Monte Carlo Methods for Multiple Target Tracking and Parameter Estimation

Daniel Duckworth

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-68
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-68.html

May 9, 2012

Acknowledgement

Stuart Russell

Monte Carlo Methods for Multiple Target Tracking and Parameter Estimation

Daniel Duckworth

May 9, 2012

Contents

Abstract

Multiple Target Tracking (MTT) is the problem of identifying and estimating the state of an unknown, time-varying number of targets. A successful algorithm will identify how many unique targets have existed, at what times they were active, and what sequence of states they followed when active.

This work presents two novel algorithms for MTT, Particle Markov Chain Monte Carlo Data Association (PMCMCDA) and Particle Filter Data Association (PFDA). These algorithms consider MTT in a Bayesian Framework and seek to approximate the posterior distribution over track states, data associations, and model parameters by combining Markov Chain Monte Carlo and Particle Filtering to perform approximate inference. Both algorithms are evaluated experimentally on two pedagogical examples, and proofs of convergence in the limit of infinite samples are given.

Chapter 1

Introduction

As humans, identifying and tracking the objects that surround is such an essential part of our daily lives that we do so almost automatically. For example, when crossing an intersection, we wait for the proper traffic light to turn green, avoid other pedestrians and bicyclists, and check to see if turning vehicles are respecting our right of way. Each of these actions requires not only identifying moving objects but also *modeling how they interact* such that we may predict their movements and plan our own. Multiple Target Tracking (MTT) is the problem of computationally constructing such a model.

Let us consider what an ideal MTT algorithm provided perfect measurements would be able to do. First and foremost, the algorithm should be able to track the behaviour of all targets in view. Secondly, it should be able to recognize new targets as they enter as well as identify when a target is leaving. Third, it should be able to distinguish targets as separate entities.

This simple definition hides much of the complexity of MTT. A complete algorithm, for example, does not know beforehand how many targets need to be tracked, and thus must be ready for any number of new targets at any point in time. In addition, targets can become obscured even while they are still of interest, and the algorithm must be able to track them even when they cannot be viewed directly. Targets may also interact, altering each others' behavior. Finally, the observations given to an algorithm are not segmented into unique, identifiable objects beforehand, and thus how observations correspond to targets must be inferred as well.

Let us consider a concrete example. Suppose we have camera at the top of a skyscraper overlooking an intersection, as in Figure 1.1. The camera is tasked with identifying drivers who speed

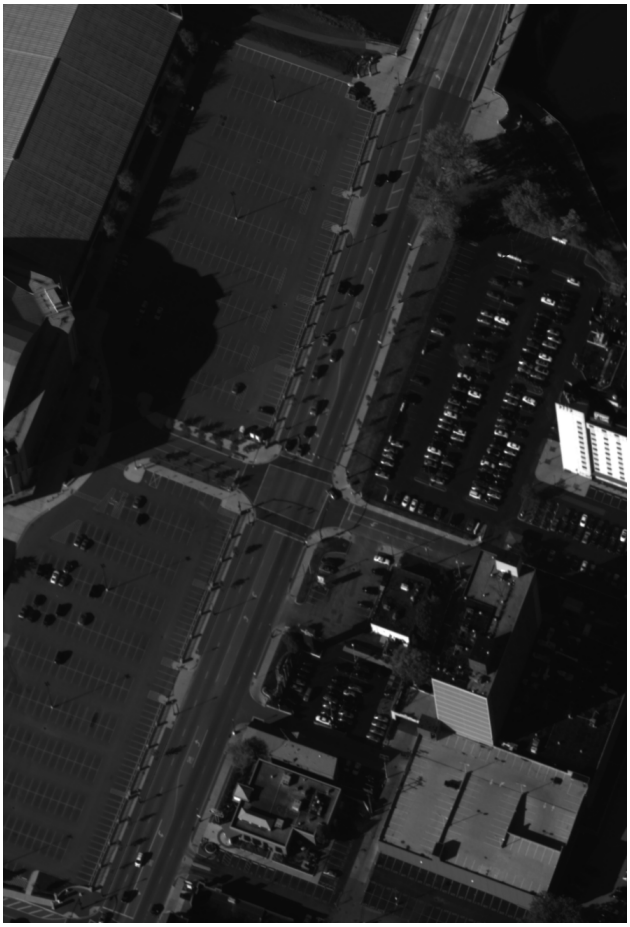
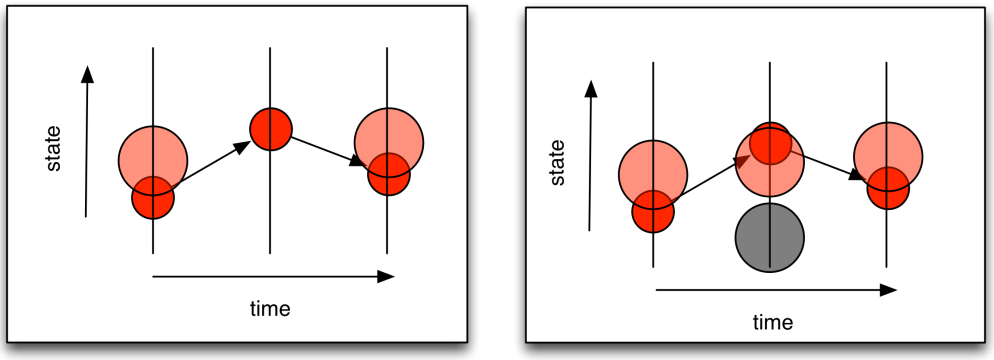


Figure 1.1: A camera overlooking an intersection. An MTT algorithm must track the position and velocity of all moving cars.

or run a red light. In order to do so, the camera must first identify in each frame the speed and velocity of each new car entering the scene. This is not as simple as looking for movement at the top or bottom as cars may also come in and out of parking spaces. The dark shadows on the left and right sides also obscure any vehicle passing under them. When multiple cars approach the intersection, they line up in an orderly fashion, but when the road is clear they move independently. Finally, many cars look almost identical, and the camera must ascertain what parts of the video correspond to each individual car. This is particularly challenging when two cars enter an obscured area simultaneously or when the video frame rate is too low to distinguish motion unambiguously.

In order to build robots capable of interacting intelligently with the world around them, we first need algorithms capable of understanding the world they live in. This thesis introduces two new algorithms that take a step in that direction.



(a) Detection Failure (b) False Detection

Figure 1.2: Detection errors an MTT algorithm must correct. Small circles are true positions and larger, transparent circles of the same color are detections they generated. The larger grey circle is a detection generated by clutter.

1.1 Past Work

Modern MTT algorithms must work in a variety of environments with a range of different sensor types. For example, one may be tasked with tracking the positions of people in a hallway given a video feed. What if one is then asked to perform the same task, but with an array of microphones? To generalize to a wider range of sensor inputs, MTT algorithms are often designed with the assumption that a secondary algorithm will transform the raw sensor signal into sets of **Detections** discretized in time. Ideally, each detection will correspond to a potentially noisy observation of exactly one target (e.g., a car), but there may be times when clutter is misidentified as a target (**False Detection**) and when a target is not recognized at all (**Detection Failure**). It is then up to the MTT algorithm to characterize each target's behaviour and correct for these mistakes.

Having transformed observations into sets of detections, an MTT algorithm must now also identify which detections corresponds to which target, a task known as **Data Association**. Formally, we require that the set of all detections across all time is partitioned such that,

1. A target can only be associated with one detection at any given point in time
2. Each detection is associated with either one target or clutter

The number of legal data associations is exponential in the number of targets even when the

latter is fixed, and identifying high probability data associations can become a task perhaps more difficult than estimating the states of the targets themselves.

MTT algorithms can be categorized based on how they handle data association and target state estimation. Algorithm designers may eschew data associations altogether and work with the raw input signal, heuristically choose a single 'reasonably likely' data association to rely on, or maintain a collection of likely associations pruned in some judicious way. When estimating state, designers may assume additive Gaussian noise in order to apply some form of the Kalman Filter or may instead handle a wider range of models by employing approximate methods such as those described in Chapter 3.

However, each of these algorithms makes sacrifices in one aspect in order to gain more flexibility in the other. The original Multiple Hypothesis Tracking (MHT) algorithm [Blackman, 1986], for example, assumes each target moves independently under Linear-Gaussian dynamics. This allows the algorithm to efficiently estimate the posterior distribution for each target given a data association, and thus MHT is able to enumerate many likely data associations. Breitenstein et al. [2009] on the other hand allows one to define an arbitrary target dynamics model but restricts itself to a single, heuristically chosen data association in order to employ isolated Particle Filters for each target. In addition, both algorithms assume model parameters are known and fixed before the algorithm is initialized and are unable to recover them otherwise.

1.2 Overview

This work presents two new algorithms that give one the ability to infer target states, data associations, and model parameters simultaneously without being restricted to any form of dynamics model. We do so by framing all of the above in single Bayesian model upon which we employ Particle Markov Chain Monte Carlo [Andrieu et al., 2010] and Resample-Move Particle Filters [Gilks and Berzuini, 2001] to jointly infer all desired variables.

In order to understand these algorithms, we first present in Chapter 2 the tools necessary to describe a Bayesian model with an unknown, time-varying number of targets and detections with identity uncertainty. We present partially observable Markov models as the rough Bayesian framework within which our algorithms operate. We describe the insufficiencies of this model,

then present Contingent Bayesian Networks capable of describing the context-specific independence necessary for defining the joint distribution. We finally formulate Multiple Target Tracking as a Contingent Bayesian Network, then show via the work of Milch [2006] and Goodman et al. [2008] that this distribution is well defined.

In Chapter 3, we review Monte Carlo algorithms employed in performing approximate inference in Bayesian Models. We describe fundamental methods such as Importance Sampling, Markov Chain Monte Carlo, and Particle Filters, as well as two recent algorithms, Particle MCMC and the Resample-Move Particle Filter, for combining MCMC and Particle Filters. These latter two will form the foundation of our new algorithms.

In Chapter 4, we review the latest work in Multiple Target Tracking (MTT). We describe the modern algorithms employed, their strengths and weaknesses, and define a metric by which tracking performance can be evaluated.

Finally, in Chapter 5, we define the conditional distributions required to fully specify the probabilistic model upon which inference is being performed and introduce two novel algorithms, Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA). We prove correctness of these two algorithms, then follow with experimental results on the performance of these methods in two different dynamics models: a discrete, random walk and a two-dimensional road network.

Chapter 2

Stating the Model

Both Particle Markov Chain Monte Carlo Data Association (PMCMCDA) and Particle Filter Data Association (PFDA) approximate the posterior distribution of a particular probability model. Before stating the model, let us state the problem of MTT more formally,

Definition 1. *A Multiple Target Tracking (MTT) algorithm is given, at each time step t, a set of indexed detections $\{y_t^{(i)}\}_{i=1}^{N_t^y}$ where N_t^y is the number of detections at time t. Ideally each detection corresponds to a noisy observation attributable to precisely one target, but the algorithm must be able to tolerate detection failures and false detections (see Section 1.1) as well. The goal of the algorithms is to recover,*

1. *t_k^{start} and t_k^{end}, the times target k enters and leaves the field of view, for all observable targets k. If $t_k^{start} \leq t \leq t_k^{end}$, we say that target k is 'active,' 'in view', or 'alive.' A target cannot become active again after leaving the field of view.*

2. *$x_t^{(k)}$, the unknown state of target k at time t, for all time t and observable targets k. If a target is outside of the field of view at time t, let $x_t^{(k)} = null$.*

3. *data associations $a_t : \mathbb{N} \to \mathbb{N}$ such that $a_t(i) = k$ if $y_t^{(i)}$ was a detection generated by target k at time t for all time.*

In addition to the definition of a_t, we require that k be in the range of a_t only when $t_k^{start} \leq t \leq t_k^{end}$. We denote $a_t(i) = 0$ if detection i at time t was generated by clutter. Furthermore, we require

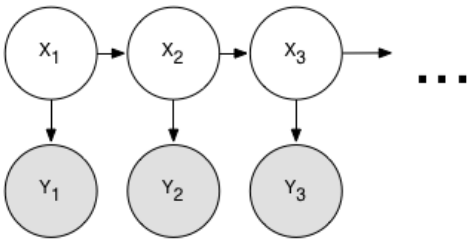


Figure 2.1: Hidden Markov Model. Observed variables are colored grey and unobserved variables are colored white.

that with the exception of 0, $a_t(i)$ *is one-to-one. This allows us to denote* $a_t^{-1}(k) = i$ *when detection* i *was generated by target* k *at time* t *and to set* $a_t^{-1}(k) = 0$ *if* k *did not generate a detection.*

Finally, an MTT algorithm need not give a point estimate but may instead provide a distribution over possible values for unknown variables $\{x_t^{(k)}\}_{k,t}$, $\{a_t\}_t$, *and* $\{(t_k^{start}, t_k^{end})\}_k$.

With this definition in mind, we now seek to construct a Bayesian model we may employ to infer $\{x_t^{(k)}\}_{t,k}$, $\{t_k^{\text{start}}, t_k^{\text{end}}\}_k$ and $\{a_t\}_t$ given $\{y_t^{(i)}\}_{t,i}$ for all t, k, and i. To do so, we will first describe the Hidden Markov Model (HMM), a sequential model where the past and future are conditionally independent given the present. By formulating MTT as an HMM, we will be able justify the concept of 'Filtering.' We will then explain why the HMM formalism alone is insufficient, and will then present a new formulation using Contingent Bayesian Networks. This new representation will allow us to take advantage of context-specific conditional independence assumptions necessary for representing samples from the model in finite space and for performing tractable approximate inference.

2.1 Bayesian Filtering

We begin our discussion by presenting the **Hidden Markov Model** (HMM), a template for describing a partially observable Markov process. This template is the basis of many models where variables are naturally described sequentially; applications include single target tracking, speech recognition, part-of-speech tagging, click-through prediction, and character recognition. The HMM is defined by a Markov chain with an unknown hidden state $X_t \sim P(X_t|X_{t-1})$ and noisy observations $Y_t \sim P(Y_t|X_t)$. A graphical model representation is given in 2.1.

This model is special in that there exists a well-known, recursive method for estimating X_t

given $Y_{1:t} := \{Y_1, \dots Y_t\}$ which is as follows,

$$P(X_{t+1}|Y_{1:t+1}) = \frac{P(X_{t+1}, Y_{t+1}|Y_{1:t})}{P(Y_{t+1}|Y_{1:t})} \tag{2.1}$$

$$= \frac{P(Y_{t+1}|X_{t+1})P(X_{t+1}|Y_{1:t})}{\int P(Y_{t+1}|X_{t+1})P(X_{t+1}|Y_{1:t})dX_{t+1}} \tag{2.2}$$

$$= \frac{P(Y_{t+1}|X_{t+1}) \int P(X_{t+1}|X_t)P(X_t|Y_{1:t})dX_t}{\int P(Y_{t+1}|X_{t+1}) \left[\int P(X_{t+1}|X_t)P(X_t|Y_{1:t})dX_t\right] dX_{t+1}} \tag{2.3}$$

In other words, a simple algorithm for computing $P(X_{t+1}|Y_{1:t+1})$ for all X_{t+1} is as follows,

1. Propagate: calculate $P(X_{t+1}|Y_{1:t}) = \int P(X_{t+1}|X_t)P(X_t|Y_{1:t})dX_{1:t}$
2. Reweight: calculate $P(X_{t+1}, Y_{t+1}|Y_{1:t}) = P(Y_{t+1}|X_{t+1})P(X_{t+1}|Y_{1:t})$
3. Normalize: calculate $P(Y_{t+1}|Y_{1:t}) = \int P(X_{t+1}, Y_{t+1}|Y_{1:t})dX_{t+1}$ and divide $P(X_{t+1}, Y_{t+1}|Y_{1:t})$ by it to get $P(X_{t+1}|Y_{1:t+1})$

The above algorithm is known as 'Filtering' and, in spite of its simplicity, can become alarmingly complex depending on what X_t and Y_t represent. We have 'swept under the rug' the difficulty of calculating the above integrals whose solutions are specific to $P(X_{t+1}|X_t)$ and $P(Y_t|X_t)$.

There exist two primary cases where the above algorithm can be implemented exactly. The first is when there are finitely many values for X_t; then the integrals above turn to sums, and as long as the number of unique values for X_t is small, we may implement the algorithm word for word. The second is when $X_{t+1} \in \mathbb{R}^d$ is defined linearly in terms of X_t and Y_t linearly in terms of X_t, both with additive Gaussian noise (i.e., $X_{t+1} = AX_t + \epsilon_x$ and $Y_t = CX_t + \epsilon_y$ where A and C are matrices and ϵ_x and ϵ_y are drawn from a Multivariate Normal distribution with known mean and covariance). While less simple than the discrete case, a closed form solution exists known as the Kalman Filter.

We have taken X_t so far to be something atomic and undecomposable, but often we may find X_t to be made up of smaller, lower dimensional variables with additional conditional independence assertions we can make. With this in hand, we may then design algorithms that can more effectively perform inference. This is the basis of **Dynamic Bayesian Networks** (DBNs) [Dean and Kanazawa, 1989, Murphy, 2002]. A simple example of this is that of a two independently moving robots; then we can represent the state as $X_t := \{X_{1,t}, X_{2,t}\}$ as in Figure 2.2.

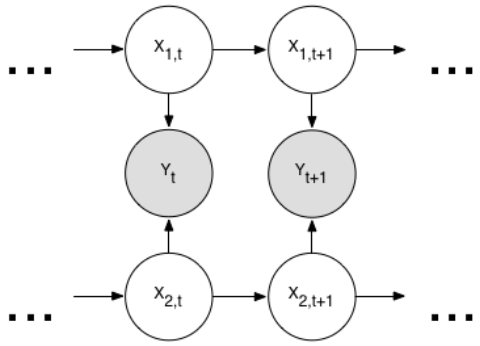


Figure 2.2: Decomposing the hidden state of two independently moving robots.

