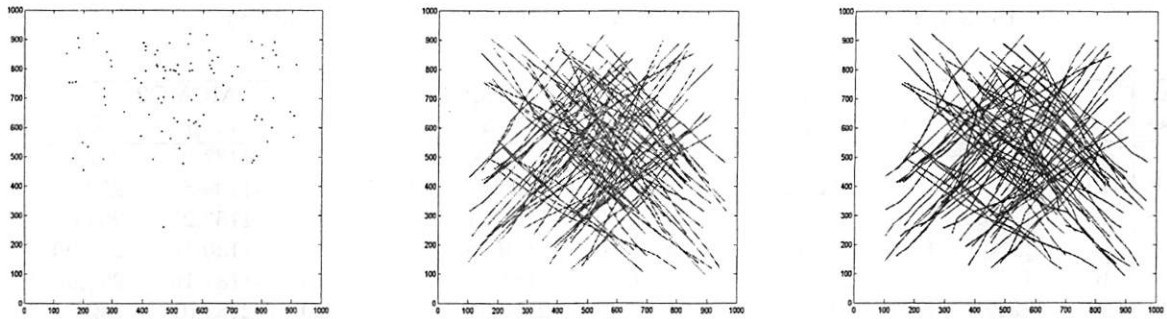
(a) (b) (c)

Figure 3.11: Test case 35 ($t = 7, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



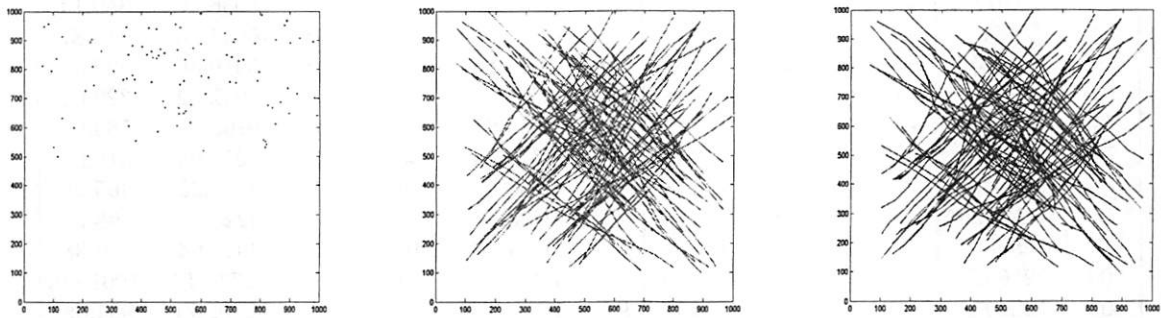
(a) (b) (c)

Figure 3.12: Test case 35 ($t = 8, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



(a) (b) (c)

Figure 3.13: Test case 35 ($t = 9, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks



(a) (b) (c)

Figure 3.14: Test case 35 ($t = 10, K = 100$): (a) observations; (b) true tracks; (c) estimated tracks

Table 3.2: Experiment I (Number of Tracks, Complex Scenarios)

Test	True		Greedy			MCMC1			MCMC2		
Case	K	log-p	K	log-p	time	K	log-p	time	K	log-p	time
1	10	-1177.25	10	-1177.25	0.88	10	-1177.25	26.38	10	-1177.25	252.67
2	10	-1203.46	10	-1199.81	0.66	10	-1199.81	26.47	10	-1199.81	253.59
3	10	-1153.91	10	-1163.27	0.64	10	-1163.27	26.60	10	-1163.27	254.00
4	10	-1130.18	10	-1132.44	0.72	10	-1130.56	27.29	10	-1130.18	258.90
5	10	-1182.82	10	-1181.16	0.59	10	-1181.16	26.74	10	-1181.16	251.99
6	20	-2289.17	20	-2322.98	3.63	20	-2310.91	29.88	20	-2289.01	262.67
7	20	-2264.83	19	-2361.84	4.47	20	-2295.60	30.90	20	-2287.04	264.33
8	20	-2188.34	20	-2211.27	3.66	20	-2208.79	30.22	20	-2206.62	263.47
9	20	-2208.07	20	-2208.07	4.34	20	-2205.65	31.67	20	-2204.09	271.03
10	20	-2371.71	20	-2369.62	4.44	20	-2366.78	31.32	20	-2365.77	264.51
11	30	-3456.53	30	-3488.35	16.26	30	-3484.01	43.00	30	-3478.68	280.10
12	30	-3416.30	30	-3441.69	14.56	30	-3430.24	42.78	30	-3413.30	291.02
13	30	-3507.44	30	-3534.33	13.83	30	-3524.84	41.00	30	-3517.72	277.89
14	30	-3382.82	28	-3539.04	13.48	30	-3494.55	41.80	30	-3472.43	275.58
15	30	-3488.15	30	-3596.72	12.92	30	-3573.52	42.38	30	-3521.00	297.16
16	40	-4586.11	40	-4628.19	26.03	40	-4622.25	53.53	40	-4600.88	292.67
17	40	-4534.19	39	-4643.68	25.20	40	-4608.79	52.66	40	-4584.21	290.85
18	40	-4605.69	40	-4656.39	25.94	40	-4651.85	53.73	40	-4637.61	292.78
19	40	-4578.14	40	-4670.10	28.67	40	-4622.87	57.65	40	-4590.60	307.10
20	40	-4686.27	39	-4783.64	28.73	40	-4756.28	57.09	40	-4721.35	296.58
21	50	-5874.48	50	-5915.82	54.34	50	-5909.25	83.57	50	-5896.37	335.29
22	50	-5926.87	49	-6073.31	54.25	49	-6061.46	91.42	50	-6036.16	359.14
23	50	-5955.08	48	-6189.99	61.03	49	-6112.98	101.85	49	-6044.76	418.87
24	50	-5855.53	50	-5988.29	51.88	50	-5975.89	80.59	50	-5919.36	322.49
25	50	-5880.14	49	-6039.39	53.38	50	-6001.72	82.04	50	-5952.32	323.61
26	75	-8983.74	70	-9340.10	170.53	74	-9244.61	212.21	74	-9163.04	563.62
27	75	-8965.92	70	-9442.77	190.09	74	-9347.76	230.19	75	-9253.98	510.04
28	75	-9031.75	72	-9313.48	184.27	75	-9238.49	219.30	75	-9174.25	465.25
29	75	-8947.20	71	-9371.86	199.06	74	-9293.76	242.63	74	-9225.16	598.64
30	75	-8956.57	73	-9194.08	164.23	75	-9144.20	210.23	75	-9097.64	489.38
31	100	-12166.58	92	-12921.48	502.03	96	-12825.28	560.19	97	-12738.13	1001.69
32	100	-12220.28	93	-13009.55	433.17	98	-12884.61	488.19	99	-12802.71	934.69
33	100	-12319.48	91	-13023.92	446.44	96	-12938.79	496.99	97	-12861.30	873.28
34	100	-12196.97	94	-12810.76	451.08	97	-12748.25	505.54	97	-12670.33	923.49
35	100	-12259.77	90	-13061.97	428.31	98	-12902.09	472.96	98	-12799.74	807.21

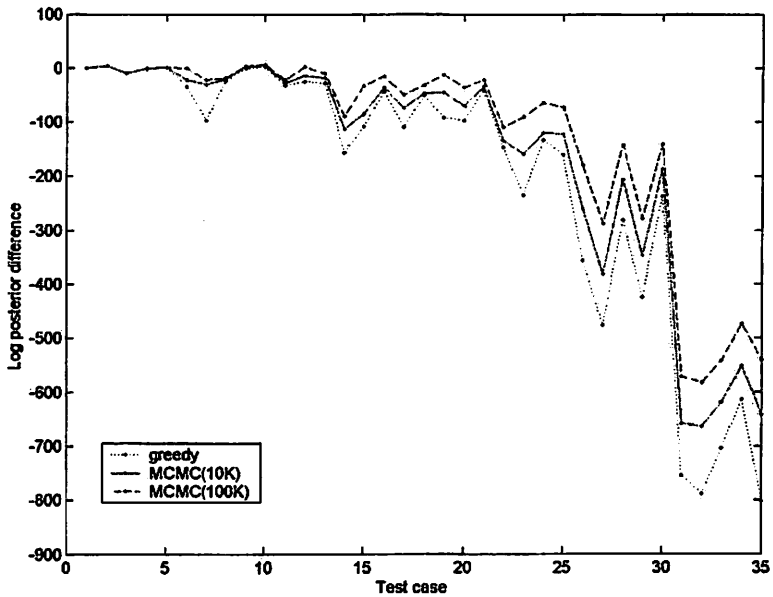


Figure 3.15: Relative log posterior: Experiment I (number of tracks, complex scenarios)

3.5.2 Experiment II (False Alarms)

Now the settings are the same as the previous experiment but we vary the false alarm rates while the number of tracks is fixed at $K = 10$. The test cases for this experiment are prepared as following. We first generated five different random scenarios each with 10 tracks. Then we applied different false alarm rates to generate test cases. The false alarm rates are varied from $\lambda_f V = 1$ to $\lambda_f V = 100$ with an increment of 10.

Table 3.3 and Table 3.4 show the summary of simulation runs. As before the results for MCMC in the table are averaged over 10 repeated runs. We used 10,000 samples for MCMC1 and 100,000 samples for MCMC2. Figure 3.16 shows the relative log posterior and the x-axis is indexed by the test case ID listed in Table 3.3 and 3.4. The MCMC algorithm performs well up to $\lambda_f V = 70$ but reports extra tracks for $\lambda_f V \geq 80$ due to the large number of false alarms.

Table 3.3: Experiment II (False Alarms, Test Cases 1-25)

Test		True		Greedy			MCMC1			MCMC2		
Case	$\lambda_f V$	K	log-p	K	log-p	time	K	log-p	time	K	log-p	time
1	1	10	-1174.35	10	-1174.57	0.75	10	-1174.35	37.24	10	-1174.35	362.78
2	1	10	-1170.19	10	-1183.83	0.80	10	-1170.16	38.03	10	-1170.16	365.79
3	1	10	-1249.11	9	-1302.52	0.81	10	-1261.82	32.36	10	-1261.82	309.95
4	1	10	-1127.46	10	-1130.14	0.69	10	-1127.46	31.24	10	-1127.46	295.02
5	1	10	-1201.44	10	-1228.07	0.64	10	-1205.24	32.41	10	-1205.24	308.06
6	10	10	-2670.62	10	-2672.06	1.59	10	-2670.62	44.39	10	-2670.62	426.10
7	10	10	-2669.19	10	-2682.83	1.11	10	-2669.17	40.50	10	-2669.17	388.84
8	10	10	-2215.16	9	-2244.91	1.14	10	-2224.92	34.55	10	-2223.20	329.33
9	10	10	-2254.76	10	-2262.11	1.19	10	-2254.76	32.45	10	-2254.76	307.59
10	10	10	-2622.22	10	-2626.32	1.33	10	-2626.01	35.10	10	-2626.01	330.83
11	20	10	-3903.93	10	-3902.27	2.34	10	-3902.07	45.27	10	-3902.04	425.14
12	20	10	-4018.67	10	-4083.63	2.28	10	-4028.91	45.77	10	-4024.07	431.37
13	20	10	-4143.07	10	-4183.23	1.92	10	-4174.32	39.08	10	-4172.07	367.10
14	20	10	-3893.36	10	-3896.05	2.28	10	-3893.13	37.44	10	-3893.13	344.41
15	20	10	-3617.11	10	-3621.79	2.27	10	-3621.68	39.95	10	-3621.68	369.66
16	30	10	-5353.26	9	-5365.49	4.29	10	-5356.68	54.11	10	-5354.74	497.01
17	30	10	-5434.27	10	-5454.04	5.01	10	-5436.77	56.26	10	-5436.79	508.37
18	30	10	-4912.82	9	-4932.85	3.31	9	-4932.14	45.33	9	-4932.85	415.92
19	30	10	-5792.97	10	-5797.56	4.92	10	-5797.38	46.20	10	-5797.38	406.58
20	30	10	-5135.46	10	-5189.54	3.08	10	-5173.54	43.49	10	-5171.94	396.79
21	40	10	-6553.33	10	-6565.26	5.75	10	-6565.05	59.31	10	-6565.03	532.48
22	40	10	-6448.16	10	-6450.36	5.98	10	-6450.36	60.62	10	-6450.36	538.09
23	40	10	-7094.37	10	-7098.61	5.97	10	-7096.40	53.59	10	-7095.84	468.68
24	40	10	-6510.52	10	-6545.15	7.60	10	-6527.33	50.51	10	-6526.30	430.59
25	40	10	-7164.53	10	-7165.99	8.61	10	-7165.99	58.08	10	-7165.99	488.62