Let us now consider defining Multiple Target Tracking (MTT) within a Hidden Markov Model (HMM) framework. Let $X_t = (\{x_t^{(k)}\}_{k\in\mathbb{N}}, a_t)$ and $Y_t = \{y_t^{(i)}\}_{i\leq N_t^y}$. Note that t_k^{start} and t_k^{end} are fully determined by $x_t^{(k)}$ being null or otherwise. In particular we may derive their values via,

$$t_k^{\text{start}} = \inf\{t : x_t^{(k)} \neq \text{null}\} \tag{2.4}$$

$$t_k^{\text{end}} = \inf\{t : t_k^{\text{start}} < t,\, x_t^{(k)} = \text{null}\} \tag{2.5}$$

With X_t and Y_t now defined, we may hope to apply some form of Filtering to infer the current state given past detections. However, it is immediately clear that no tractable algorithm can be derived from this representation; indeed, even representing a realization of X_t would require an unbounded amount of memory as there are an unbounded number of targets. We may alternatively hope to decompose X_t into a Dynamic Bayesian Network in hopes of exploiting other conditional independence assumptions, but data association uncertainty requires the parents of each $y_t^{(i)}$ to be the infinite set $\{x_t^{(k)}\}_{k\in\mathbb{N}}$. This alone makes the DBN representation ill defined as it prevents one from defining the topological ordering of variables necessary for constructing a joint distribution. Figure 2.3 presents a graphical model representation of this faulty model.

While the HMM formalism is insufficient for developing a reasonable representation of MTT, it does provide us with the concept of Filtering. While we may not be able to apply it directly, Filtering will be key to the implementation of Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA).

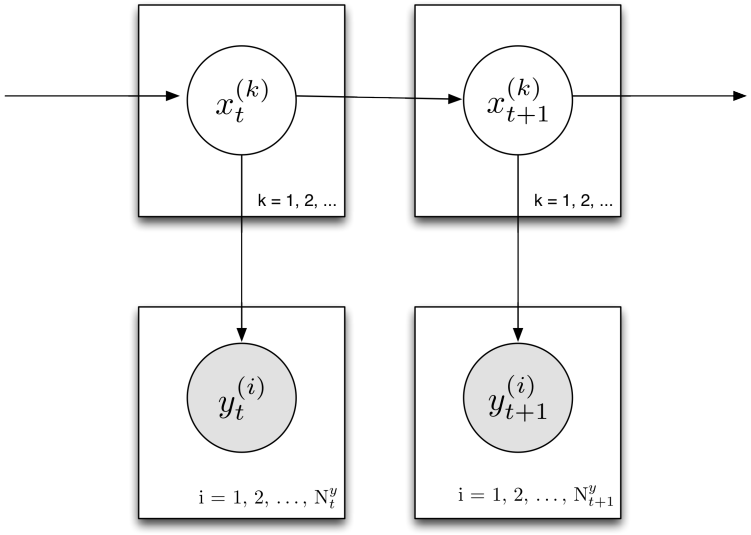


Figure 2.3: A graphical model for Multiple Target Tracking as as Dynamic Bayesian Network. This DBN does not correspond to a joint distribution as each $y_t^{(i)}$ has an infinite number of parents.

2.2 Contingent Bayesian Networks

Before continuing, we must identify precisely what Dynamic Bayesian Networks lack and what is needed to define a proper joint distribution. In particular, we begin by noting that although any target could potentially influence detection $y_t^{(i)}$, at most one can have any direct influence. In fact, by knowing a_t, we know precisely which target generated $y_t^{(i)}$, and all other targets become irrelevant. Similarly, we may decide not to model target interactions outside of our field of view. If we further limit queries to targets in the same way, we may be able to safely ignore all targets except those directly in our field of view.

Each of the previously stated conditions is some form of **Context-Specific Independence**. In other words, conditioned on the particular value of some variables, we may be able to make additional conditional independence assumptions that do not always exist. For example, if $a_t^{-1}(k') = i$, we know that $y_t^{(i)}$ is independent of all $x_t^{(k)}$ for all $k \neq k'$. If we know which targets are 'alive' (that is, within our field of vision), we may integrate away the states of all targets except those we may potentially observe directly. By doing so, we limit the number of variables of interest in our model to a finite number at any given time.

Finally, we note that the indexing used for targets k and detections i is entirely arbitrary and exists only to uniquely identify the objects in the model. In other words, whether target 5 or 500

started at time t is irrelevant, our only concern is that a new track started. By this simple idea, we may exploit exchangeability of indices to aggregate realizations of the model which are identical up to track and detection reindexing.

Both context-specific independence and exchangeable indices are two concepts which cannot be expressed in vanilla Bayesian Networks. Thus, we instead consider **Contingent Bayesian Networks** (CBNs) [Milch, 2006], a modified form of Bayesian Networks that introduces a new tool, labeled edges.

Labeled edges allows us to specify when an edge $v \to u$ is active in a Bayesian Network given some other variables $\{v'\}$. One may think of these labeled edges as choosing which version of a Bayesian Network to use. Care must be taken when designing CBNs to avoid edge labels that make deciding an ordering among variables impossible. In particular, a CBN is well defined as long as

1. No consistent path in the CBN forms a cycle
2. No consistent path has a variable with infinitely many ancestors
3. No variable has a consistent, infinite number of parents

These three conditions imply a CBN has a supportive numbering (i.e., the variables can be made into a DAG for each consistent realization of the model) and in consequence a well defined joint probability distribution exists [Milch, 2006, Theorem 3.17].

Let us now construct a Contingent Bayesian Network model of Multiple Target Tracking (Figure 2.4). Let alive_t be the set of indices of all targets that are in view at time t, $\#\text{new}_t$ be the number of new tracks at time t, and $\#\text{clut}_t$ be the number of clutter detections at time t. We will assume that both of the latter depend only on the time step t. As we are only interested in active tracks, we will implicitly integrate away the states of all targets outside of view and only consider $\{x_t^{(k)}\}_{k \in \text{alive}_t}$. As we have assumed targets outside of view do not interact with other targets until they come within view, $\{x_t^{(k)}\}_{k \in \text{alive}_t}$ is conditionally independent of all other targets except those in view at time step $t-1$. Furthermore, as only visible objects can generate detections, a_t is conditionally independent of all targets except those in $\{x_t^{(k)}\}_{k \in \text{alive}_t}$. For the same reasons, $\{y_t^{(i)}\}_{i \leq N_t^y}$ is conditionally independent of all other targets except those in $\{x_t^{(k)}\}_{k \in \text{alive}_t}$. Finally, since the indexing of tracks is exchangeable, alive_t is dependent only on alive_{t-1} and $\{x_{t-1}^{(k)}\}_{k \in \text{alive}_{t-1}}$

as new tracks will be given arbitrary, unique indices.

We present an alternative, procedural view of the Multiple Target Tracking CBN in Algorithm 1. This random function defines how one may sample one time slice from the generative model defined by Figure 2.4 given the set of active targets and their respective states at the previous time step. As this procedure may also be encoded in Church, a stochastic variant of Lisp, it also serves as proof of the existence of both a joint distribution over all variables and a posterior distribution over all other variables given detections [Goodman et al., 2008, Lemma 2.2, Theorem 2.3].

While the model presented here is not a Hidden Markov Model (HMM), it retains the same Markov property between time steps. In other words, if we define

$$X_t = (\text{alive}_t, \#\text{new}_t, \#\text{clut}_t, a_t, \{x_t^{(k)}\}_{k \in \text{alive}_t}) \tag{2.6}$$

$$Y_t = \{y_t^{(i)}\}_{i \leq N_t^y} \tag{2.7}$$

then all of the conditional independence assumptions in an HMM are satisfied. This will allow us to perform (approximate) filtering, the core idea in Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA).

The model presented in Figure 2.4 is still incomplete; after all, we have yet to define the forms of the conditional distributions. As these are problem-specific, we leave the definition of the complete model to Section 5.1.

Algorithm 1 Sampling from the Contingent Bayesian Network version of Multiple Target Tracking

Input: alive_t, $\{x_t^{(k)}\}_{k \in \text{alive}_t}$

Output: $\#\text{new}_{t+1}$, $\#\text{clut}_{t+1}$, alive_{t+1}, a_{t+1}, $\{y_{t+1}^{(i)}\}_{i \leq N_{t+1}^y}$

1: Sample $\#\text{new}_{t+1} \sim P(\cdot)$ and $\#\text{clut}_{t+1} \sim P(\cdot)$
2: Sample $\text{alive}_{t+1} \sim P(\cdot|\text{alive}_t, \{x_t^{(k)}\}_{k \in \text{alive}_t}, \#\text{new}_{t+1})$
3: Sample $\{x_{t+1}^{(k)}\}_{k \in \text{alive}_{t+1}} \sim P(\cdot|\text{alive}_t, \text{alive}_{t+1}, \{x_t^{(k)}\}_{k \in \text{alive}_t})$
4: Sample $a_{t+1} \sim P(\cdot|\text{alive}_{t+1}, \{x_{t+1}^{(k)}\}_{k \in \text{alive}_{t+1}}, \#\text{clut}_{t+1})$
5: Sample $\{y_t^{(i)}\}_{i \leq N_{t+1}^y} \sim P(\cdot|a_t, \{x_{t+1}^{(k)}\}_{k \in \text{alive}_{t+1}})$

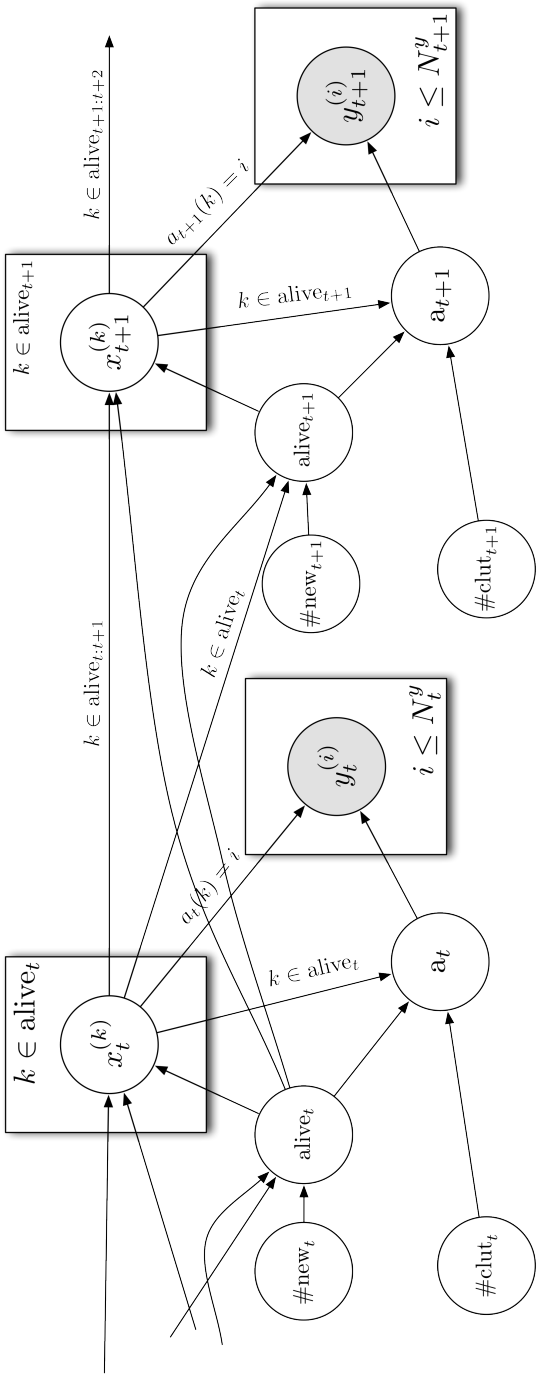


Figure 2.4: Multiple Target Tracking represented as a Conditional Bayesian Network.

Chapter 3

Monte Carlo Methods Today

With the basic form of the Bayesian model for Multiple Target Tracking (MTT) defined in Section 2.2, we now turn to methods for inferring the unobserved variables given sets of detections at each time step t. While there exists a range of algorithms for performing exact inference in Bayesian Networks, we cannot hope that any will be efficient in our model formulation. Even if all detections and target states are discrete and the set of active targets is known and fixed, the number of potential data associations is exponential in the number of active targets. Instead, we will turn to methods for approximate inference.

The fundamental goal of inference is to calculate $\mathbb{E}_\pi[f(X)] = \int f(X)\pi(X)dX$ for some function f and distribution π. Often, we will be able to evaluate $\bar{\pi}(x) = C\pi(X)$ pointwise for some unknown constant C but will be unable to evaluate the previous integral exactly. A common special case of this goal is when $\bar{\pi}(X) = P(X, Y = y)$, $\pi(X) = P(X|Y = y)$, and $f(X) = \mathbb{1}[X \in A]$.

Our modus operandi for approximating these integrals will be **Monte Carlo Methods**; that is, methods based on random sampling. By generating more and more samples, we hope to better approximate $\mathbb{E}_\pi[f(X)]$, and ideally guarantee that our approximation converges to its true value as the the number of samples approaches infinity.

As an aside, we will often write $\mathbb{E}[f(X)]$ instead of $\mathbb{E}_\pi[f(X)]$ except when it is unclear which distribution we are taking an expectation with respect to.

Looking forward, we will begin by reviewing three standard methods in Monte Carlo: Importance Sampling (IS), Markov Chain Monte Carlo (MCMC), and the Particle Filters (PF). We will then describe Reversible Jump Markov Chain Monte Carlo (RJMCMC), a method extending

MCMC to distributions of variable dimension. We conclude with two newer methods that combine MCMC and PFs, the Resample-Move Particle Filter and Particle Markov Chain Monte Carlo (PMCMC). These final two methods form the basis of Particle Filter Data Association (PFDA) and Particle MCMC Data Association (PMCMCDA) respectively.

3.1 Importance Sampling (IS)

The first such method we will describe is perhaps the simplest and most intuitive. Suppose we are able to sample from $\pi(X)$ directly. Then by generating $X_i \sim \pi(X)$, we will always have an unbiased estimate for $\mathbb{E}_\pi[f(X)]$ given by,

$$\mathbb{E}[f(X)] = \int f(x)\pi(x)dx \tag{3.1}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} f(X_i) \tag{3.2}$$

Indeed, $\mathbb{E}[f(X_i)] = \mathbb{E}[f(X)]$ for all i, so the average does as well. The Law of Large Numbers also guarantees convergence to $\mathbb{E}[f(X)]$ as $N \to \infty$ as long as the expectation is defined in the first place (i.e., $\mathbb{E}[|f(X)|] < \infty$).

Of course, sampling from $\pi(X)$ is a luxury we rarely have, so suppose instead we are only able to evaluate $\pi(X)$ pointwise and have some other distribution $q(X)$ we can draw samples from and evaluate pointwise. Henceforth we will refer to q as a **Proposal Distribution** as it is the source from which samples are proposed. We may then derive an unbiased estimate for $\mathbb{E}_\pi[f(X)]$ by sampling $X_i \sim q(X)$ and 'weighting' our samples by $w(x) = \frac{\pi(x)}{q(x)}$ like so,

$$\mathbb{E}_\pi[f(X)] = \int f(x)\pi(x)dx \tag{3.3}$$

$$= \int f(x)\frac{\pi(x)}{q(x)}q(x)dx \tag{3.4}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} w(X_i)f(X_i) \tag{3.5}$$

Intuitively, the weight function $w(X)$ 'corrects' our samples based on how likely they are under $\pi(X)$. One must be careful to ensure that $\pi(X) > 0 \Rightarrow q(X) > 0$ as violating this would imply that $\mathbb{E}_q[w(X)f(X)] = \infty$ (the Law of Large Numbers then no longer applies), but otherwise choice of

$q(X)$ is unrestricted. However, one's choice of q can greatly affect how accurate one's approximation is; in particular, the higher the variance, the more unreliable the estimates. The variance of the approximation can be derived to be,

$$\begin{aligned}
\mathrm{Var}_q(f(X)w(X)) &= \mathbb{E}_q[(f(X)w(X))^2] - \mathbb{E}_q[f(X)w(X)]^2 && (3.6)\\
&= \int f(x)^2 \frac{\pi(x)^2}{q(x)^2} q(x)dx - \left(\int f(x)\frac{\pi(x)}{q(x)} q(x)dx\right)^2 && (3.7)\\
&= \int f(x)^2 \frac{\pi(x)}{q(x)} \pi(x)dx - \left(\int f(x)\pi(x)dx\right)^2 && (3.8)\\
&= \mathbb{E}_\pi[f(X)^2 w(X)] - \mathbb{E}_\pi[f(X)]^2 && (3.9)
\end{aligned}$$

The dependence of that variance on the query function f is unfortunate, and it is thus often assumed $f(X) = \mathbb{1}[X \in A] \leq 1$ in which case $\mathrm{Var}_q(w(X)) = \mathbb{E}_\pi[w(X)] - 1$. If we are able to choose $q(x) = \pi(x)$, we see that $w(x) = 1$ for all x and thus $\mathrm{Var}_q(w(X)) = 0$. Even if we cannot sample from $\pi(x)$ directly, using proposal distributions as 'close' to $\pi(x)$ as possible is valuable; indeed, if $w(X) \leq W_{\max}$ for all X and $f(X) = \mathbb{1}[X \in A]$, then we may apply Hoeffding's Inequality,

$$P\left(\left|\frac{1}{N}\sum_{i=1}^{N} w(X_i)f(X_i) - \mathbb{E}_\pi[f(X)]\right| \geq \epsilon\right) \leq 2\exp\left(\frac{-2\epsilon^2 N}{W_{\max}}\right) \tag{3.10}$$

This shows that our estimate for $\mathbb{E}[f(X)]$ converges *exponentially* in the number of samples with a constant depending only on how large $w(X)$ can be. While the upper bound $W_{\max}$ can rarely be evaluated directly, it gives us proof that Importance Sampling is a sound method.

Finally, we consider the case where we may only evaluate $\bar{\pi}(x) = C\pi(X)$ pointwise for unknown C and use proposal distribution $q(X)$ from which we can both sample and evaluate pointwise. In this case, we can design an estimate of $\mathbb{E}_\pi[f(X)]$ by *self-normalizing*,

$$\begin{aligned}
\mathbb{E}_\pi[f(X)] &= \frac{\int f(x)\bar{\pi}(x)dx}{\int \bar{\pi}(x)dx} && (3.11)\\
&= \frac{\int f(x)\frac{\bar{\pi}(x)}{q(x)} q(x)dx}{\int \frac{\bar{\pi}(x)}{q(x)} q(x)dx} && (3.12)\\
&\approx \frac{\frac{1}{N}\sum_{i=1}^{N} w(X_i)f(X_i)}{\frac{1}{N}\sum_{i=1}^{N} w(X_i)} && (3.13)
\end{aligned}$$

One important side-effect of self-normalization is that we now have a biased approximation to

$\mathbb{E}_\pi[f(X)]$. To see this, simply consider the case when $N = 1$; the approximation always yields one sample with expectation $\mathbb{E}_q[f(X_1)]$ instead of $\mathbb{E}_\pi[f(X_1)]$. In spite of this, it can be shown [Whiteley, 2011] that the approximation is still **consistent**; that is, it converges to $\mathbb{E}_\pi[f(X)]$ as $N \to \infty$.

While Importance Sampling (IS) is attractive for its simplicity, it is often only practical to use when $\pi(X)$ is no more than 10 dimensions. Identifying provably efficient $q(X)$ requires significant mathematical insight, and choices based on intuition rarely scale. However, IS is the basis of Particle Filters (PFs) which have proven effective in many high-dimensional scenarios, particularly those involving HMM-style models.

3.2 Markov Chain Monte Carlo (MCMC)

The next method we shall describe is considerably more complex than Importance Sampling but has been invaluable in performing inference in high dimensional distributions. **Markov Chain Monte Carlo** works by generating a sequence of variables from a specially crafted Markov Chain. Unlike Importance Sampling where each X_i is independent, MCMC relies on the fact that after a sufficiently long time, the output of this MCMC chain has the same distribution as $\pi(X)$.

Suppose once again we have target distribution $\pi(X) = \bar{\pi}(X)/C$ for which we may only evaluate $\bar{\pi}(X)$ pointwise. Suppose further that $X = (X^{(1)}, \ldots, X^{(n)})$ where n is in the range of hundreds to thousands. While one may use intuition to choose an adequate proposal distribution for importance sampling when n is small (proving how 'close' $\pi(X)$ and $q(X)$ are is rarely possible), designing $q(X)$ to match $\pi(X)$ in higher dimensions becomes increasingly difficult.

Instead of generating each sample X_i independently, MCMC works by generating a sequence of samples where each X_i depends on X_{i-1} via an MCMC kernel $K(X_i|X_{i-1})$. Typically, K will generate X_i that differs from X_{i-1} in only a handful of components by sampling X' from an MCMC proposal distribution $q(X'|X_{i-1})$ and 'accepting' or 'rejecting' this proposal based on a how likely it is in under $\bar{\pi}(X)$ compared to X_{i-1}. This is the method used in one of the most popular MCMC techniques called the **Metropolis-Hastings** Algorithm [Metropolis et al., 1953]. A single iteration of this algorithm is presented in Algorithm 2.

In order to understand MCMC, we must first explain what it means for a Markov Chain defined by a transition distribution $K(X'|X)$ to have a **Stationary Distribution**. Mathematically, $\pi(X)$

Algorithm 2 One iteration of the Metropolis-Hastings algorithm

Input: input state X_i
Output: output state X_{i+1} such that $\mathbb{E}[f(X)] \approx^{!}_{N} \sum_i f(X_i)$ as $N \to \infty$
1: Sample $X' \sim q(X'|X_i)$
2: Calculate acceptance ratio

$$\alpha(X_i, X') = \min\left(\frac{\bar{\pi}(X')q(X_i|X')}{\bar{\pi}(X_i)q(X'|X_i)}, 1\right) \tag{3.14}$$

3: Sample $u \sim \text{Unif}(0, 1)$. If $u < \alpha(X_i, X')$ let $X_{i+1} = X'$, else $X_{i+1} = X_i$

Algorithm 3 One iteration of Gibbs Sampling

Input: input state $X_i = (X^{(1)}, \ldots, X^{(n)})$
Output: output state X_{i+1} such that $\mathbb{E}[f(X)] \approx^{!}_{N} \sum_i f(X_i)$ as $N \to \infty$
1: Sample index k uniformly from $1 \ldots n$
2: Sample $\overline{X}^{(k)} \sim P(X^{(k)}|X^{(1)}, \ldots X^{(k-1)}, X^{(k+1)}, \ldots, X^{(n)})$
3: Let $X_{i+1} = (X^{(1)}, \ldots X^{(k-1)}, \overline{X}^{(k)}, X^{(k+1)}, \ldots, X^{(n)})$

is the stationary distribution of a Markov Chain with transition density $K(X'|X)$ if $\pi(X') = \int K(X'|x)\pi(x)dx$ for all X'. In other words, if X is drawn from $\pi(X)$, then $X' \sim K(X'|X)$ is as well. Furthermore, if $X_i \sim K(X_i|X_{i-1})$ for $i = 1 \ldots n$, then for any starting distribution of X_0, the distribution of X_n approaches $\pi(X)$ as $n \to \infty$.

The question then becomes "Under what conditions does K have a stationary distribution?" and "How can we ensure K's stationary distribution is our desired $\pi(X)$?" One way of answering both is via **Detailed Balance**; that is, requiring that $\pi(X)K(X'|X) = \pi(X')K(X|X')$ for all X and X'. This is the approach taken in Metropolis-Hastings and **Gibbs Sampling** (Algorithm 3), another common MCMC method. For example, consider Algorithm 2. In this case,

$$K(X'|X) = \alpha(X, X')q(X'|X) + (1 - \alpha(X, X'))\delta(X = X') \tag{3.15}$$

We may verify detailed balance by assuming $\alpha(X, X') < 1$ (then $\alpha(X', X) = 1$) and observing,

$$\pi(X)q(X'|X)\alpha(X, X') = \pi(X')q(X|X')\alpha(X', X) \tag{3.16}$$

$$= \pi(X')q(X|X') \tag{3.17}$$

$$\Rightarrow \alpha(X, X') = \frac{\pi(X')q(X|X')}{\pi(X)q(X'|X)} \tag{3.18}$$

As in IS, a good choice of proposal distribution $q(X'|X)$ is vital to ensuring convergence of X_i to $\pi(X)$ in a reasonable number of iterations. Intuitively, we would like to choose q such that $\alpha(X, X')$ is relatively high; having $X_i = X_{i-1}$ for many iterations does not provide good sample diversity. On the other hand, $\alpha(X, X')$ being close to 1 all the time may imply that we are taking relatively 'conservative' steps in the domain X and thus will not explore the space sufficiently. If the state space of X is finite and discrete, we may characterize the rate of convergence via how close the second eigenvalue of the transition matrix $P_{i,j} = K(X' = i|X = j)$ is to 1 [Andrzejewski and Chawla, 2007], but in most scenarios such calculations are impossible to perform directly. As with IS, the choice of $q(X'|X)$ is often left to intuition.

In practice, care must be taken when designing $q(X'|X)$ to ensure that high probability regions of the state space are easy to transition within. If such is the case, the MCMC algorithm will be able to traverse the 'most important' parts of the state space with ease; on the other hand, a proposal distribution that requires the chain to traverse low probability regions in order to pass between two high probability ones will intuitively require a large number of iterations to transition. Assessing what set corresponds to a 'high probability' region of the state space is a difficult problem in itself.

In spite of these difficulties, MCMC has proven effective in performing inference on a variety of complex models in Statistics, Bioinformatics, and the Social Sciences [Diaconis, 2009]. Except when working with naturally sequential models, MCMC is often the only reasonable method for performing Bayesian inference, and even then MCMC has shown significant promise [Marthi et al., 2002].

3.3 Particle Filter (PF)

Turning back to IS, we now consider an alternative to MCMC for sampling from high dimensional distributions. Unlike MCMC, where adding a new variable to a model makes reusing previously generated samples unclear (e.g. when extending $P(X_t|Y_{1:t})$ to $P(X_{t+1}|Y_{1:t+1})$), Particle Filters work by maintaining a collection of partial instantiations of the model, augmenting those instantiations with values for new variables, then reweighting those samples, and finally *resampling* from those proposals.

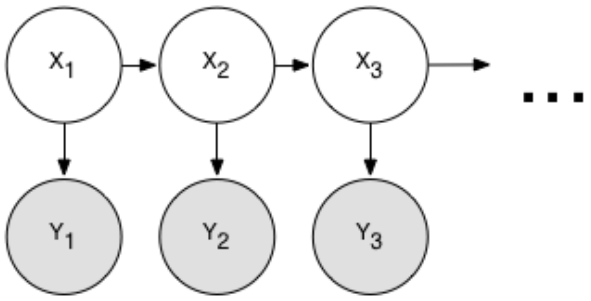


Figure 3.1: A Hidden Markov Model

Algorithm 4 One iteration of a vanilla Particle Filter

Input: particles $X_t^{(p)}$ weighted by $w_t^{(p)}$ approximating $P(X_t|Y_{1:t})$ and Y_{t+1}
Output: particles $X_{t+1}^{(p)}$ weighted by $w_{t+1}^{(p)}$ approximating $P(X_{t+1}|Y_{1:t+1})$

1: Propose $\bar{X}_{t+1}^{(p)} \sim q(X_{t+1}^{(p)}|X_t^{(p)}, Y_{t+1})$ for all particles p
2: Reweight $\bar{X}_{t+1}^{(p)}$ by

$$\bar{w}_{t+1}^{(p)} = \frac{P(Y_{t+1}|\bar{X}_{t+1}^{(p)})P(\bar{X}_{t+1}^{(p)}|X_t^{(p)})}{q(\bar{X}_{t+1}|X_t^{(p)}, Y_{t+1})} \tag{3.19}$$

3: Resample index $Z_p \sim \text{Mult}(\bar{w}_{t+1}^{(p)})$ for all p
4: Set $X_{t+1}^{(p)} = \bar{X}_{t+1}^{(Z_p)}$ and $w_{t+1}^{(p)} = \frac{1}{P}$

The general idea behind **Particle Filter**ing is best understood through a Hidden Markov Model as in Figure 3.1. In this case, Particle Filters approximate $P(X_t|Y_{1:t})$ via a set of 'particles' $X_t^{(p)}$ with weights $w_t^{(p)}$ for $p = 1 \dots P$. By doing so, the filter can then approximate $\mathbb{E}[f(X_t)|Y_{1:t}]$ with $\sum_{p=1}^{P} w_t^{(p)} f(X_t^{(p)})$. The implementation of the PF is nearly identical to IS, except for a technique called 'resampling,' wherein particles that are likely under $Y_{1:t}$ are replicated at the expense of unlikely particles.

Suppose we already have a set of particles for approximating $P(X_t|Y_{1:t})$. We propose new particles for X_{t+1} by first sampling a potential future for $X_t^{(p)}$ at time $t+1$ via $q(X_{t+1}|X_t, Y_{t+1})$ for each particle $X_t^{(p)}$. We then reweight this particle to calculate an IS weight $\bar{w}_{t+1}^{(p)}$. Finally, we use these new weights to create a Multinomial distribution and sample P indices from the propagated candidates. These resampled indices select our new particles, and the algorithm is repeated. Pseudocode is given in Algorithm 4.

Again, the accuracy of the algorithm for fixed P is highly dependant on how closely $q(X_{t+1}|X_t, Y_{t+1})$

matches the ideal posterior $P(X_{t+1}|X_t, Y_{t+1})$ (were we able to do so, all particles would have equal weight). Even more so than Importance Sampling, maintaining low variance weights is key as resampling will replicate particles with high weights. As mentioned in Section 3.1, self-normalization results in a biased estimator of $\mathbb{E}[f(X_{1:t})|Y_{1:t}]$; in spite of this, we can still guarantee convergence to the true posterior as the number of particles approaches infinity [Del Moral, 2004] even with resampling. However, the number of particles necessary to prevent diverging from $P(X_t|Y_{1:t})$ can be anywhere from constant to exponential in t depending on the properties of the underlying model [Doucet and Johansen, 2009].

It is worth noting that Algorithm 4 is the 'vanilla' version of Particle Filtering and many variations exist. For example, particles may be weighted by $P(Y_{t+2}|\bar{X}_{t+1}^{(p)})$ before resampling as in the Auxiliary Particle Filter [Pitt and Shephard, 1999], or weights may depend on all particles at time t as in the Marginal Particle Filter [Klaas et al., 2005]. Resampling need not be restricted to the Multinomial distribution, but can instead be replaced with the lower variance methods such as Stratified Resampling, Systematic Resampling, or Residual Resampling [Douc, 2005]. The number of particles at each time step need not be constant, but can be replaced with a function based on the weights $\bar{w}_t^{(p)}$ [Fox, 2003].

On a more practical note, care must be taken to avoid **Particle Wipeout**, wherein all but a handful of weights are near 0 and the approximation diverges significantly from the true posterior. In this case, either the proposal distribution should be redesigned or the number of particles increased. Secondly, repeated resampling operations result in **Particle Degeneracy**, wherein all particles eventually share the same ancestor near the beginning of time. While this may not be initially alarming, let us present a particular example involving time-invariant parameters to make its importance concrete.

Suppose that in addition to X_t we also seek to infer model parameters $\theta = (\theta_x, \theta_y)$ as in Figure 3.2. Suppose we naively sample $(X_1^{(p)}, \theta) \sim q(X_1, \theta|Y_1)$ at $t = 1$, then 'carry' θ with $X_t^{(p)}$ unaltered as t increases. Due to particle degeneracy, θ will quickly converge to a single value as resampling can only reduce the number of unique candidate values for θ. An example of this is presented in Figure 3.3. We will address these concerns with the Resample-Move Particle Filter (RMPF) in Section 3.5.

In spite of these dangers, Particle Filters have proven particularly effective in models with a

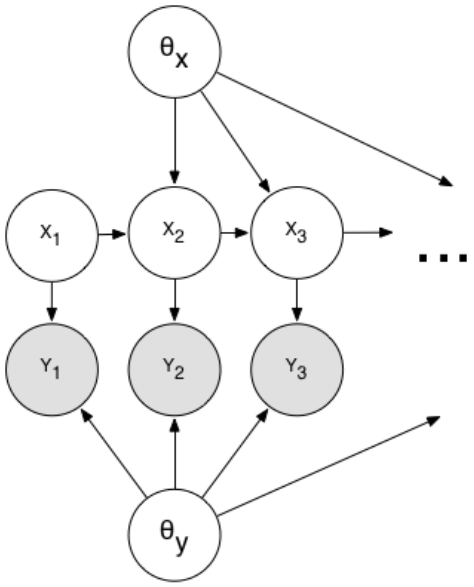


Figure 3.2: A Hidden Markov Model with time-invariant parameters

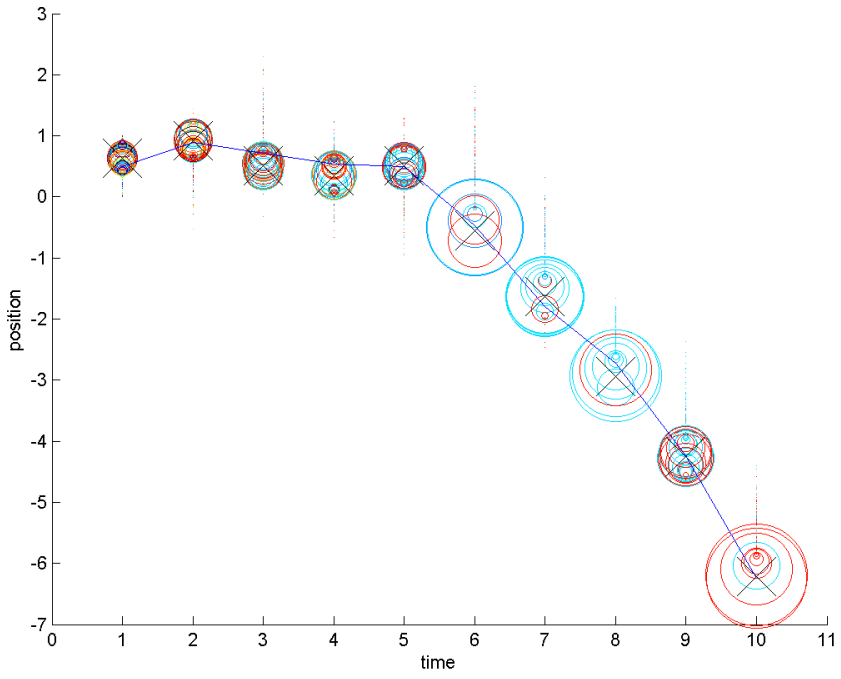


Figure 3.3: A naive Particle Filter tracking a single target. Colors denote the value of a single model parameter sampled at $t = 1$ which has no effect on state transitions or observations. Circle sizes denote particle weights and circle centers denote states in one dimension. Crosses denote observations, and lines denote the true state sequence.

natural sequential construction such as visual target tracking [Nummiaro et al., 2003] and robot localization [Montemerlo et al., 2002]. In Section 4.1, we shall see that Particle Filters remain one of the most popular techniques in Multiple Target Tracking (MTT) as well.

3.4 Reversible Jump MCMC (RJMCMC)

In Section 3.2, we introduced MCMC, a method for performing approximate inference when X is of a large but *fixed* dimension. While this is sufficient for many models, it cannot handle cases where the dimensionality of X is unknown. This is precisely the case in Multiple Target Tracking (MTT) as the number of active targets varies with time.

To understand how **Reversible Jump MCMC** [Green, 1995] works, let us consider the simple Gaussian Mixture Model. In this model, there are K Gaussian distributions, each with its own mean $\mu_{(k)}$ and variance $\sigma^2_{(k)}$. To draw a sample from this model, one first samples which Gaussian to use (Z) then samples a point (X) using that Gaussian's parameters. If we let component k be chosen with probability $w_{(k)}$, then the density of any sample X, Z is

$$P(X, Z) = P(Z|w_{(1:K)})P(X|Z, \mu_{(Z)}, \sigma^2_{(Z)}) \tag{3.20}$$

One may then ask, "What if K is unknown?" We may place a prior over K, mixture weights $w_{(k)}$, and mixture parameters $\mu_{(k)}$ and $\sigma^2_{(k)}$ to give a new density,

$$P(X, Z, w_{(1:K)}, \mu_{(1:K)}, \sigma^2_{(1:K)}) = P(K)P(w_{(1:K)}|K)\left(\prod_{k=1}^{K} P(\mu_{(k)}P(\sigma^2_{(k)}))\right) \times \tag{3.21}$$
$$P(Z|w_{1:K})P(X|Z, \mu_{(Z)}, \sigma^2_{(Z)})$$

Given this new model, how then may we apply MCMC? Reversible Jump MCMC allows us to apply the familiar change-of-variable formula used in calculating integrals to derive a new acceptance ratio $\alpha(\cdot)$. The overall idea is to first sample from a collection of MCMC proposal types $m \sim q(m|X)$, then generate random numbers $U \sim q(U|X, m)$, and finally to construct $(X', U') = h(X, U)$ via an invertible function h where $dim(X', U') = dim(X, U)$. The mapping $h(X, U) \to (X', U')$ is a change of variables, and thus must be accounted for via the determinant of its Jacobian. A single iteration of RJMCMC is presented in Algorithm 5.