Table 3.4: Experiment II (False Alarms, Test Cases 26-55)

Test		True		Greedy			MCMC1			MCMC2		
Case	$\lambda_f V$	K	log-p	K	log-p	time	K	log-p	time	K	log-p	time
26	50	10	-7856.05	9	-7867.56	9.03	9	-7867.38	63.63	9	-7867.34	541.91
27	50	10	-8036.08	11	-8061.31	11.35	11	-8053.08	67.06	11	-8052.51	556.25
28	50	10	-8437.32	9	-8472.13	10.82	9	-8472.13	61.29	9	-8472.13	504.02
29	50	10	-8180.13	11	-8234.77	14.70	11	-8205.45	62.61	11	-8201.86	488.76
30	50	10	-8047.15	9	-8080.22	8.81	9	-8074.04	58.59	9	-8074.01	495.25
31	60	10	-9504.44	12	-9572.64	20.43	12	-9543.51	79.64	12	-9543.40	599.58
32	60	10	-10147.65	10	-10168.40	21.22	10	-10160.39	84.77	10	-10158.13	632.38
33	60	10	-9416.06	11	-9492.17	12.95	11	-9467.68	64.05	11	-9467.13	512.29
34	60	10	-9144.81	10	-9205.11	16.44	10	-9173.30	66.93	10	-9171.80	512.87
35	60	10	-9780.04	12	-9900.56	25.19	12	-9862.05	79.08	12	-9860.98	550.73
36	70	10	-10539.08	13	-10665.08	26.00	13	-10615.34	88.84	13	-10606.83	634.52
37	70	10	-10482.04	10	-10532.26	18.99	10	-10512.09	78.04	10	-10510.15	587.15
38	70	10	-10770.95	11	-10835.99	19.52	11	-10805.57	73.80	11	-10801.59	553.09
39	70	10	-11589.53	12	-11621.68	27.75	12	-11618.54	80.65	12	-11618.41	543.17
40	70	10	-10313.27	11	-10384.30	18.28	11	-10342.98	69.77	11	-10335.84	517.25
41	80	10	-12687.47	17	-12955.88	41.93	17	-12842.29	107.62	17	-12837.89	664.43
42	80	10	-11987.91	15	-12129.44	32.80	15	-12072.62	99.13	15	-12073.48	658.75
43	80	10	-12764.25	18	-12977.33	51.07	18	-12895.99	102.19	18	-12894.13	566.25
44	80	10	-12161.23	13	-12262.33	35.23	13	-12220.48	85.63	13	-12214.45	532.31
45	80	10	-12446.56	16	-12650.26	43.92	16	-12575.97	96.49	16	-12573.88	565.19
46	90	10	-13369.79	18	-13613.91	60.46	18	-13508.18	124.94	18	-13509.31	671.80
47	90	10	-13953.74	22	-14371.54	59.30	22	-14186.22	123.73	22	-14169.47	671.03
48	90	10	-12850.74	17	-13048.56	55.04	17	-12962.91	104.86	17	-12957.12	559.87
49	90	10	-14237.42	22	-14606.52	89.83	22	-14441.09	141.36	22	-14440.30	632.11
50	90	10	-14479.98	22	-14876.88	71.25	22	-14728.39	123.13	22	-14728.26	592.94
51	100	10	-15109.45	23	-15539.62	95.47	23	-15342.70	162.90	23	-15343.32	734.55
52	100	10	-14816.08	22	-15206.05	97.81	22	-15033.23	192.75	22	-15035.52	768.74
53	100	10	-15455.53	26	-15940.48	139.23	26	-15726.68	300.07	26	-15724.96	1118.71
54	100	10	-15101.24	25	-15577.06	85.20	25	-15389.21	242.53	25	-15383.08	1099.15
55	100	10	-15554.05	28	-16075.47	95.53	28	-15851.56	263.20	28	-15847.51	1136.50

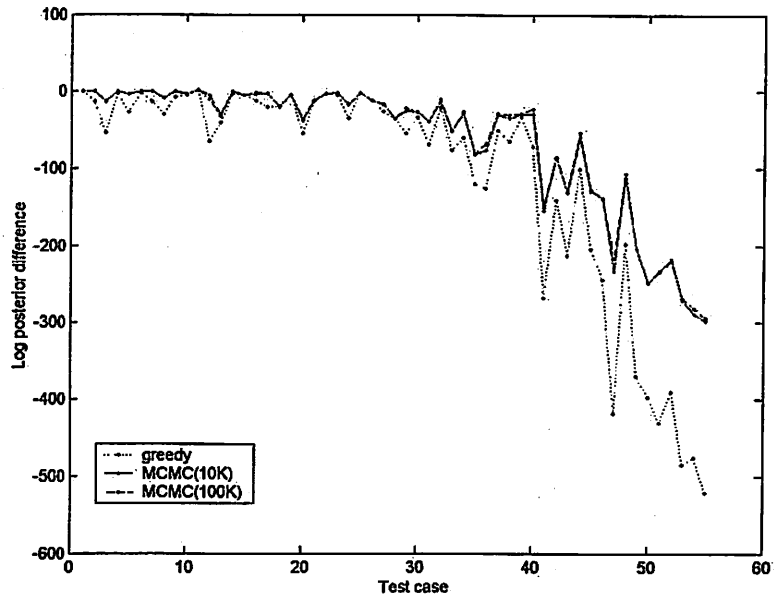


Figure 3.16: Relative log posterior: Experiment II (false alarms)