Algorithm 5 One iteration of the Reversible Jump MCMC

Input: input state X_i

Output: output state X_{i+1} such that $\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_i f(X_i)$ as $N \to \infty$

1: Sample move-type $m \sim q(m|X_i)$

2: Sample random numbers $U \sim q(U|X_i, m)$

3: Calculate $(X', U') = h(X_i, U)$

4: Calculate acceptance ratio

$$\alpha(X_i, U, X', U') = \min\left(\frac{\bar{\pi}(X')q(m|X')q(U'|m, X')}{\bar{\pi}(X)q(m|X)q(U|m, X)}\left|\frac{\partial}{\partial(X_i, U)}h(X_i, U)\right|, 1\right) \tag{3.22}$$

5: Sample $u \sim \text{Unif}(0, 1)$. If $u < \alpha(X_i, U, X', U')$ let $X_{i+1} = X'$, else $X_{i+1} = X_i$

Let us now consider what form $h(X, U)$ and $\alpha(\cdot)$ take in the Gaussian Mixture Model. Our primary concern is MCMC moves that alter the number of mixture components as all other moves may be handled with standard MCMC techniques. Suppose we have a pair of MCMC moves designed as follows,

1. Birth: propose a new cluster by sampling an unnormalized mixture weight $u_1 \sim q(w)$ ($0 \leq u \leq 1$), mean $u_2 \sim q(\mu)$, and variance $u_3 \sim q(\sigma^2)$. Normalize all weights by $w_k \leftarrow w_k(1-u_1)$, then set $w_{K+1} = u_1$, $\mu_{K+1} = u_2$, and $\sigma^2_{K+1} = u_3$

2. Death: delete cluster $K+1$ and set $w_k \leftarrow w_k/(1-w_{K+1})$ for all $k = 1 \dots K$.

Notice that both moves result in changing $\{w_k\}$ in order to ensure summation to one; this is the key reason h is necessary. We will assume that each move-type m (including standard MCMC moves not described here) is selected with probability $q(m|\cdot)$ where $\cdot$ denotes the state of the RJMCMC chain. Suppose a birth move is selected and we need to calculate the acceptance ratio $\alpha_{K\to K+1}(\cdot)$. In order to ease notation, let $\theta_K = (w_{1:K}, \mu_{1:K}, \sigma^2_{1:K})$. If we define,

$$h_{K\to K+1}(w_{1:K}, \mu_{1:K}, \sigma^2_{1:K}, u_{1:3}) = \left[w_{1:K}(1-u_1), u_1, \mu_{1:K}, u_2, \sigma^2_{1:K}, u_3\right] \tag{3.23}$$

Then we may calculate the determinant of $h_{K\to K+1}$'s Jacobian as $(1-u_1)^K$ and the acceptance ratio $\alpha_{K\to K+1}(\cdot)$,

$$\alpha_{K\to K+1}(\cdot) = \min\left(\frac{P(X, Z, \theta_{K+1})q(m = \text{death}|\theta_{K+1})}{P(X, Z, \theta_K)q(m = \text{birth}|\theta_K)q(u_{1:3}|\text{birth})}(1-u_1)^K, 1\right) \tag{3.24}$$

If a death move is selected instead, $h_{K+1\to K}$ is defined to be the inverse of $h_{K\to K+1}$. In consequence, $\alpha_{K+1\to K}(\cdot)$ uses the inverse of the contents of the min. Note that if h does not alter any preexisting variables, the determinant of its Jacobian is 1.

3.5 Resample Move (RM)

In Sections 3.2 and 3.3, we discussed MCMC and Particle Filters as two methods for performing Monte Carlo inference for probability distributions in high dimensional spaces. One may now consider if these methods can be combined, and if so, how? While Particle Filters are well suited for time-varying distributions with frequent observations, their design prohibits them from 'changing the past' – that is, one cannot change X_t once it has been sampled. MCMC lacks this flaw, but it is not so simple to extend previously generated samples when considering a new time step (see Marthi et al. [2002] for an MCMC alternative to Filtering). **Resample-Move** [Gilks and Berzuini, 2001] is one such approach to marry the two.

Resample-Move is essentially a method of embedding MCMC within a Particle Filter. In words, Resample-Move runs the same PF algorithm presented in Section 3.3, but with the addition of a final step after resampling. Here, each particle is passed through an MCMC kernel with invariant distribution $P(X_{1:t+1}|Y_{1:t+1})$ one or more times without waiting until a stationary distribution is reached. The intuition is that since $\{X^{(p)}_{1:t+1}\}_{p=1}^{P}$ are already distributed approximately like $P(X_{1:t+1}|Y_{1:t+1})$, there is no need to wait. Pseudocode is presented in Algorithm 6.

While at first the value of this method may seem suspect, its utility is evident when considering the time-invariant parameter scenario presented in Section 3.3. Recall the modified HMM model presented again in Figure 3.4. While the original Particle Filter is unable to alter θ after sampling it at $t = 1$, Resample-Move allows gives one the opportunity to resample θ at any time by applying MCMC moves with stationary distribution $P(\theta, X_{1:t}|Y_{1:})$. In the example presented, θ actually has no effect on X_t or Y_t, so resampling from the prior over θ is equivalent to Gibbs Sampling. Finally, it can be shown that as the number of particles P approaches infinity, Resample-Move provides the same guarantees as the original Particle Filter for any number of MCMC iterations.

Finally, it is worth considering how to efficiently perform an MCMC step when proposing changes to static parameters θ. Were the Metropolis-Hastings algorithm applied, calculating the

Algorithm 6 One iteration of a Resample-Move Particle Filter

Input: particles $X_{1:t}^{(i)}$ weighted by $w_t^{(i)}$ approximating $P(X_{1:t}|Y_{1:t})$ and Y_{t+1}
Output: particles $X_{1:t+1}^{(i)}$ weighted by $w_{t+1}^{(i)}$ approximating $P(X_{1:t+1}|Y_{1:t+1})$

1: Execute Algorithm 4 to obtain $X_{1:t+1}^{(p)}, w_{t+1}^{(p)}$.
2: For each particle p, sample $X_{1:t+1}^{(p)} = X_{1:t+1}^{(p)\prime}$ where $X_{1:t+1}^{(p)\prime} \sim K(X_{1:t+1}^{(p)\prime}|X_{1:t+1}^{(p)})$ where K's invariant distribution is $P(X_{1:t+1}|Y_{1:t+1})$

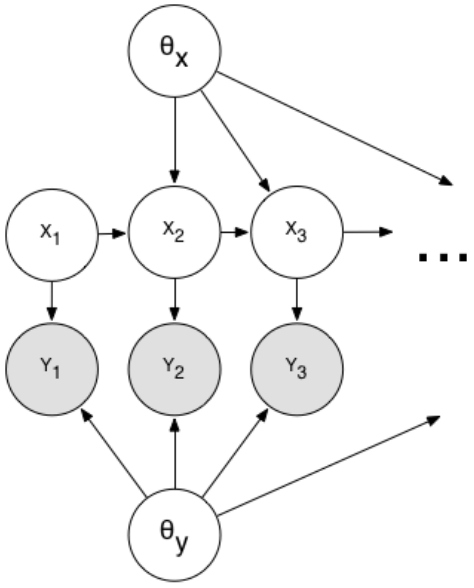


Figure 3.4: A Hidden Markov Model with unknown static parameters θ_X and θ_Y

acceptance ratio $\alpha(\cdot)$ would include a number of terms linear in t. An alternative method forgoing acceptance ratios is Gibbs Sampling which requires that $P(\theta|X_{1:t}, Y_{1:t})$ can be drawn from directly. This is the case when $P(\theta|X_{1:t}, Y_{1:t})$ falls into a simple enough distribution that can be summarized via **Sufficient Statistics**. For example, if θ represents the probability a robot moves at time t, then a sufficient statistic is the number of $X_{1:t}$ wherein the robot moved. Further details can be found in Fearnhead [2002].

In summary, the Resample-Move is a method for obtaining the benefits of Particle Filters without prohibiting one from 'changing the past.' However, it should be noted that if the particles $\{X_{1:t}^{(p)}\}_p$ diverge significantly from $P(X_{1:t}|\theta)$, one cannot hope to successfully recover samples from $P(\theta|Y_{1:t})$ without a large number of MCMC iterations.

3.6 Particle MCMC (PMCMC)

While Resample-Move embeds MCMC within a Particle Filter, **Particle Markov Chain Monte Carlo** [Andrieu et al., 2010] achieves the exact opposite by embedding Particle Filters within MCMC. Instead of applying MCMC moves to each particle in isolation, Particle MCMC creates a single MCMC chain with Particle Filters run at each iteration.

Suppose once again that we would like to generate samples from $P(\theta, X_{1:T}|Y_{1:T})$, this time in a batch fashion with T fixed. Assume that we are able to calculate $P(\theta)$ and $P(Y_{1:T}|\theta) = \int P(Y_{1:T}|X_{1:T})P(X_{1:T}|\theta)dX_{1:T}$ exactly. If this is the case, we may apply Metropolis-Hastings directly using acceptance ratio,

$$\alpha(\theta, \bar{\theta}) = \frac{P(Y_{1:T}|\bar{\theta})P(\bar{\theta})q(\theta|\bar{\theta})}{P(Y_{1:T}|\theta)P(\theta)q(\bar{\theta}|\theta)} \tag{3.25}$$

This will allow us to draw samples approximately from $P(\theta|Y_{1:T})$ while ignoring $X_{1:T}$. Now suppose that we are not able to calculate $P(Y_{1:T}|\theta)$ exactly, but instead use an approximation $\hat{P}(Y_{1:T}|\theta)$ calculated via the average particle weights output by the particle filter,

$$\hat{P}(Y_{1:T}|\theta) = \prod_{t=1}^{T} \hat{P}(Y_t|Y_{1:t-1}) = \prod_{t=1}^{T} \frac{1}{P} \sum_{p=1}^{P} \bar{w}_t^{(p)} \tag{3.26}$$

We already know that a Particle Filter provides a biased estimate for $\mathbb{E}[f(X_{1:T})|Y_{1:T}]$, so why should we even consider replacing $P(Y_{1:T}|\theta)$ with its approximation? It so happens that in spite of resampling, a Particle Filter always produces an unbiased estimate of $P(Y_{1:T}|\theta)$ for any number of particles [Del Moral, 2004, Pitt et al., 2010], and furthermore, replacing $P(Y_{1:T}|\theta)$ with an unbiased approximation will result in the same stationary distribution as if we used its true value. Finally, we may select a single particle from $\{X_{1:T}^{(p)}\}_{p=1}^{P}$ (the output of the particle filter) according to $w_T^{(p)}$ to pair with candidate $\bar{\theta}$ to create a new MCMC chain with posterior distribution $P(X_{1:T}, \theta|Y_{1:T})$. The precise algorithm is presented in Algorithm 7.

In summary, Particle MCMC allows one to do 'approximate' Rao-Blackwellization by replacing exact integration with a Particle Filter. Unlike Resample-Move, Particle MCMC does not directly suffer from $\{X_{1:t}^{(p)}\}_p$ diverging from $P(X_{1:t}|Y_{1:t}, \theta)$; however, inaccurate approximations to $P(Y_{1:T}|\theta)$ may slow convergence to the stationary distribution.

Finally, Particle MCMC is presented in Andrieu et al. [2010] and here using the vanilla Particle Filter presented in Section 3.3, but convergence can be guaranteed beyond this simple case. We contribute the following two theorems with proofs presented in the Appendix.

Theorem 1. *Suppose a Resample-Move Particle Filter is executed as described in Algorithm 6. Let* $\hat{P}(Y_{1:T}|\theta)$ *be defined as in Equation 3.26. Then* $\mathbb{E}[\hat{P}(Y_{1:T}|\theta)] = P(Y_{1:T}|\theta)$.

Theorem 2. *Suppose a Resample-Move Particle Filter were to replace the vanilla Particle Filter in Algorithm 7. Then the resulting algorithm is an MCMC chain with stationary distribution* $P(X_{1:T}, \theta|Y_{1:t})$.

Algorithm 7 One iteration of a Particle MCMC

Input: state $(\theta, X_{1:T})_i$ and likelihood approximation $\hat{Z} = \hat{P}(Y_{1:T}|\theta)$
Output: state $(\theta, X_{1:T})_{i+1}$ such that $\mathbb{E}[f(X_{1:T}, \theta)|Y_{1:T}] \approx \frac{1}{N}\sum_i f((X_{1:T}, \theta)_i)$ as $N \to \infty$

1: Propose $\bar{\theta} \sim q(\bar{\theta}|\theta_i)$
2: Run algorithm 4 to generate weights $\bar{w}_t^{(p)}$ for all $t = 1 \dots T$ and $p = 1 \dots P$. Use these to calculate $\bar{Z} = \hat{P}(Y_{1:T}|\bar{\theta})$.
3: Choose one particle $X_{1:T}^{(p)}$ according to weights $w_T^{(p)}$ from the output of the previous step. Let this be $\bar{X}_{1:T}$.
4: Calculate acceptance ratio,

$$\alpha = \min\left(\frac{\bar{Z}P(\bar{\theta})q(\theta_i|\bar{\theta})}{\hat{Z}P(\theta_i)q(\bar{\theta}|\theta_i)}, 1\right) \tag{3.27}$$

5: Sample $u \sim \text{Unif}(0, 1)$. If $\alpha < u$, set $(X_{1:T}, \theta)_{i+1} = (\bar{X}_{1:T}, \bar{\theta})$, else $(X_{1:T}, \theta)_i$

3.7 Summary

In this chapter, we have presented Importance Sampling, Markov Chain Monte Carlo, and Particle Filters, three standard algorithms for doing approximate inference in Bayesian Models. In addition, we have described Reversible Jump MCMC, the Resample-Move Particle Filter, and Particle MCMC, the latter two of which form the basis of Particle Filter Data Association (PFDA) and Particle MCMC Data Association (PMCMCDA). We have described their primary usage cases and drawbacks and given broad statements about their convergence properties. In summary, we present Table 3.1 as an overview of the methods, their usage cases, benefits, drawbacks, and computational cost.

Method Name	Builds On	Usage Case	Online/Batch	Requires	Dangers	Complexity
Importance Sampling		low-dimension distributions	Online	$q(X)$	High variance weights	$O(N_i T)$
Particle Filters	Importance Sampling	time-indexed distributions	Online	$q(X_t \| X_{t-1}, Y_t)$	High variance weights, inability to alter past	$O(TP)$
Markov Chain Monte Carlo	Markov Chain Theory	high-dimension distributions	Batch/Online	$q(X' \| X)$	unknown running time, difficulty in checking convergence	$O(N_i)$
Reversible Jump MCMC	MCMC	Distributions with unknown dimension	Batch	$q_i(U \| X)$, $h(X, U)$	unknown running time, difficulty in checking convergence	$O(N_i)$
Resample-Move	PF, MCMC	time-indexed distr. with static parameters	Online	$q(X_t \| X_{t-1}, Y_t)$, $q(X'_{1:t} \| X_{1:t})$	Particle Degeneracy, additional computational complexity	$O(TPN_i)$
Particle MCMC	PF, MCMC	time-indexed distr. with static parameters	Batch	$q(X_t \| X_{t-1}, Y_t)$, $q(\theta' \| \theta)$	Large running time, difficulty in checking convergence	$O(TPN_i)$

Table 3.1: A comparison of Monte Carlo algorithms presented in this chapter. In the (per iteration) 'Complexity' column, T = number of time steps, P = number of particles, N_i = number of MCMC or IS iterations. Note that the 'Complexity' column has no correspondence to the necessary number of iterations or particles required for adequate convergence.

Chapter 4

Multiple Target Tracking

Multiple Target Tracking is the problem of state estimation for an unknown, time-varying number of targets. For example, we may have a naval radar system and would like to identify the position and velocity of all ships in range or a surveillance camera with the intent of identifying the times people enter and leave a building.

While this problem is mundane and trivial for most humans, it has proven to be computationally a very difficult. Consider the surveillance camera example. Given a single frame, traditional Computer Vision methods for object recognition rely on a "bottom-up" approach where low-level characteristics of the input (such as the color distribution of a person) are used to train a discriminative algorithm for identifying 'people-like' pixel patches in the video. However, these algorithms ignore the true structure of what is being observed (e.g., people in a scene), and thus are prone to making mistakes due to occlusion, changes in lighting, and camera movement.

On the other hand, "top-down" generative models of vision suffer from the large number of variables and high degree of uncertainty. While larger, more complex models may better correspond to reality, the burden of inference may be too great to accurately approximate the posterior in a reasonable amount of time. Thus a balance must be struck between model complexity and model accuracy, resulting in a small number of models for which inference can be done efficiently.

Current solutions lie somewhere between "top-down" and "bottom-up" approaches. While discriminative approaches have proven to be very robust at object recognition, reasoning about abstract objects remains difficult. On the other hand, generative models are capable of inferring behaviour among abstract objects but have difficulty dealing with sensors directly. Thus, the two

are often combined in a **Track-By-Detection** framework where the former identifies 'detections' that are likely to correspond to abstract objects and the former reasons about them. This is the methodology taken in this work and many others as shall be described within this Chapter.

In order to motivate Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA), we survey the state of the art in Multiple Target Tracking algorithms in Section 4.1. We describe these algorithms based on how they handle data association and target state estimation. In Section 4.2, we describe the CLEAR metric, a measure of Multiple Target Tracking performance. This is the metric used in our own experiments in the following Chapter.

4.1 Algorithms

In this section we will describe the current state of the art in Multiple Target Tracking algorithms based on the track-by-detection framework. We will discuss their design decisions and the consequences that has on their behaviour. We will categorize them according to their approach to data association and state estimation. A more complete treatment of discriminative detection-generation models, state representations, and bottom up methods is left to Yilmaz et al. [2006].

4.1.1 Data Association

Typically, data association algorithms assume that one target can correspond to one or zero detections (in our notation, a_t is one-to-one). This has not always been the case, however; in Fortmann et al. [1983] the Joint Probabilistic Data Association (JPDA) assumed that every target could contribute to every detection. Thus, the probability of all detections at time t, denoted $\{y_t^{(i)}\}_{i=1}^{N_t^y}$, given the state of a track k at time t, denoted $x_t^{(k)}$, is given by $P(\{y_t^{(i)}\}_{i=1}^{N_t^y}|x_t^{(k)}) = \sum_{i=1}^{N_t^y} P(y_t^{(i)}|x_t^{(k)})$. This approximation allows one to track each of an assumed known, fixed number of active targets independently. While this is a reasonable approximation for highly dispersed targets, it results in targets being 'drawn together' when the two tracks pass nearby.

Other algorithms which sidestep the data association problem include Karlsson and Gustafsson [2001], Kreucher et al. [2004], Mahler [2003], Vo et al. [2005]. Karlsson and Gustafsson [2001] integrates JPDA directly into a particle filtering framework, resulting in the same weaknesses as the original Kalman-Filter based JPDA algorithm but the ability to do inference in general

dynamics models. Kreucher et al. [2004] on the other hand eschews detections altogether and treats observations as a pixel-level signal. Finally, Mahler [2003] introduces the Probability Hypothesis Density (PHD) Filter derived from Finite Set Statistics, a new theoretical formulation separate (but not unlike) typical measure theory-based probability theory. Vo et al. [2005] then transforms this formulation into a Particle Filter capable of estimating the expected number of targets in a given area but unable to identify unique targets.

Soon after Fortmann et al. [1983], Blackman [1986] introduced Multiple Hypothesis Tracking (MHT), an algorithm which enumerates all data associations $a_{1:t}$ using the one-to-one assumption mentioned previously. While continually maintaining all possible hypotheses would be computationally infeasible (indeed, the number such associations is exponential in the number of active targets), pruning is done by calculating the likelihood of each maintained hypothesis and removing unlikely candidates. A distance gating condition is used to prevent distant detections from being attributed to the same target. It is worth noting that MHT is a popular method in military applications even today due to its tunable computational cost [Blackman, 2004].

Many recent approaches to tracking have maintained data associations within the particles in a Particle Filter. Särkkä et al. [2007], for instance, applies a Rao-Blackwellized Particle Filter to maintain a set of likely data associations then uses a Kalman Filter to track individual targets. On the other hand, many others including Breitenstein et al. [2009], Hess and Fern [2009], Yang et al. [2005] ignore all but one data association and rely on heuristic or discriminative methods for deciding which targets are responsible for which detections.

A final approach is to use MCMC to enumerate the space of data associations. While only Marthi et al. [2002], Pasula et al. [1999], Oh et al. [2004], Vu et al. [2011] chose to use MCMC as their primary tool for inference, Benfold and Reid [2011], Khan et al. [2004] use MCMC in an online tracking framework to 'correct' proposed hypotheses generated by other algorithms in an online fashion. For example, Benfold and Reid [2011] uses MCMC as a post-processing step in a discriminative tracking algorithm to suggest new associations while Khan et al. [2004] uses it as a method for combining multiple 1-target-per-filter particle filters. Oh et al. [2004] on the other hand uses MCMC to propose data associations in an offline algorithm while Vu et al. [2011] combines the PHD Filter and Particle MCMC. Pasula et al. [1999], Marthi et al. [2002] are unique in that they employ MCMC to perform Filtering directly.

Finally, it should be noted that not all algorithms mentioned above are addressing the same problem. Some, such as Karlsson and Gustafsson [2001], Kreucher et al. [2004], consider a fixed, known number of targets while others also allow the number of tracks to vary from time step to time step.

4.1.2 State Estimation

State estimation in Multiple Target Tracking is approached from one of two perspectives: the Gaussian noise case and the general model case. In the former, one is able to apply to Kalman Filter, Extended Kalman Filter, or Unscented Kalman Filter [Wan and Van Der Merwe, 2000] to estimate the state in a (pseudo) exact way. In the latter, the method of choice is often the Particle Filter, though its exact implementation varies.

If each target were known to move independently and its active times were known, each can safely be assigned its own particle filter which operates without knowledge of its neighbors. Even if these times are not known, discriminative or heuristic methods are often applied to decide when one should start and stop a new track; this is the approach taken in Breitenstein et al. [2009], Hess and Fern [2009], Yang et al. [2005]. If multiple targets are allowed to contribute to each detection (in other words, if a_t is not one-to-one), this results in 'particle hijacking', wherein multiple independent particle filters begin tracking the same target, a Particle Filter manifestation of JPDA's flaws. Methods to combat this while maintaining one particle filter per target include heuristics [Breitenstein et al., 2009, Hess and Fern, 2009] and on-the-fly creation of Markov Random Fields when targets are nearby [Khan et al., 2004].

The alternative to the one-particle-filter-per-track method is naturally to represent the joint state of all tracks as a single particle. This is the approach taken in Khan et al. [2004], albeit with a twist; instead of using the typical propagate-resample framework of the usual particle filter, Khan runs an MCMC chain over the product space of all targets and takes the final samples generated by it as his unweighted 'particles'. The benefit of aggregating all track states into single particles is the ability to seamlessly represent interacting targets in a unified framework; on the other hand, the drawback is that the number of particles necessary for maintaining an accurate estimate of the posterior may be significantly increased. In fact, if each track moves independently, the variance of the particle weights grows exponentially in the number of active tracks.

To see this, let us suppose we would like to approximate two distributions $\pi(X)$ and $\pi(Y)$ where X is independent of Y. A particle filter would generate samples for $X^{(p)} \sim q(X)$ and $Y^{(p)} \sim q(Y)$, then calculate a weight

$$w^{(p)} = w_x^{(p)} w_y^{(p)} = \frac{\pi(X^{(p)})}{q(X^{(p)})} \frac{\pi(Y^{(p)})}{q(Y^{(p)})} \tag{4.1}$$

We may calculate $\mathrm{Var}(w^{(p)})$ directly as

$$\mathrm{Var}(w^{(p)}) = \mathbb{E}[w_y^{(p)}]^2 Var(w_x^{(p)}) + \mathbb{E}[w_x^{(p)}]^2 Var(w_y^{(p)}) + \mathrm{Var}(w_x^{(p)})\mathrm{Var}(w_y^{(p)}) \tag{4.2}$$

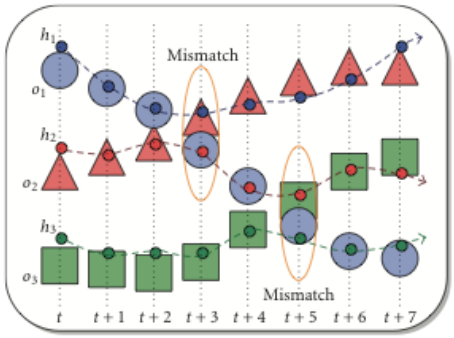
This final term multiplying the two variances makes $\mathrm{Var}(w^{(p)})$ at least exponential in the number of active, independently moving targets.

The astute reader will note that not once was estimation of model parameters included in any of the aforementioned algorithms. Indeed, these variables are assumed known and fixed in all of the above methods. In addition, no method allows one to consider both non-Gaussian noise models and multiple data associations simultaneously. It is with these goals in mind that Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA) are formulated.

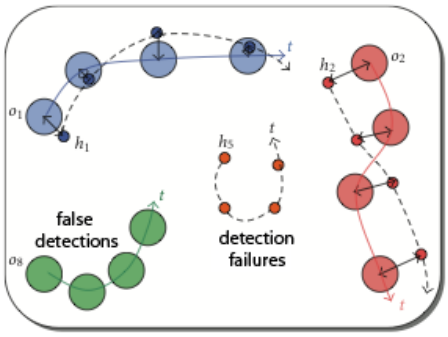
4.2 CLEAR Metric

In spite of the long history of Multiple Target Tracking, there do not exist any standard metrics by which to measure performance. This stems from the variety of behaviours an algorithm may exhibit and how much the user may object to it. For example, an algorithm may incorrectly swap the latter halves of two tracks, as in Figure 4.1a; did this algorithm only label half of its detections correctly? For vision applications, how should one measure how well a track's state is estimated given bounding boxes and mean positions? How about false detections and detection failures, or when a track is cut into multiple pieces?

One such metric which has attempted to aggregate all of these considerations is the CLEAR Multiple Object Tracking metric [Bernardin and Stiefelhagen, 2008]. In words, this metric identifies at each time step a mapping between the targets proposed by the tracker and by ground truth. Given this, it calculates the position error and the number of false detections, detection failures,



(a) An error wherein an algorithm mistakenly trades the places of two targets



(b) An error wherein an algorithm mistakenly identifies clutter as an actual state sequence (green) and fails identify a sequence of detections as an actual target (orange)

Figure 4.1: Common errors in Multiple Target Tracking. Small balls of the same color represent detections generated by an actual target, while centers of large shapes of the same color represent hypothesized position sequences of the same object. Taken from Bernardin and Stiefelhagen [2008].

and 'mismatches' (see Figure 4.1). These numbers are then appropriately averaged to present an idea of the tracker's performance.

Formally, the CLEAR metric initializes with an empty mapping between true targets and hypotheses at time $t = 0$. At each time step, Algorithm 8 is executed with parameter $d_{\max}$, the maximum distance between a true and hypothesized target. Finally, the mean false detection rate, detection failure rate, mismatch rate, and position error rate are averaged as follows,

$$\overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \tag{4.3}$$

$$\overline{fn} = \frac{\sum_t fn_t}{\sum_t g_t} \tag{4.4}$$

$$\overline{mm} = \frac{\sum_t mm_t}{\sum_t g_t} \tag{4.5}$$

$$\text{MOTA} = 1 - \overline{fn} - \overline{fp} - \overline{mm} \tag{4.6}$$

$$\text{MOTP} = \frac{\sum_t d_t}{\sum_t c_t} \tag{4.7}$$

Readers should note that the Multiple Object Tracking Accuracy (MOTA) is based entirely on the quality of a data association and the Multiple Object Tracking Position (MOTP) is solely dependent on the state estimates. An ideal algorithm will score 1.0 for MOTA and 0.0 for MOTP. For reference, typical results for person tracking have a $\overline{fn}$ around 12% and $\overline{fp}$ between 5 and

30%. We refer the reader to Bernardin and Stiefelhagen [2008] for example evaluations of several algorithms.

Algorithm 8 The CLEAR MOT metric for time t

Input: a mapping between true targets and target hypotheses $\{o_k, h_{k'}\}_{t-1}$ from time $t-1$

Output: Counts for false detections fp_t, detection failures fn_t, mismatches mm_t, true targets g_t, matches c_t, total distance error d_t, and a new mapping $\{o_k, h_{k'}\}_t$

1: Keep each correspondence $o_k, h_{k'}$ from time $t-1$ where true target o_k and hypothesis $h_{k'}$ are still active at time t and are within a distance $d_{\max}$ of each other.

2: Establish possible correspondences between all unmatched o_k active at time t with unmatched hypotheses $h_{k'}$ if their position distance is less than $d_{\max}$. Make a mapping between them by minimizing their sum position distance. This establishes $\{o_k, h_{k'}\}_t$.

3: Let

- fp_t the number of $h_{k'}$ assigned to a target at time t when $h_{k'}$ actually corresponds to clutter
- fn_t the number of $h_{k'}$ assigned to clutter when actually t when $h_{k'}$ was actually generated by a target
- mm_t the number of o_k matched at time t with a different $h_{k'}$ than they were at $t-1$
- g_t the number of o_k
- c_t be the size of $\{o_k, h_{k'}\}_t$
- d_t the sum of position distances between all matches at time t

Chapter 5

Contributions

With the necessary background now sufficiently covered, we embark on describing two new algorithms for Multiple Target Tracking: Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA). We begin by making concrete the form of the joint distribution over which inference is being performed. The following two sections motivate the algorithms, describe their implementation, and prove their correctness. We continue with experiments describing the strengths and weaknesses of both algorithms. Finally, we conclude with a summary of our results.

5.1 Model

We begin by reviewing the notation used throughout this chapter based on that presented in Figure 2.4 in Section 2.2. By convention, we will denote $x_t^{(k)}$ denote the hidden state of target k at time t, alive_t the set of all targets k in view, $\#\text{new}_t$ the number of new targets, and $\#\text{clut}_t$ the number of clutter detections. As for detections, $y_t^{(i)}$ denotes the ith detection generated at time t and $Y_t = \{y_t^{(i)}\}_i$ the collection of all detections. We say $a_t(i) = k$ or $a_t^{-1}(k) = i$ if target k generated detection i at time t. We reserve $a_t(i) = 0$ to denote that an observation was generated by clutter and $a_t^{-1}(k) = 0$ to denote that target k failed to generate a detection. We denote the beginning and end of target k with t_k^{start} and t_k^{end}. The set of all static parameters of the model will be denoted by θ. When the notation $1 : t$ is used, it is to denote the set of variables from times 1 to t; eg, $X_{1:t} = \{X_1 \dots X_t\}$. We will always index time by t, targets by k, observations by i and

particles by p. T will denote the end of time.

The goal of the algorithm is to simultaneously estimate the distribution over all target states X_t, data associations a_t, and static parameters θ given detections $Y_{1:t}$; that is, $P(X_{1:t}, a_{1:t}, \theta | Y_{1:t})$.

5.1.1 Birth/Death and Data Association Model

The birth/death and data association model mirrors that used in Oh et al. [2004]. That is for each time step t,

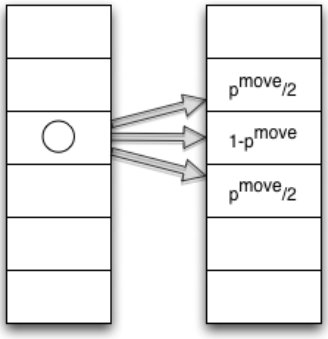
- $\#\text{new}_t$, the number of new targets, is distributed according to Poisson(λ^{new})
- $\#\text{clut}_t$, the number of clutter detections, is distributed according to Poisson(λ^{false})
- Each active target generates an observation at time t with probability p^{obs}; equivalently, $P(a_t^{-1}(k) = 0) = 1 - p^{\text{obs}}$.
- Each active target will end with probability p^{end} at each time step; that is, a target's length is distributed according to Geometric(p^{end})

Thus, the static parameters so far are λ^{new}, λ^{false}, p^{obs}, and p^{end}. We will assume that each of these is distributed according to its corresponding conjugate prior.

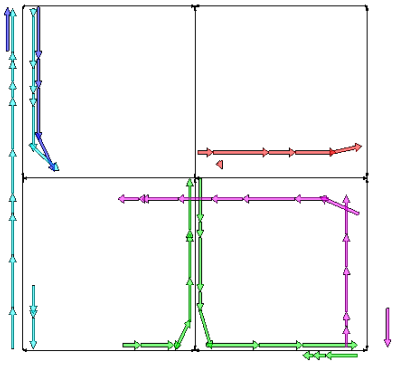
We will further assume that all tracks move independently. Thus, all that is left is to define the per-target dynamics model $P(x_t^{(k)} | x_{t-1}^{(k)})$, per-target initial state distribution $P(x_{t_k^{\text{start}}}^{(k)})$, observation distribution $P(y_t^{(i)} | x_t^{(k)}, a_t(i) = k)$, and false observation distribution $p(y_t^{(i)} | a_t(i) = 0)$. Given these, the likelihood of any given hypothesis is,

$$\begin{aligned} P(X_{1:T}, Y_{1:T}, a_{1:T}, \theta) = & P(\theta) \left(\prod_t P(\#\text{clut}_t) P(\#\text{new}_t) \right) \times \\ & \left(\prod_t \prod_i P(y_t^{(i)} | a_t(i), x_t^{(a_t(i))}) \right) \times \\ & \left(\prod_k P(t_k^{end} - t_k^{start}) \times \right. \\ & \left. \prod_{t=t_k^{start}}^{t_k^{end}} P(x_t^{(k)} | x_{t-1}^{(k)}) P(a_t^{-1}(k) = 0) \right) \end{aligned} \tag{5.1}$$

We assume that all target tracks have ended by time T in this definition, though it is straightforward to calculate otherwise. In words, the first line covers the static parameters, the number of



(a) A state transition in the Random Walk model



(b) States sequences for five cars in the traffic model

targets initialized, and number of false detections. The next accounts for the detections themselves given the associated target's state, if any. The final line accounts for the sequence of states for each target k, whether or not it was detected, and its length.

5.1.2 Dynamics and Observation Models

We define two models: a finite, discrete, random-walk model and a traffic model.

Random Walk

Consider a simple model where a vacuum robot is in one of N states, arranged in a line. With probability p^{move}, the robot moves to one of the two adjacent states, each selected with equal probability. If the robot is observed, with probability p^{true}, the robot reports its true position, else it reports one of the N states uniformly at random. When a new robot enters, it will start in one of the N states uniformly at random. If a false detection is made, we will assume that it is generated uniformly at random from the states.

In this model, the unknown static parameters are p^{move} and p^{true}, which we will again assume are distributed according to their corresponding conjugate prior.

Traffic

Consider now a single car travelling on a road network represented by a graph where edges are roads and nodes are intersections. The state of the car is its velocity v, the current road it's traveling on e, and the distance along the road the car has traveled d. At each time step, the car selects its new velocity v' from a Truncated Normal distribution with mean v, variance σ_v^2, minimum value 0, and maximum value $v_{\max}$. Given this new velocity, the car then continues onwards for 1 time step; i.e., until it has traveled a distance equal to v'. If this distance is less than the remaining length of the road, the car continues up to that point. If it encounters an intersection before then, however, a new road is selected from the roads connected to that node from a Multinomial distribution specific to that intersection; i.e., $P(e'|i) = p^{e',i}$ with $\sum_{e'} p^{e',v} = 1$. The vehicle then continues on this new road.

If a vehicle is detected, we assume that it generates an observation from a two dimensional Normal distribution centered at its current location with variance $\sigma_o^2 I$. On the other hand, a false detection is assumed to be generated from a two dimensional uniform distribution with length and width twice the distance between the farthest two intersections of the road network.

Here, the model-specific parameters are $p^{e,i}$, σ_o^2, and σ_v^2. We will assume that the road network itself and $v_{\max}$ are given.