3.5.3 Experiment III (Detection Probability)

In this experiment we vary the detection probability p_d from .5 to .9 while keeping the other parameters as previous experiments except $K = 4$, $\lambda_f V = 1$ and $\bar{d} = 3$. Now the tracks are not observed all the time. A small number of tracks is used so we can compare the performance of the greedy and MCMC algorithms against the performance of MHT. Table 3.5 lists test cases 1 through 25 with the log posterior estimated from the greedy and MHT. The next table Table 3.6 lists the same test cases with the estimated log posteriors from three different variations of the MCMC algorithm. The entries for the MCMC algorithm are average values of 10 repeated runs. The column labeled as MCMC1 lists the results from MCMC with 10,000 samples where the Markov chain is initialized with the result from the greedy algorithm. MCMC2 and MCMC3 are not initialized with the greedy algorithm, instead they started with an empty track. MCMC2 is run for 10,000 steps (or samples) while MCMC3 is run for 100,000 steps.

The relative log posterior is drawn in Figure 3.17. The x-axis is indexed by the test case ID shown in Table 3.5 and 3.6. It shows that MCMC performs very well when the detection probability is low compared with the greedy and MHT algorithms and gives a steady performance for high detection probability. Also the uninitialized MCMC performs well in the low detection probability region while the MCMC initialized with the greedy algorithm works well in the high detection probability region. Although, in theory, MHT gives an optimal solution in the sense of MAP, it performs poorly when the detection probability is low due to the heuristics such as pruning and N-scan-back techniques used to reduce the complexity. The heuristics are required parts of MHT in practice. For example, when there are 10 new observations, one has to consider all the associations from the previous hypotheses to the new observations and consider the cases when one or more of the new observations are initiating new tracks. The complexity of associating the new observations with the previous hypotheses can be reduced by the gating, which is another heuristic used in MHT, but, for a new track initiation, we need to consider all 2^{10} possibilities. Without the pruning and N-scan-back logic the problem grows exponentially fast even for a small problem. In practice, MHT with heuristics works well when there are a few number of hyphotheses which carry the most of weights. But when the detection probability is low there are many hypotheses with low weights and there is no small set of dominating hypotheses. So MHT cannot perform well when the detection probability is low. But when the detection probability is high, MHT again suffers from a large number of observations. Another noticeable benefit of the MCMC algorithm is that its running time can be regulated by the number of samples as shown in Table 3.5 and 3.6 (also from the other experiments). But the running time of MHT depends on the complexity of the problem instance and is not predictable in advance.

Table 3.5: Summary of Experiment III (Detection Probability, Part 1)

Test		True		Greedy			MHT		
Case	p_d	K	log-p	K	log-p	time	K	log-p	time
1	0.50	4	-529.44	3	-586.11	0.58	10	-614.48	50.69
2	0.50	4	-648.15	4	-722.62	0.14	10	-697.60	114.28
3	0.50	4	-630.09	4	-710.33	0.41	10	-694.17	46.53
4	0.50	4	-577.89	5	-609.79	0.08	12	-658.43	79.90
5	0.50	4	-610.30	2	-690.91	0.28	10	-677.09	38.06
6	0.60	4	-588.27	4	-692.42	0.33	8	-640.27	84.68
7	0.60	4	-663.10	5	-707.44	0.20	6	-680.49	79.76
8	0.60	4	-683.68	5	-770.56	0.20	8	-740.54	60.82
9	0.60	4	-663.28	4	-715.05	0.19	10	-752.72	629.93
10	0.60	4	-572.01	5	-645.21	0.11	7	-608.49	25.82
11	0.70	4	-677.99	4	-794.38	0.28	6	-705.00	130.20
12	0.70	4	-681.25	3	-768.95	0.56	8	-738.32	178.07
13	0.70	4	-720.60	5	-770.44	0.17	9	-800.52	160.61
14	0.70	4	-678.98	4	-819.29	0.33	7	-734.94	362.78
15	0.70	4	-744.53	4	-836.38	0.19	10	-846.91	287.68
16	0.80	4	-857.44	4	-942.83	0.27	9	-961.85	206.56
17	0.80	4	-720.79	4	-781.88	0.23	6	-759.97	126.85
18	0.80	4	-679.42	4	-783.20	0.36	4	-679.42	54.77
19	0.80	4	-801.30	4	-886.89	0.36	10	-910.22	368.47
20	0.80	4	-741.40	4	-849.55	0.48	7	-812.68	217.55
21	0.90	4	-730.47	4	-789.75	0.28	4	-730.47	1484.96
22	0.90	4	-850.02	4	-907.20	0.30	7	-938.74	2140.90
23	0.90	4	-711.77	4	-811.97	0.34	4	-711.77	1309.84
24	0.90	4	-745.47	4	-847.83	0.24	5	-777.04	91.34
25	0.90	4	-746.35	5	-846.86	0.58	5	-778.91	239.32

Table 3.6: Experiment III (Detection Probability, Part 2)

Test		True		MCMC1			MCMC2			MCMC3		
Case	p_d	K	log-p	K	log-p	time	K	log-p	time	K	log-p	time
1	0.50	4	-529.44	4	-567.61	14.99	4	-565.41	14.73	4	-555.45	150.33
2	0.50	4	-648.15	4	-666.81	18.97	4	-677.89	18.75	4	-668.80	183.37
3	0.50	4	-630.09	4	-662.60	16.70	4	-656.54	16.89	5	-652.43	164.67
4	0.50	4	-577.89	4	-602.81	16.67	4	-608.57	16.46	4	-597.03	159.56
5	0.50	4	-610.30	3	-645.45	17.19	3	-648.97	17.52	3	-642.15	160.50
6	0.60	4	-588.27	4	-614.14	16.88	4	-623.93	20.70	4	-614.22	183.72
7	0.60	4	-663.10	4	-696.13	19.19	4	-694.04	18.38	4	-683.92	184.03
8	0.60	4	-683.68	5	-719.30	18.83	5	-727.10	17.72	5	-716.75	171.94
9	0.60	4	-663.28	4	-708.15	19.65	4	-704.20	19.41	4	-703.56	187.65
10	0.60	4	-572.01	5	-612.81	17.44	5	-606.94	17.79	5	-601.98	165.19
11	0.70	4	-677.99	4	-702.72	20.20	4	-739.49	19.58	4	-713.41	197.02
12	0.70	4	-681.25	3	-736.74	18.84	4	-718.31	18.79	4	-707.57	182.25
13	0.70	4	-720.60	5	-749.81	19.42	5	-778.24	18.23	5	-769.82	178.44
14	0.70	4	-678.98	4	-736.73	20.23	4	-749.62	19.84	4	-735.08	189.84
15	0.70	4	-744.53	4	-805.88	20.41	4	-810.98	20.30	4	-804.71	195.43
16	0.80	4	-857.44	4	-920.20	20.35	4	-909.21	20.64	4	-898.45	198.51
17	0.80	4	-720.79	4	-769.26	20.92	4	-788.17	19.86	4	-774.90	196.27
18	0.80	4	-679.42	4	-749.70	20.44	4	-754.67	20.73	4	-741.29	203.49
19	0.80	4	-801.30	4	-847.51	22.34	4	-868.62	21.83	4	-871.90	209.73
20	0.80	4	-741.40	4	-840.47	21.03	5	-823.19	21.00	5	-807.92	200.96
21	0.90	4	-730.47	4	-789.75	22.70	4	-834.94	21.90	4	-821.50	217.86
22	0.90	4	-850.02	4	-907.20	22.60	4	-942.18	22.37	4	-926.95	221.14
23	0.90	4	-711.77	4	-771.88	23.16	4	-837.38	22.04	4	-825.33	217.50
24	0.90	4	-745.47	4	-802.02	21.91	4	-835.93	20.98	4	-818.08	202.19
25	0.90	4	-746.35	5	-845.50	22.11	4	-849.52	22.23	4	-833.06	216.84

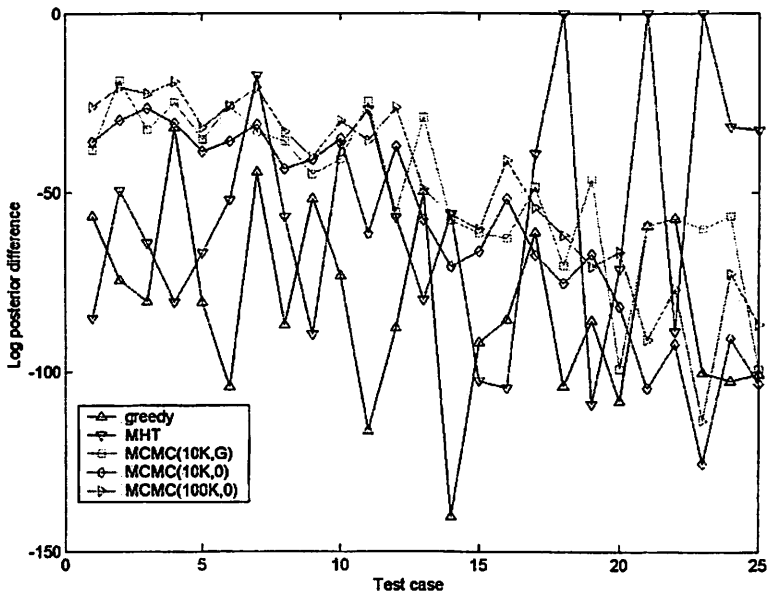


Figure 3.17: Relative log posterior: Experiment III (detection probability)

3.5.4 Online MCMC Multiple Target Tracker

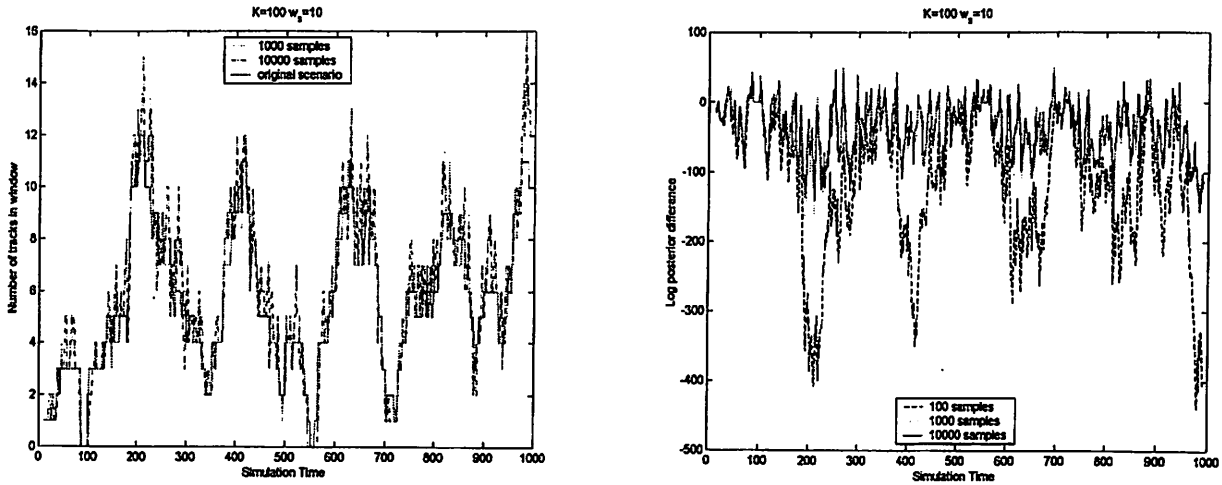
The extension of the MCMC tracker to an online, real-time tracking is a trivial task. As we have seen from the previous experiments that the MCMC tracker works well even when T is small. Hence, we implement a sliding window of size w_s using Algorithm 3.1. Suppose that we are at time t and receiving new observations, we use the previous estimate to initialize the MCMC tracker and run the MCMC tracker on the observations from $t - w_s + 1$ to t which combines both previous and new observations.