5.2 Particle MCMC Data Association (PMCMCDA)

Particle MCMC Data Association is perhaps easiest understood as Particle MCMC algorithm with a Resample-Move Particle Filter as applied to MCMC Data Association algorithm [Oh et al., 2004]. In PMCMCDA, MCMC is used in an outer loop to explore the space of possible data associations using the move-types described in Oh et al. [2004], while a Resample-Move Particle Filter is applied in each MCMC iteration to simultaneously approximate the likelihood of all detections and propose candidate state sequences and static parameters. Unlike previous methods, this algorithm is designed to approximate the posterior over data associations, target states, and model parameters, and unlike Oh et al. [2004], non-linear non-Gaussian target dynamics can be modeled.

The intuition behind Particle MCMC Data Association is that while Particle Filters are most effective at tracking dynamic states, the number of possible data associations is too large to expect

particles representing the most likely candidates to be proposed. With Particle MCMC, we are now able to place that burden on an outer loop MCMC algorithm. On the other hand, designing a proposal distribution for static parameters that closely mirrors the posterior would require knowing beforehand what likely state sequences were. However, these depend highly on the data association, and thus we are motivated to believe that a Resample-Move Particle Filter would be more capable at estimating the posterior distribution over parameters than the outer loop MCMC algorithm.

Let us now describe the algorithm itself as given in Algorithm 9. Suppose we have the ith MCMC state $\rho_i = (a_{1:T}, \theta, X_{1:T})_i$ containing a data association, static parameters, and the states of all tracks when they are active as well as an estimate $\hat{Z} = \hat{P}(Y_{1:t}|a_{1:T,i})$. The PMCMCDA algorithm 9 first proposes a move-type selected from those in Table 5.1, then samples a new data assocatiion $\overline{a}_{1:T}$ using the proposal distribution given by the move-type. As in Oh et al. [2004], we will assume that a track necessarily generates a detection when it is born and when it dies. Thus, knowing $a_{1:T}$ is equivalent to knowing t_k^{start} and t_k^{end}.

The algorithm then runs a Resample-Move particle filter (Algorithm 10) targeting $P(X_{1:t}, \theta|Y_{1:t}, \overline{a}_{1:T})$ for $t = 1 \dots T$. This is done by first initializing $\theta^{(p)} \sim P(\theta^{(p)})$ and $X_1^{(p)} \sim q(X_1^{(p)}|Y_1, a_1)$ where $q(X_1^{(p)}|Y_1, a_1)$ is a dynamics model-specific proposal distribution. In implementation, this proposal distribution samples each active track's new state independently (i.e. each $x_1^{(k)}$ is sampled independently given $a_1^{-1}(k)$), and the joint state of all active tracks becomes X_1. The filter continues by proposing new states $X_t^{(p)}$, reweighting them, resampling from the particle pool, and finally performing a Gibbs move on the parameters θ for each particle given the particle's history from $1 \dots t$.

Once the filter is complete, the weights $\overline{w}_t^{(p)}$ are used to compute an estimate for $P(Y_{1:T}|\overline{a}_{1:T})$ called $\overline{Z}$. Furthermore, a single particle $(\overline{X}_{1:T}, \overline{\theta})$ is sampled from the final particle pool $\{X_{1:T}^{(p)}, \theta^{(p)}\}_{p=1}^{P}$ to become part of the outer MCMC state $\overline{\rho}$. Then, the MCMC acceptance ratio α is calculated and $\overline{\rho}$ is accepted or rejected based on it. These steps are repeated until convergence.

While Particle MCMC is shown to target the correct posterior when using a vanilla Particle Filter in Andrieu et al. [2010], its analysis did not account for a Resample-Move type Particle Filter. The proof for this scenario is given in Appendix A.2.

Move name	Reverse move	Move description
Birth	Death	Let k be a new unique track id. Choose a time t and an unassigned detection y_t^i and assign it to track k, then continue as in the Extend move.
Death	Birth	Choose a track k uniformly at random. Assign all its observations to clutter.
Extend	Reduce	Choose a track k uniformly at random. Sample a continuation time step t no more than $t_{\max}$ time steps after the end of the track k ($t_{\max}$ is a hyperparameter chosen beforehand). Sample an unassigned detection uniformly at random from those reasonably close and assign it to k. Sample a new t after k's new final time step with probability p_{cont} (another hyperparameter), else stop.
Reduce	Extend	Choose a track k uniformly at random. Sample a time t between the start and end of the track. Assign all observations assigned to k after t to clutter.
Merge	Split	Choose 2 tracks k and k' such that k' starts less than $t_{\max}$ time steps after k ends. Assign all of k's detections to k
Split	Merge	Choose a track k uniformly at random. Sample a time step t from the times k is active. Assign all observations assigned to k after t to a new track k'
Swap	Swap	Choose 2 tracks k and k' such that there is an overlap in their active times. Choose time step t such both tracks are active, then assign all detections assigned to k after t to k' and likewise for k' to k.
Update	Update	Perform a Reduce then an Extend to the same track.

Table 5.1: MCMC moves used in proposing new data associations

Algorithm 9 A single iteration of Particle MCMC Data Association

Input: MCMC state $\rho_i = (X_{1:T}, a_{1:T}, \theta)_i$ and data likelihood approximation $\hat{Z}_i = \hat{P}(Y_{1:T}|a_{1:T,i})$
Output: MCMC state $\rho_{i+1} = (X_{1:T}, a_{1:T}, \theta)_{i+1}$

1: Sample a move type $m \sim q(m|\rho_i)$
2: Sample new data association $\overline{a}_{1:T} \sim q(\overline{a}_{1:T}|\rho_i, m)$
3: Run Algorithm 10 with $\overline{a}_{1:T}$ fixed to generate weights $\overline{w}_t^{(p)}$ for all t, p and candidate state sequences/parameters $\{X_{1:T}^{(p)}, \theta^{(p)}\}_{p=1}^P$
4: Approximate data likelihood $\overline{Z} = \hat{P}(Y_{1:T}|\overline{a}_{1:T}) = \prod_{t=1}^T \frac{1}{P} \sum_{p=1}^P \overline{w}_t^{(p)}$
5: Sample a state sequence and parameter $\overline{X}_{1:T}, \overline{\theta}$ from $\{X_{1:T}^{(p)}, \theta^{(p)}\}_{p=1}^P$ using weights $w_T^{(p)}$. Let $\overline{\rho} = (\overline{X}_{1:T}, \overline{a}_{1:T}, \overline{\theta})$
6: Calculate acceptance ratio

$$\alpha = \frac{\overline{Z} P(\overline{a}_{1:T}) q(m|\overline{\rho}) q(a_{1:T,i}|\overline{\rho}, m)}{\hat{Z}_i P(a_{1:T,i}) q(m|\rho_i) q(\overline{a}_{1:T}|\rho_i, m)} \tag{5.2}$$

7: Sample $u \sim \text{Unif}(0, 1)$. If $u < \alpha$, let $\rho_{i+1} = \overline{\rho}$, else ρ_i

Algorithm 10 A single iteration of PMCMCDA's Resample-Move Particle Filter

Input: Weighted particles $\{X_{1:t}^{(p)}, \theta^{(p)}\}_{p=1}^{P}$ approximating $P(X_{1:t}, \theta | Y_{1:t}, a_{1:t})$, new detections Y_{t+1}

Output: Weighted particles $\{X_{1:t+1}^{(p)}, \theta^{(p)}\}_{p=1}^{P}$ approximating $P(X_{1:t+1}, \theta | Y_{1:t+1}, a_{1:t+1})$

1: Propagate: sample $\overline{X}_{t+1}^{(p)} \sim q(\overline{X}_{t+1}^{(p)} | Y_{t+1}, X_t^{(p)}, \theta^{(p)}, a_{t+1})$ for all particles p

2: Reweight: Calculate particle weights

$$\overline{w}_{t+1}^{(p)} = \frac{P(Y_{t+1} | \overline{X}_{t+1}^{(p)}, a_{t+1}, \theta^{(p)}) P(\overline{X}_{t+1}^{(p)} | X_t^{(p)}, \theta^{(p)})}{q(\overline{X}_{t+1}^{(p)} | Y_{t+1}, X_t^{(p)}, \theta^{(p)}, a_{t+1})} \tag{5.3}$$

3: Resample: draw P particles $\{X_{t+1}^{(p)}\}_{p=1}^{P}$ from $\{\overline{X}_{t+1}^{(p)}\}_{p=1}^{P}$ such that the expected number of times $\overline{X}_{t+1}^{(p)}$ is replicated is $P\overline{w}_{t+1}^{(p)}$ (e.g., via Residual Resampling)

4: Move: Perform a Gibbs move on $\theta^{(p)}$ by sampling $\theta^{(p)} \sim P(\theta^{(p)} | X_{1:t+1}^{(p)}, Y_{1:t+1}, a_{1:t+1})$

5.3 Particle Filter Data Association (PFDA)

While PMCMCDA is modeled after Particle MCMC, Particle Filter Data Association (PFDA) is best seen as a typical Resample-Move particle filter with a proposal distribution $q(X_{t+1} | X_t, Y_{t+1})$ and Reversible Jump MCMC performed as in Resample-Move.

Particle Filter Data Association was developed after PMCMCDA in order to address the latter's shortcomings. In particular, PMCMCDA suffers from difficulty in proposing data associations that are of reasonable likelihood under the dynamics model due to its data association proposal distribution being based on the same observation-to-observation distance heuristics from Multiple Hypothesis Tracking and Joint Probabilistic Data Association. In models such as the random walk model, a detection can occur in any state with non-negligible likelihood and such distance-based heuristics become ineffective.

Instead, PFDA was designed with the idea that $X_t^{(p)}$, the state of all targets at time t, severely restricts the number of likely data associations at time $t+1$, and thus even if the number of potential data associations is high, most can immediately be disregarded by building a_{t+1} in an almost greedy manner given X_t. In addition, by structuring PFDA as a Particle Filter, we obtain an online algorithm as opposed to one where T is fixed.

Like PMCMCDA, a particle $\rho_t = (X_{1:t}, \theta, a_{1:t})$ is defined to contain the hidden state of all active targets, and the data associations mapping targets to detections up and including to time t. Unlike PMCMCDA, a target does not necessarily generate a detection at its final time step, but we will

still assume it does upon birth.

Suppose that $q(\rho_{t+1}|\rho_t, Y_{t+1})$ is known and can be sampled from. We then use the traditional Particle Filtering algorithm presented in Algorithm 11 to propagate particles forward in time and resample them, followed by an MCMC kernel intended to ensure the particle set does not collapse to a handful of hypotheses. For a more detailed discussion on the wide array of modifications that can be done to Particle Filters, we direct the reader to Doucet and Johansen [2009].

The novelty of PFDA is in the proposal distribution in Algorithm 12. In words, the algorithm begins by assigning detections to existing, active targets in a 'pseudo-greedy' manner by iterating through each new detection and proposing an association between an unassociated target and the new detection based on how likely it is that the former generated the latter. Constant weight is also given to new targets and clutter. This completes proposals for a_{t+1}, $\#\text{new}_{t+1}$, and $\#\text{clut}_{t+1}$. After all detections have been assigned, all targets that were not assigned a detection are eligible for deletion, and thus their existence at time $t+1$ is proposed (i.e., whether or not they are in alive_{t+1}). Finally, new states $x_{t+1}^{(k)}$ are proposed for each target in alive_{t+1} given $y_{t+1}^{(a_{t+1}^{-1}(k))}$.

The RJMCMC moves applied to each particle after resampling are similar to those in PMCMCDA but are far less complex (see Table 5.2). Unlike PMCMCDA, each move affects at most one time step and target (except for the Parameters move), which results in the ability to quickly evaluate the Metropolis-Hastings acceptance ratio. The Parameters move, on the other hand, is a Gibbs move (that is, θ is sampled from the posterior $P(\theta|X_{1:t}, Y_{1:t}, a_{1:t})$) and thus is accepted with probability 1, meaning no acceptance ratio need be calculated [Fearnhead, 2002]. Finally, since all variables are proposed directly and no transformation is taking place, h is always the identity function and its determinant is always 1.

As PFDA is simply a Resample-Move Particle Filtering algorithm, the proof of its convergence relies only on the conditions necessary for such an algorithm to converge. These conditions from [Gilks and Berzuini, 2001, Theorem 2] can best be summarized as for each X_t and Y_{t+1}

1. $P(X_{t+1}|X_t) > 0, P(Y_{t+1}|X_{t+1}) > 0 \Rightarrow q(X_{t+1}|X_t, Y_{t+1}) > 0$
2. $\text{Var}_{P(X_{t+1}|X_t, Y_{t+1})}[w(X_{t+1})] < \infty$

We note that Algorithm 12 satisfies both of these requirements so long as the per-track proposal distributions $q(x_{t+1}^{(k)}|x_t^{(k)}, y_{t+1}^{(a_{t+1}^{-1}(k))})$ satisfy them as well. We can easily see this as the proposal

Move name	Reverse move	Move description
Birth	Death	Let k be a new unique target id. Choose a time t and an unassigned detection y_t^i and assign it to target k. Sample an initial state $x_t^{(k)} \sim q(x_t^{(k)}\|y_t^i)$.
Death	Birth	Choose a target uniformly at random from those of length 1. Delete that target's state and assign its detection to clutter.
Extend	Reduce	Choose a target uniformly at random from those that have already terminated and let t be the time of its final state. Sample whether or not it was detected with probability p_{obs}, and if so choose a detection from those assigned to clutter with weights $w^i = P(y_{t+1}^i\|x_t)$. Sample a new state $x_{t+1}^{(k)} \sim q(x_{t+1}^{(k)}\|y_t^i, x_t^{(k)})$.
Reduce	Extend	Choose a target uniformly at random from those of at least length 2. Remove the target's final state and assign any detection associated with that state to clutter.
Resample	Resample	Choose a target k and time t when k is active. Resample $x_t^{(k)}$ given $x_{t-1}^{(k)}$ and y_t.
Parameters	Parameters	Perform a Gibbs move on all static parameters

Table 5.2: RJMCMC moves in Particle Filter Data Association

distribution concerns itself only with variables who have a large but finite number of possibles, namely $a_{t+1}, \#\text{new}_{t+1}, \#\text{clut}_{t+1}$.

Algorithm 11 A particle filtering algorithm with Resample-Move

Input: a set of weighted particles for time $P(\rho|Y_{1:t})$, $\{w_t^{(p)}, \rho^{(p)}\}_{p=1}^N$, new detections Y_{t+1}

Output: a set of weighted particles for time $P(\rho|Y_{1:t+1})$, $\{w_{t+1}^{(p)}, \rho^{(p)}\}_{p=1}^N$

1: **for** each particle ρ_t in $\{\rho^{(p)}\}$ **do**
2: sample its successor state $\overline{\rho}_{t+1} \sim q(\overline{\rho}_{t+1}|\rho_t, Y_{t+1})$
3: set $\overline{w}_{t+1} = \frac{P(Y_{t+1}|\overline{\rho}_{t+1})P(\overline{\rho}_{t+1}|\rho_t)}{q(\overline{\rho}_{t+1}|\rho_t, Y_{t+1})} w_t$
4: Resample N new particles from $\{\overline{\rho}_{1:t+1}^p\}$ via weights $\overline{w}_{t+1}^{(p)}$; Give these new particles $w_{t+1}^{(p)} = \frac{1}{N}$ and call them $\rho_{1:t+1}^{(p)}$
5: Run each resampled particle through an MCMC kernel $K(\rho'_{1:t+1}|\rho_{1:t+1})$ targeting $P(\rho_{1:t+1}|Y_{1:t+1})$

5.4 Experiments

5.4.1 Particle MCMC Data Association

In this section, we consider the empirical performance of Particle MCMC data association on the models described in Section 5.1. We will use synthetically generated data created by forward

Algorithm 12 Importance sampling distribution for a new particle

Input: a particle $\rho_t = (X_t, a_t, \theta)$, new detections Y_{t+1}
Output: a new particle ρ_{t+1}

1: **for** detection i in Y_{t+1} **do**
2: propose a target k to generate y_t^i from targets active at time t not yet assigned at time $t+1$, a new target, and no target at all based weights $\omega_t^{(k)} = P(y_{t+1}^i | x_t^{(k)})$
3: **for** active targets k at time t not assigned to a detection **do**
4: propose whether k ends at time t or not
5: **for** targets k continuing to $t+1$ **do**
6: propose a new state via $q(x_{t+1}^{(k)} | x_t^{(k)}, y_{t+1}^{a_{t+1}^{-1}(k)})$
7: Let $\rho_{t+1} = (X_{t+1}, a_{t+1}, \theta_t)$

symbol	description	value
T	number of time steps	6,10
n^{states}	number of states	20
λ^{false}	mean number of false detects	1.0
λ^{new}	mean number of new targets	0.3
p^{obs}	probability of detection	0.8
p^{end}	probability of target ending	0.03
n^{targets}	number of active targets	≥ 2
p^{true}	prob. of detecting true state	0.99
p^{move}	prob. of changing state	0.33
	road network shape	3x3 grid
	road length	1.0
σ_v^2	variance of vehicle velocity	0.05
σ_o^2	variance of detection position	0.01
$v_{\max}$	maximum vehicle velocity	0.33
$n^{\text{particles}}$	number of particles	500

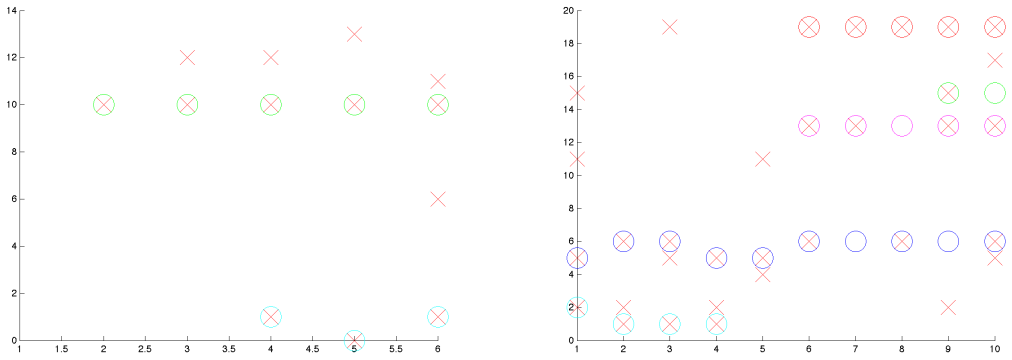
Table 5.3: Default parameters used to generate synthetic data. The top box contains parameters common to both models, while the next two boxes contain parameters relating to the Random Walk and Road models in specific. The final box contains runtime parameters of the algorithm.

sampling the model using the parameters describe in Table 5.3 with exception of the number of vehicles which is bounded above and below. The results presented here do not vary λ^{false}, λ^{new}, p^{true}, or p^{move}.

Random Walk

In this section, we present results for two runs, one with $T = 6$ and the latter $T = 10$. A quick visualization of both can be found in Figure 5.1.

Despite the similarity of the two scenarios, PMCMCDA's performance varies greatly. While it



(a) $T = 6$ (b) $T = 10$

Figure 5.1: Visualization of the two random walk scenarios used in evaluating Particle MCMC Data Association. Time is along the x-axis, state along the y-axis; crosses are detections and true states are circles.

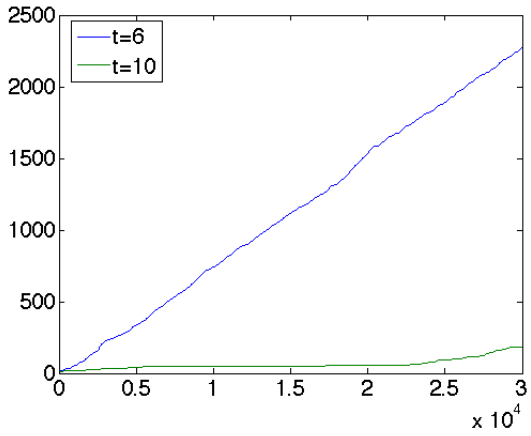
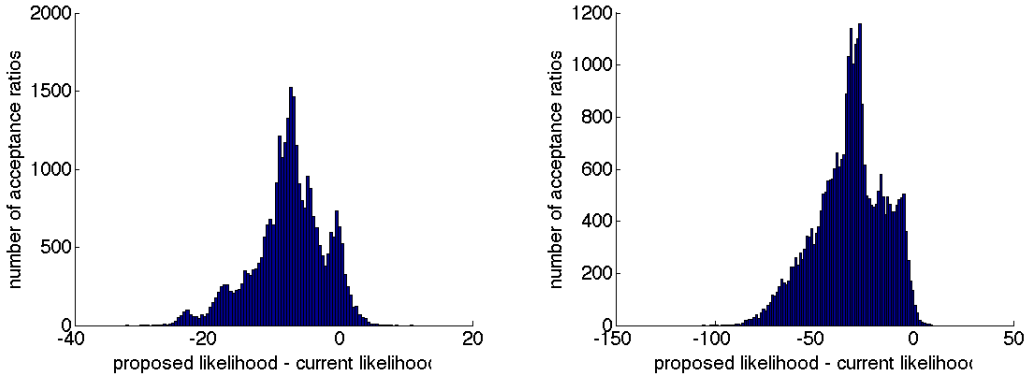


Figure 5.2: Number of accepted proposals as a function of the number of iterations for two random walk scenarios with time lengths 6 and 10.

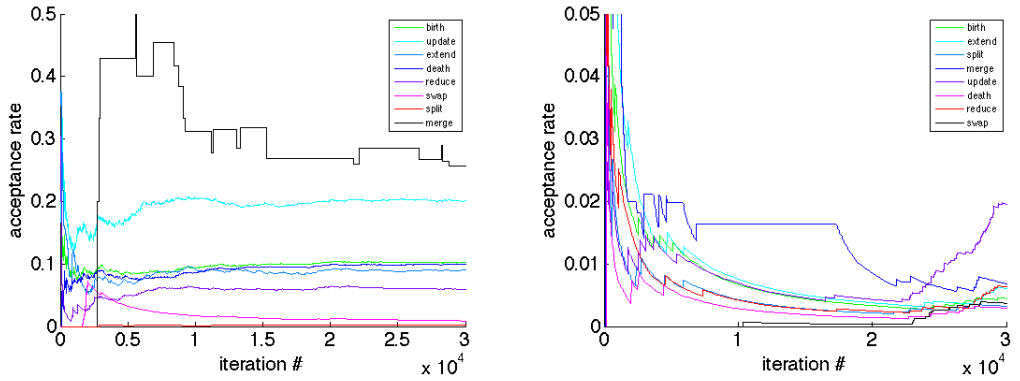
is able to mix quickly and successfully reach a region of high probability when $T = 6$, it is entirely unable to escape local optima when $T = 10$. Figure 5.2 provides a stark contrast between the two scenarios; while slightly less than 10% of proposals are accepted when $T = 6$, less than 1% are accepted when $T = 10$.

Perhaps the first and most obvious culprit for the algorithm's failure when $T = 10$ is the design of the data association proposal distribution. Figure 5.3 compares histograms of $\log \hat{P}(Y_{1:T}, \bar{a}_{1:T}) / \hat{P}(Y_{1:T}, a_{1:T,i})$. Notice that although both plots share similar shapes, the former peaks around -6 while the latter peaks around -25. In consequence, the probability of accepting a proposal is nearly 0 the majority



(a) $T = 6$ (b) $T = 10$

Figure 5.3: Likelihood of proposed data association minus likelihood of current MCMC state. Note the difference in scale.

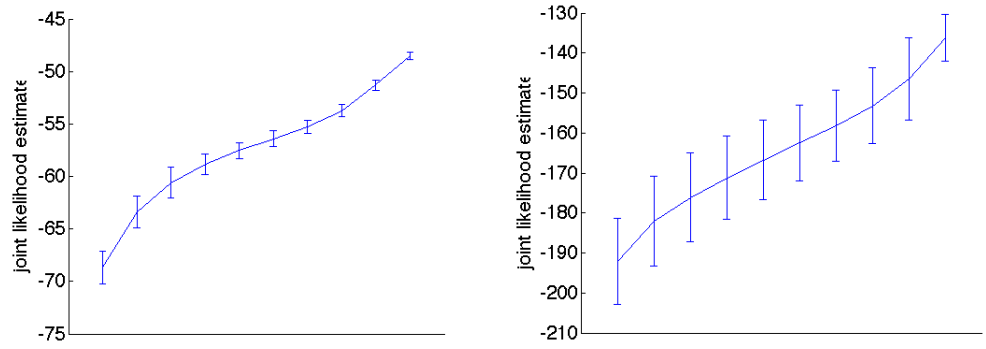


(a) $T = 6$ (b) $T = 10$

Figure 5.4: Percentage of proposals accepted for each move type as a function of the number of iterations. Note the difference in scale.

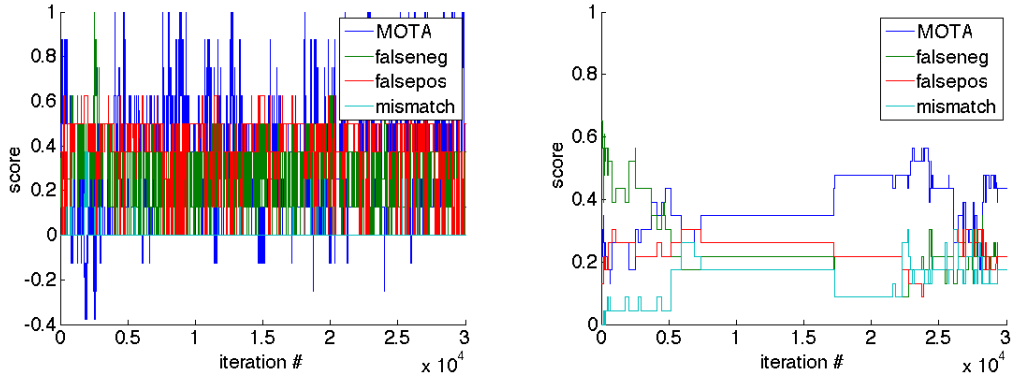
of the time when $T = 10$. Looking at Figure 5.4, we see that this is not the fault of any single move-type; rather, all moves suffer an extremely low acceptance rate. The sheer number of possible data associations makes designing a "good" proposal distribution very difficult.

Another culprit is the accuracy of the Particle Filter's approximation to the observation likelihood $P(Y_{1:T}|\overline{a}_{1:T})$, $\bar{Z}$. It is important to understand that even with a perfect approximation, Particle MCMC can only hope to do well as an MCMC algorithm without approximation; the particle approximation can only hurt. Examining Figure 5.5, we see that the variance of the Particle Filter's approximation deteriorates as the data association becomes more unlikely. In consequence, the MCMC chain is apt to get stuck when the joint likelihood $P(Y_{1:T}, a_{1:T})$ is overestimated. When



(a) $T = 6$ (b) $T = 10$

Figure 5.5: Standard deviation of joint likelihood approximation as a function of the mean. Note the difference in scale.



(a) $T = 6$ (b) $T = 10$

Figure 5.6: CLEAR score of the current MCMC sample as a function of number of iterations.

$T = 6$, this is not an issue as the scenario is so small, but when $T = 10$ the variance is simply too high. While more particles can be added to alleviate the issue, the additional computational burden significantly slows the algorithm.

Finally, it is worth noting that even when well mixing, PMCMCDA does not generate a posterior wherein the true hidden states and parameters dominate all other configurations. Rather, there exist many likely solutions which the MCMC chain explores. In the simple scenario of $T = 6$, this results in a noisy, time-varying estimate to the CLEAR metric (Figure 5.6a); when $T = 10$, the algorithm's inability to reach a high density region results in a low CLEAR score throughout (Figure 5.6b). This is not a fault of PMCMCDA itself but rather simply a result of the model formulation.

In summary, PMCMCDA is an extremely computationally intensive algorithm highly dependent both on the accuracy of the Particle Filter and the data association proposal distribution. Careful choice of both is necessary to perform well, and even then the most likely posterior solution need not correspond to a high CLEAR score.

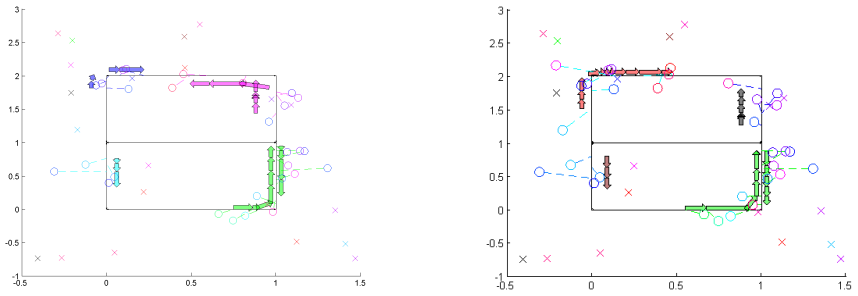
Traffic

In this section, we consider a Traffic model generated from parameters used in Table 5.3 with $T = 20$. Unlike the Random Walk model, which is initialized with a data association wherein all detections are marked as false detections, the Traffic Model initializes by running a greedy algorithm for clustering detections. In particular, the greedy algorithm initializes a target by selecting a time step t, picking a false detection uniformly at random, then continuing to $t + 1$ to pick the nearest detection to the previous one in L_2 distance, and so on until the end of time. This is repeated sequentially until no more targets can be created.

Surprisingly, PMCMCDA does significantly better in the Traffic domain than in the simple Random Walk. This may be due to the initialization scheme, which Figure 5.10b shows to have been quite well chosen already in terms of the CLEAR score. However, it should be noted that this data association was not by any means correct; Figure 5.9b shows that the (approximate) joint likelihood of the initial data association, $P(Y_{1:T}, a_{1:T,1})$, was 30 orders of magnitude less than the true value. Even still, PMCMCDA is able to converge to a high probability region and generate reasonable hypotheses.

The number of proposals accepted in this model was observed to be between the two scenarios presented in the Random Walk model, achieving roughly 3% of proposals accepted (Figure 5.8a). Figure 5.8b shows that some moves were far more successful than others, particularly the Update and Merge moves. The proposals themselves, however, were not overly unlikely; Figure 5.9a shows that most samples were within 10 orders of magnitude of the current state, but many samples were significantly less likely than the current MCMC state.

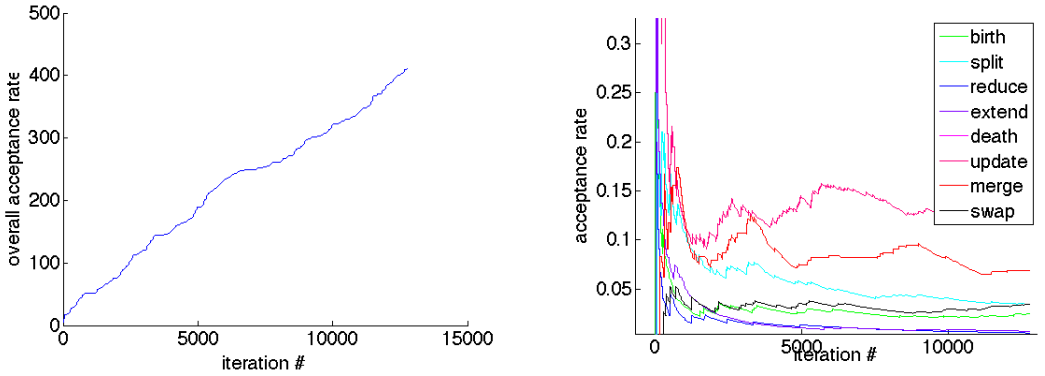
Finally, the same results regarding data association likelihood and Particle Filter approximation accuracy hold true. That is, the more unlikely a data association, the more noisy the Particle Filter's approximation to the value of $P(Y_{1:T}|a_{1:T})$. In spite of this, Particle MCMC Data Association achieved adequate performance in both acceptance rate and CLEAR.



(a) Ground truth

(b) Hypothesis at iteration 8,000

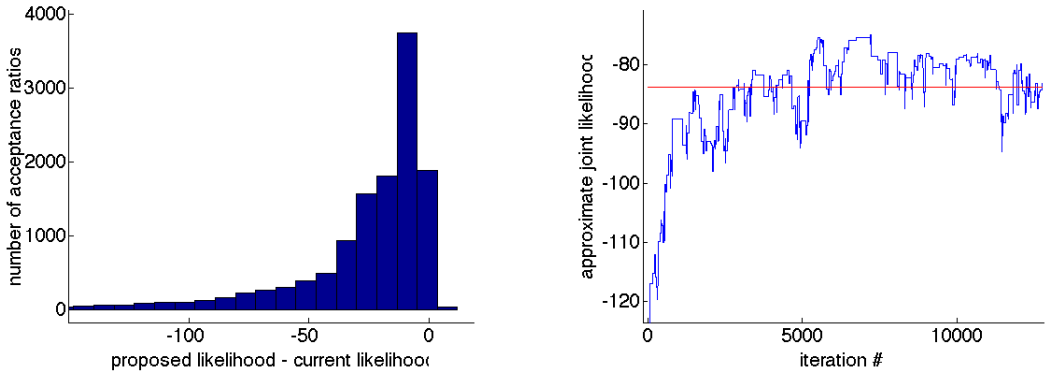
Figure 5.7: Visualizations of the Traffic scenario. Crosses are false detections, circles are correct detections, and each color of arrow represents the state sequence of one vehicle. Subsequence arrows denote time steps. Cross colors have no relation to arrow color.



(a) Acceptance counts as a function of the number of iterations

(b) Per-move acceptance rates as a function of the number of iterations

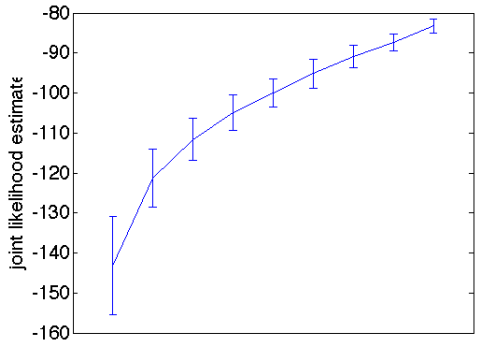
Figure 5.8: Acceptance Statistics for the Traffic model



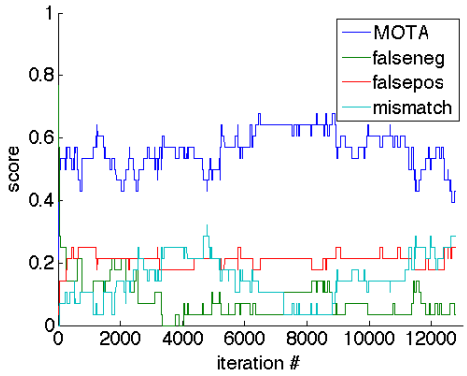
(a) Histogram of log ratio of proposal joint likelihood over current MCMC state joint likelihood

(b) Histogram of joint likelihood of current MCMC state. The red line is the joint likelihood of ground truth.

Figure 5.9: Histograms of MCMC state likelihood.



(a) Mean and standard deviation of joint likelihood approximations for the 10th, 20th, ..., 90th percentiles



(b) CLEAR score as a function of the number of iterations

5.4.2 Particle Filter Data Association

Random Walk

We now analyze the performance of Particle Filter Data Association across 3 dimensions: n^{partices}, p^{obs}, and λ^{false}. In all experiments, synthetic data is generated using parameters detailed in the previous subsection in Table 5.3 with the exception of T which is set to 50 and n^{targets} which is between 2 and 11 except when stated otherwise. Unlike Particle MCMC Data Association,λ^{false}, λ^{new}, p^{true}, and p^{move} are assumed unknown and must be inferred.

We compare the performance of our algorithm in three distinct scenarios. The first, MLE, 0 MOVES is when the algorithm is seeded with parameters maximizing the likelihood of true, unobserved data and Resample Move is disabled. This is intended to given an idea of the 'best' our algorithm can achieve. The second, PRIOR, 0 MOVES considers the case that parameters are drawn from the prior and no Resample Move is applied thereafter. The final, PRIOR, 5 MOVES applies 5 Resample Move MCMC iterations are applied to each particle at every time step.

We begin by examining expected value for each static parameter as a function of time when $T = 250$. As can be seen in Figure 5.10, both PRIOR, 0 MOVES and PRIOR, 5 MOVES largely converge after roughly 50 time steps. While PRIOR, 5 MOVES converges towards a more accurate value, it still differs significantly from the data's true value. Other parameters, not seen, converge similarly.