A total of three test cases are generated: (test case 1) 100 tracks, (test case 2) 200 tracks and (test case 3) 300 tracks. The surveillance duration is increased to $T = 1000$ and the surveillance region is now $\mathcal{R} = [0, 10000] \times [0, 10000]$. The other parameters are: $\lambda_f V = 10$, $p_d = .9$, $\bar{d} = 3$ and $\bar{v} = 200$. The objects appear and disappear at random time and location and the number of tracks for each test is the total number of tracks for the whole duration of simulation. The number of tracks change as objects appear and disappear as shown in Figure 3.18 - 3.23 (refer the figures on the left). These test cases represent true instances of the generalized (discrete-time) multiple target tracking problem described in Problem 3.1.

Each test is run with two different window sizes: $w_s = 10$ and $w_s = 20$. The results are shown in Figure 3.18 - 3.23. The figures on the left show the actual number of tracks in the window $[t - w_s + 1, t]$ and the estimated number of tracks by MCMC with 1000 samples and 10000 samples at each simulation step t. The figures on the right show the relative log posteriors from MCMC with 100, 1000, and 10000 samples for the same window $[t - w_s + 1, t]$. We see that the estimated number of tracks is very close to the true track count and the log posterior is close to the actual one. The average running time of the algorithm per simulation time step is shown below. Our experiences with other algorithms suggest that the running time of an algorithm improves dramatically when it is converted from Matlab to C/C++. So we suggest the readers to view the running time listed below as relative values.

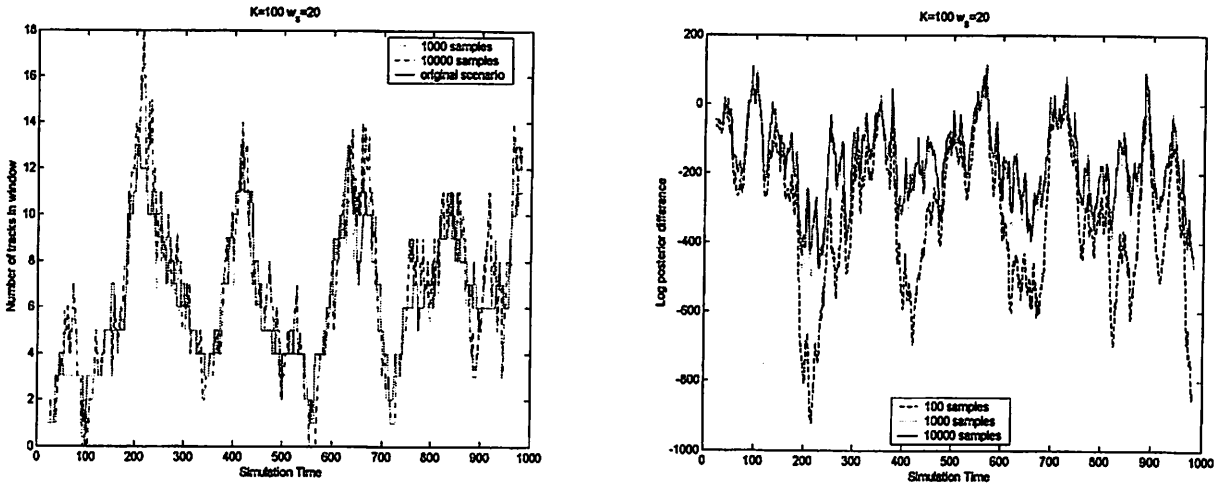
Table 3.7: Average Running Time (per step, in seconds)

Test Case	K	w_s	100 samples	1000 samples	10000 samples
1	100	10	.23	1.65	15.82
		20	.29	2.14	20.78
2	200	10	.32	1.83	17.13
		20	.38	2.41	22.79
3	300	10	.40	1.93	17.46
		20	.45	2.53	23.45



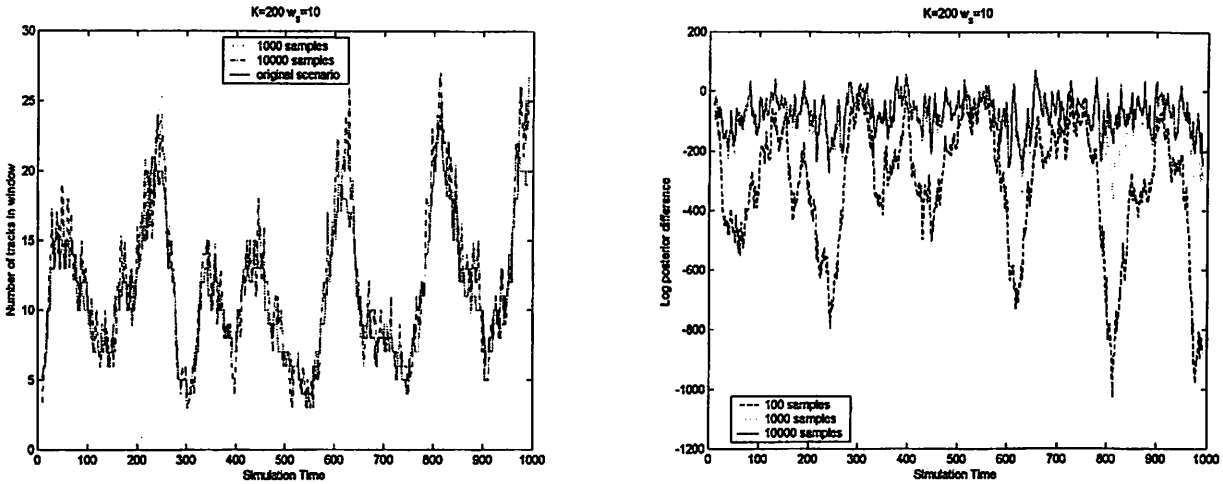
(a) (b)

Figure 3.18: Test case 1 ($K = 100, w_s = 10$) (a) number of tracks; (b) relative log posterior



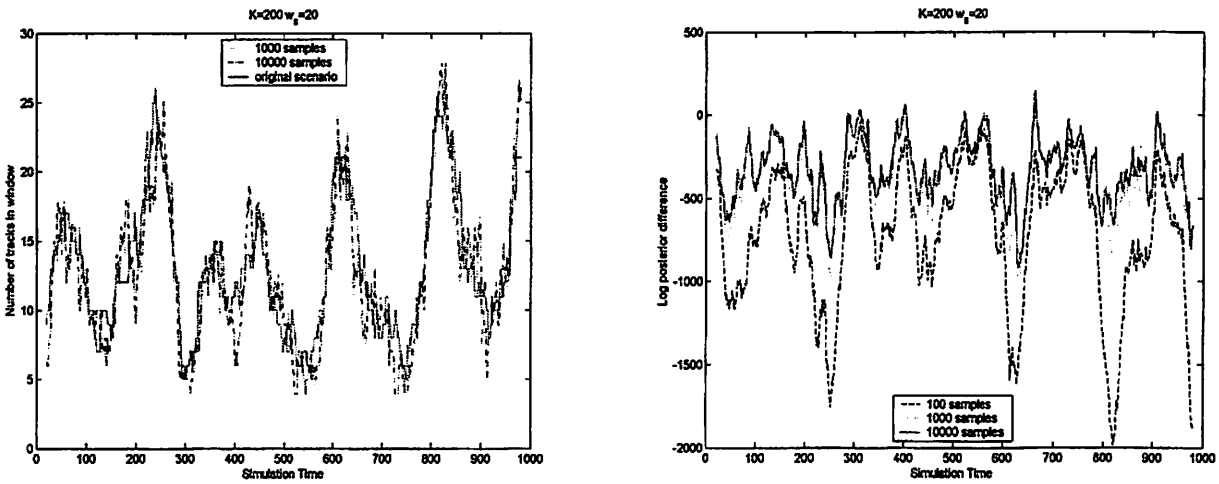
(a) (b)

Figure 3.19: Test case 1 ($K = 100, w_s = 20$) (a) number of tracks; (b) relative log posterior



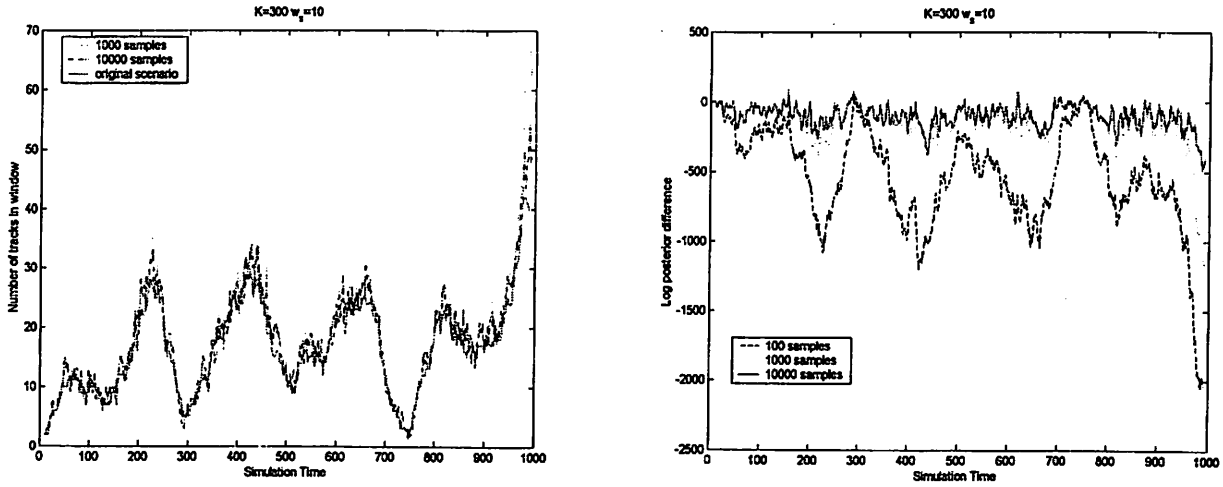
(a) (b)

Figure 3.20: Test case 2 ($K = 200, w_s = 10$) (a) number of tracks; (b) relative log posterior



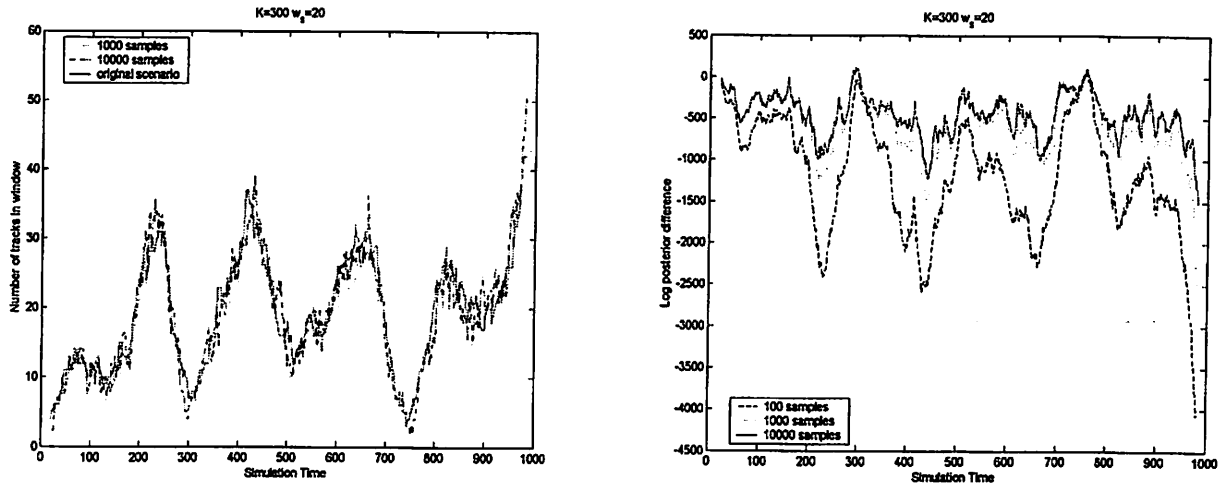
(a) (b)

Figure 3.21: Test case 2 ($K = 200, w_s = 20$) (a) number of tracks; (b) relative log posterior



(a) (b)

Figure 3.22: Test case 3 ($K = 300, w_s = 10$) (a) number of tracks; (b) relative log posterior



(a) (b)

Figure 3.23: Test case 3 ($K = 300, w_s = 20$) (a) number of tracks; (b) relative log posterior

3.6 Conclusions

The general (discrete-time) general multiple target tracking problem is defined and an MCMC algorithm is proposed. Our MCMC tracker, a data association algorithm, is very flexible and easy to incorporate any domain specific knowledge to make it more efficient. Instead of searching over the whole state space, the MCMC algorithm randomly searches over the space where the posterior is concentrated. Our simulation results show a remarkable performance of the MCMC tracker under the extreme conditions such as a large number of targets in a dense environment, low detection probabilities, and a large number of false alarms. We have shown that the algorithm can be applied as an online, real-time algorithm with an excellent performance.

Bibliography

Akaike, H. (1997). Information theory and an extension of the maximum likelihood principle. *2nd Inter. Symp. on Information Theory* (pp. 267–281).

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to mcmc for machine learning, machine learning. *Machine Learning*, vol. 50 (pp. 5–43).

Bar-Shalom, Y. & Fortmann, T. (1988). *Tracking and Data Association.* San Diego, CA: Mathematics in Science and Engineering Series 179 Academic Press.

Beichl, I. & Sullivan, F. (2000). The metropolis algorithm. *Computing in Science and Engineering*, vol. 2(1) (pp. 65–69).

Brooks, S. & Giudici, P. (1998). Convergence assessment for reversible jump mcmc simulations. *Bayesian Statistics*, vol. 6: Oxford University Press.

Collins, J. & Uhlmann, J. (1992). Efficient gating in data association with multivariate distributed states. *IEEE Trans. Aerospace and Electronic Systems*, vol. 28(3).

Cox, I. (1993). A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, vol. 10(1) (pp. 53–66).

Cox, I. & Hingorani, S. (1994). An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *International Conf. on Pattern Recognition* (pp. 437–443).

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Phil. Trans. Roy. Soc. London, Ser. A 222* (pp. 309–368).

Gelman, A., Carlin, J., Stern, H., & Rubin, D. (1995). *Bayesian Data Analysis.* Chapman and Hall.

Gilks, W., Richardson, S., & Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice.* Interdisciplinary Statistics Series. Chapman and Hall.

Green, P. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, vol. 82 (pp. 711–732).

Hansen, M. & Yu, B. (2001). Model selection and minimum description length principle. *J. Amer. Statist. Assoc.*, vol. 96 (pp. 746–774).

Huang, T. & Russell, S. J. (1997). Object identification in a bayesian context. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 1276–1283).

Hue, C., Le Cadre, J.-P., & Perez, P. (2002). Sequential monte carlo methods for multiple target tracking and data fusion. *IEEE Trans. on Signal Processing*, vol. 50(2) (pp. 309–325).

Jerrum, M. & Sinclair, A. (1996). The markov chain monte carlo method: An approach to approximate counting and integration. D. Hochbaum (Ed.), *Approximations for NP-hard Problems.* PWS Publishing, Boston, MA.

Kass, R. & Raftery, A. (1995). Bayes factors. *Journal of the American Statistical Association*, vol. 90 (pp. 773–795).

Kurien, T. (1990). Issues in the design of practical multitarget tracking algorithms. Y. Bar-Shalom (Ed.), *Multitarget-Multisensor Tracking: Advanced Applications.* Artech House: Norwood, MA.

Li, X. R. & Jilkov, V. P. (2000). A survey of maneuvering target tracking. *In SPIE: Signal and Data Processing of Small Targets*, vol. 4048-22.

Morefield, C. L. (1971). Application of 0-1 integer programming to multitarget tracking problems. *IEEE Trans. on Automatic Control*, vol. 22 (pp. 3:302–312).

Murty, K. (1968). An algorithm for rankning all the assignments in order of increasing cost. *Operations Research*, vol. 16 (pp. 682–687).

Pao, L. (1993). Multisensor multitarget mixture reduction algorithms for tracking. *Proc. AIAA Guidance, Navigation, and Control Conf.* (pp. 28–37). Monterey, CA.

Papadimitriou, C. & Steiglitz, I. (1982). *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2003). Identity uncertainty and citation matching. *In Advances in Neural Information Processing Systems 15*: MIT Press.

Pasula, H., Russell, S. J., Ostland, M., & Ritov, Y. (1999). Tracking many objects with many sensors. *IJCAI-99* Stockholm.

Poore, A. (1995). Multidimensional assignment and multitarget tracking. *Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 19 (pp. 169–196).

Reid, D. (1979). An algorithm for tracking multiple targets. *IEEE Transaction on Automatic Control*, vol. 24(6) (pp. 843–854).

Richardson, S. & Green, P. (1997). On bayesian analysis of mixtures with unknown number of components (with discussion). *Journal of the Royal Statistical Society, B*, vol. 59 (pp. 731–792).

Robert, C., Ryden, T., & Titterington, D. (2000). Bayesian inference in hidden markov models through reversible jump markov chain monte carlo. *Journal of the Royal Statistical Society, B*, vol. 62 (pp. 57–75).

Russell, S. (2001). Identity uncertainty. *In Proc. IFSA-01* Vancouver.

Schulz, D., Burgard, W., Fox, D., & Cremers, A. (2001). Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. *Proc. of the IEEE International Conference on Robotics and Automation (ICRA).*

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, vol. 6 (pp. 461–464).

Sittler, R. (1964). An optimal data association problem on surveillance theory. *IEEE Trans. on Military Electronics*, vol. MIL-8 (Apr) (pp. 125–139).

Stein, J. & Blackman, S. (1975). Generalized correlation of multi-target track data. *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-11(6) (pp. 1207–1217).

Streit, R. & Luginbuhl, T. (1994). Maximum likelihood method for probabilistic multi-hypothesis tracking. *Proc. SPIE*, vol. 2235 (pp. 394–405).

Valiant, L. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, vol. 8 (pp. 189–201).