The reason for this failure of convergence to the true parameter is Particle Degeneracy, the

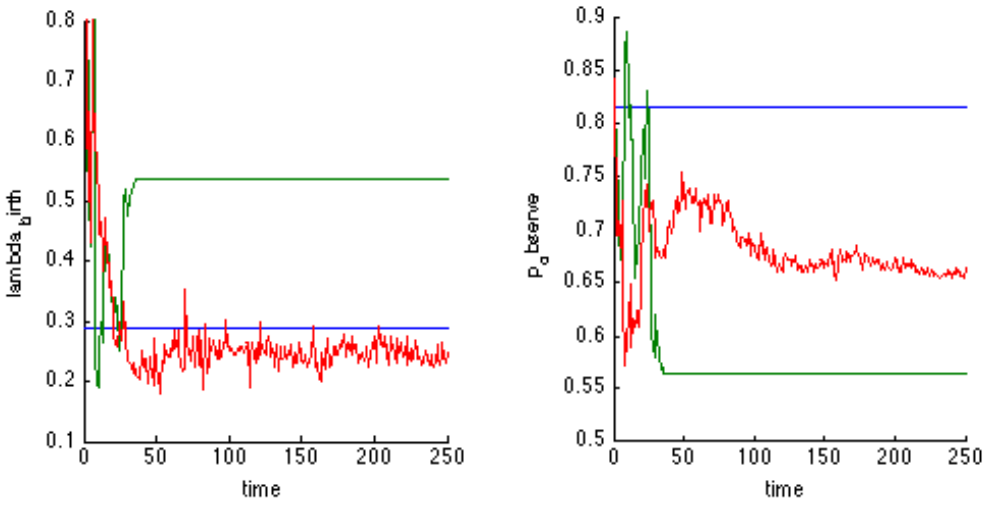


Figure 5.10: $\mathbb{E}[\lambda^{birth}|Y_{1:t}]$ (a) and $\mathbb{E}[p^{obs}|Y_{1:t}]$ (b) as a function of time in the random walk model. Blue is ground truth, green is PRIOR, 0 MOVES red is PRIOR, 5 MOVES .

phenomena wherein all particles eventually inherit the same ancestor due to resampling. How Particle Degeneracy affected each scenario, however, differs. PRIOR, 0 MOVES samples parameters only at $t = 0$, and thus can only lose diversity to resampling. On the other hand, PRIOR, 5 MOVES can alter parameters as time progresses; however, as t grows, resampling decreases the particle diversity of $\{x_{t-L}^{(k)}\}_k$ for large L. Unless an increasing number of MCMC iterations is also applied, the Particle Filter converges to a single hypothesis for $\{x_\tau^{(k)}\}_{k,1\leq\tau\leq t-L}$. The inability to well approximate $P(\{x_\tau^{(k)}\}_{k,1\leq\tau\leq t-L}|Y_{1:t})$ in turn biases $P(\theta|\{x_\tau^{(k)}\}_{k,1\leq\tau\leq t-L}, Y_{1:t})$, causing the Particle Filter to peak at an incorrect value as seen in Figure 5.10.

The inclusion of MCMC seems to have had little effect on the particle filter's accuracy in estimating the states of the targets themselves. In spite of having less error in parameter values, the MOTA score is nearly identical with and without MCMC, as seen in Figure 5.11.

Varying $n^{\text{particles}}$, we see in Figure 5.13 that our algorithm's performance is largely independent of the number of particles, at least within the range of 100 to 1,000. Further investigation indicates that PRIOR, 0 MOVES has a slightly higher mismatch and false negative rate, but results are too close to be conclusive. The difference between known and unknown parameters is noticable but not significant. Further investigation reveals that the Particle Filter becomes more unstable (Figure 5.12) as the number of active targets increases. We may formalize this via the **Effective Sample**

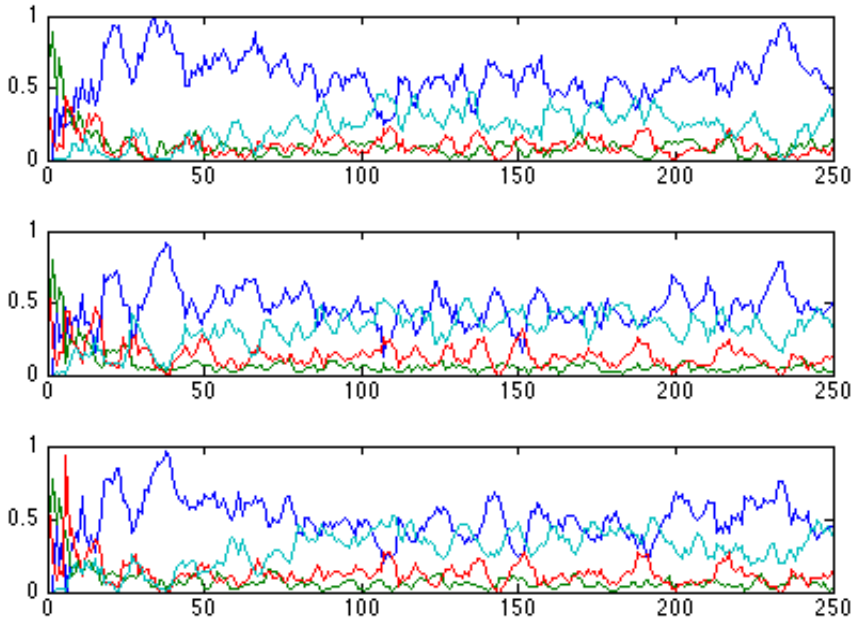


Figure 5.11: CLEAR as a function of time for MLE, 0 MOVES PRIOR, 0 MOVES and PRIOR, 5 MOVES respectively in the random walk model. MOTA is blue, false negative rate is green, false positive rate is red, and mismatch rate is aqua. The CLEAR score is calculated using the last 5 time steps only.

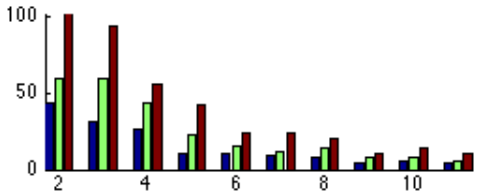


Figure 5.12: 25th, 50th, and 75th percentile of the effective sample size as a function of the true number of active targets in the random walk model with $T = 250$. The true number of particles is 500.

Size, computed as $ESS := 1/\sum_p (\omega^{(p)})^2$ where $\omega^{(p)}$ is the (normalized) weight of particle p before resampling (that is, $\overline{w}^{(p)}$ normalized).

Varying p^{obs}, we see in Figure 5.14 that performance is highly dependent on how likely it is for a target to be detected. The benefits of resample move, however, are rather unclear; the performance of PRIOR, 0 MOVES and PRIOR, 5 MOVES is neck-and-neck. A closer look at the large dip in PRIOR, 0 MOVES 's performance for $p^{\text{obs}} = 0.7$ reveals that the static parameters settled in such a way that short tracks and clutter are favored, resulting in a false negative rate 20% and a mismatch rate 14% above their corresponding values in MLE, 0 MOVES .

Varying λ^{false} (Figure 5.15) reveals the sensitivity of the algorithm to the amount of noise in

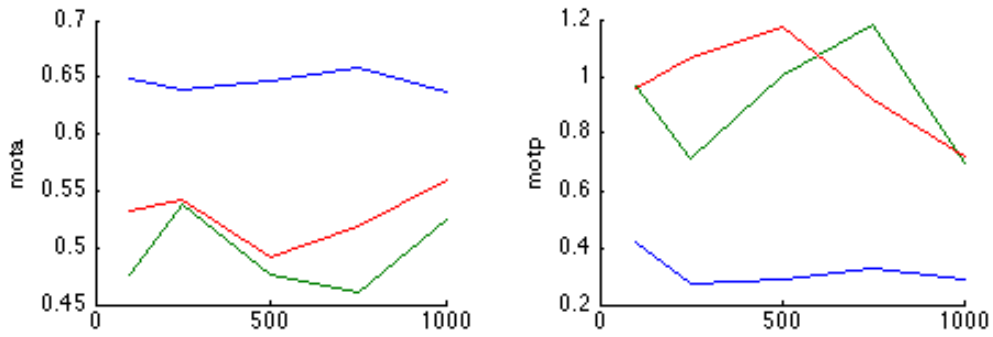


Figure 5.13: MOTA (a) and MOTP (b) as a function of $n^{particles}$. Blue is MLE, 0 MOVES green PRIOR, 0 MOVES and red is PRIOR, 5 MOVES .

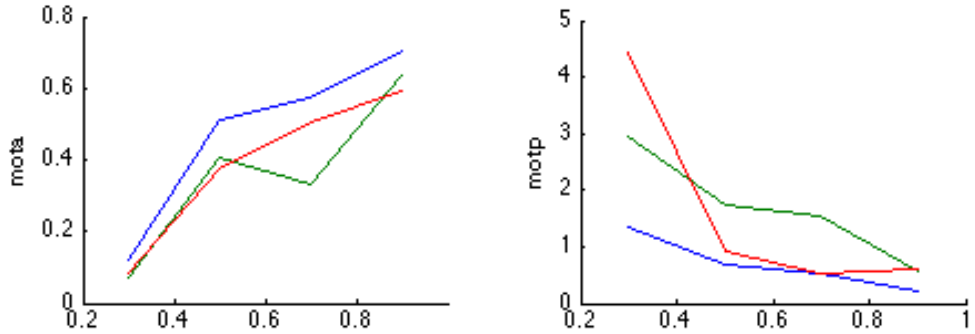


Figure 5.14: MOTA (a) and MOTP (b) as a function of p^{obs} in the random walk model. Blue is MLE, 0 MOVES green PRIOR, 0 MOVES and red is PRIOR, 5 MOVES .

the observations. At an average 4.5 false detects per time step, the MOTA falls to an abysmal -0.8 in both PRIOR, 0 MOVES and PRIOR, 5 MOVES whereas MLE, 0 MOVES achieved slightly below 0. Again, we see that the MCMC moves do not noticably affect performance with runs other than MLE, 0 MOVES having nearly identical mismatch, false positive, and false negative rates.

Overall, experiments indicate that the benefit of MCMC moves to CLEAR score performance is negligible at best. Even given the best possible parameters, the problem of Multiple Target Tracking is difficult in a Sequential Monte Carlo framework, and the burden lies on the designer to choose proposal distributions that reflect the unknown posterior in order to minimize the number of samples necessary.

Traffic

For the Traffic model, we run experiments only with default parameters and $T = 250$.

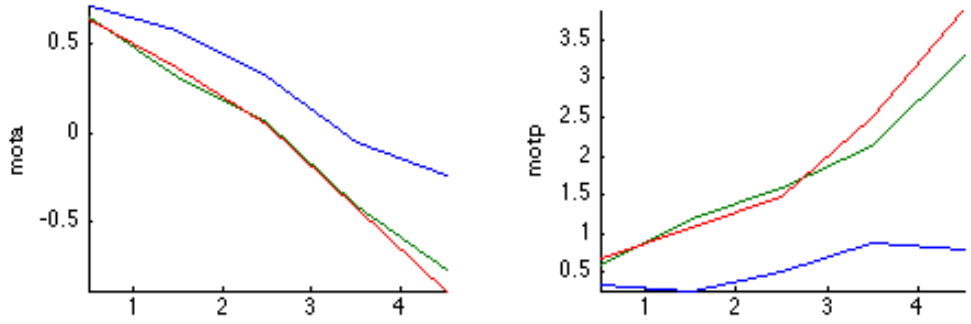


Figure 5.15: MOTA (a) and MOTP (b) as a function of λ^{false} in the random walk model. Blue is MLE, 0 MOVES green PRIOR, 0 MOVES and red is PRIOR, 5 MOVES .

Results from the Traffic model were similar to those found in the Random Walk model. CLEAR scores with MCMC moves performed slightly better than those found without (Figure 5.18), and parameters converged but again not to their correct values (Figure 5.16). When the number of targets is relatively small, particles generate reasonable hypotheses as in Figure 5.19; however, when the number of targets exceeded 5 or more, the exponential variance of multiple, independently moving targets renders hypotheses very difficult to interpret.

Unlike the Random Walk experiments, the Traffic model's runs had a much higher variance in particle weight. We believe that this is due to the low variance of detections in vehicle position and the high number of independently moving vehicles. The consequences of this is an increased rate in particle degeneracy, which led to the convergence of parameters in PRIOR, 0 MOVES in less than 5 iterations (Figure 5.16) and low effective sample size throughout (Figure 5.17).

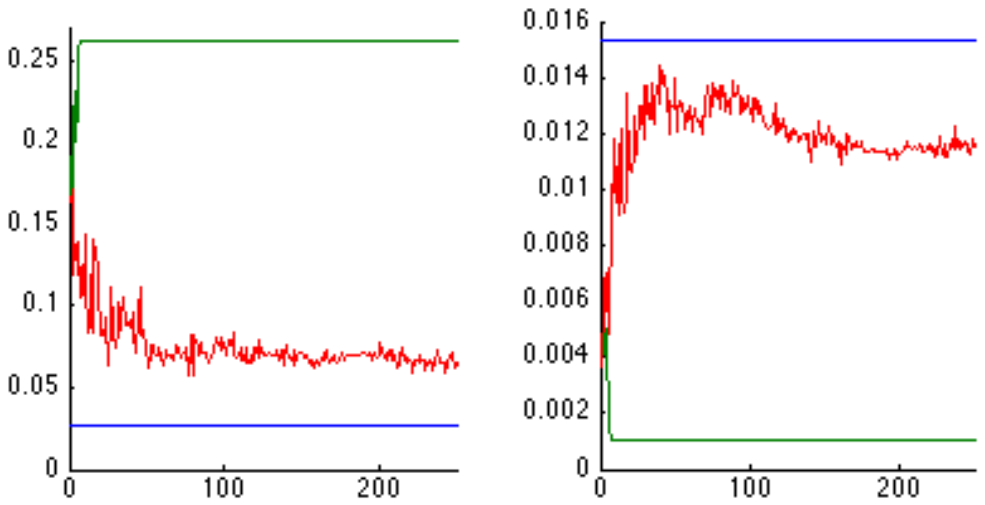


Figure 5.16: $\mathbb{E}[p^{end}|Y_{1:t}]$ and $\mathbb{E}[\sigma_v^2|Y_{1:t}]$ as a function of time in the traffic model. Blue is ground truth, red is PRIOR, 5 MOVES and green is PRIOR, 0 MOVES .

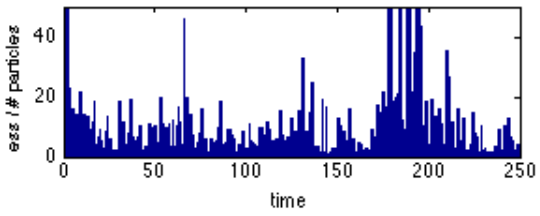


Figure 5.17: Effective sample size as a function of time in the traffic model, clipped at 50. The true number of particles is 500. It is not uncommon for the effective sample size to drop as low as 2 or 5 whereas over 200 is considered 'healthy.'

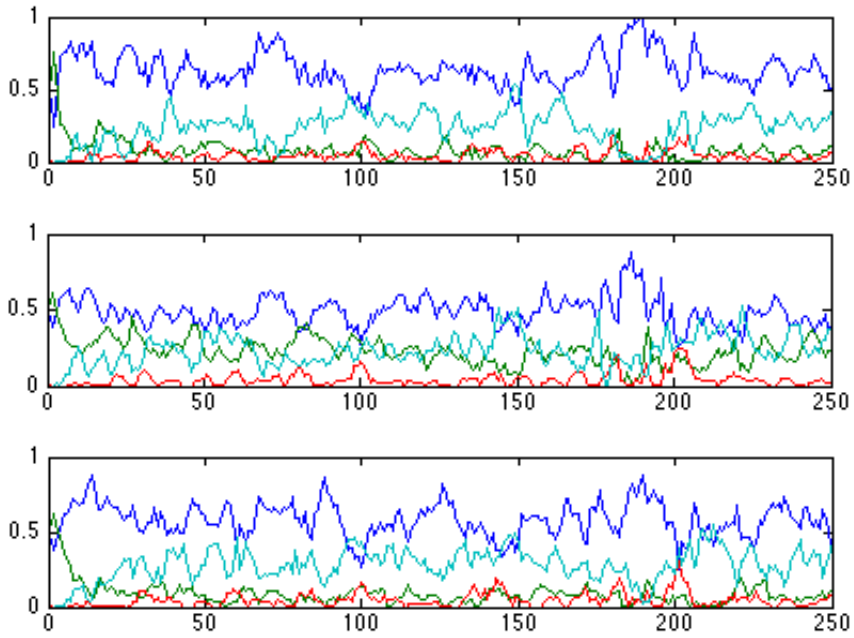


Figure 5.18: CLEAR as a function of time for MLE, 0 MOVES PRIOR, 0 MOVES and PRIOR, 5 MOVES respectively in the traffic model. MOTA is blue, false negative rate is green, false positive rate is red, and mismatch rate is aqua.

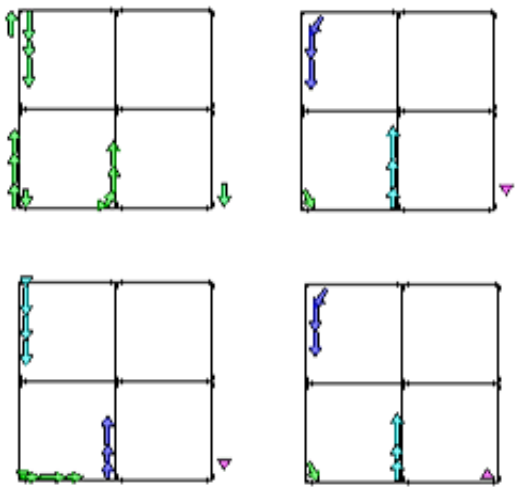


Figure 5.19: True state (top left) and top 3 hypotheses (clockwise from top right) in the traffic model.

5.5 Conclusions

In this work we have formulated a Contingent Bayesian Network model for Multiple Target Tracking and presented two new Monte Carlo algorithms for performing approximate inference over data associations and target states called Particle MCMC Data Association (PMCMCDA) and Particle Filter Data Association (PFDA). Unlike previous algorithms, both PMCMCDA and PFDA are capable of performing inference in models without Linear-Gaussian assumptions, with unknown time-invariant model parameters, and while considering multiple data associations. These algorithms have been shown to approximate the true posterior as the number of iterations and particles, respectively, approaches infinity, and their effectiveness has been tested on two synthetic per target dynamics models.

Particle MCMC Data Association combines Particle MCMC [Andrieu et al., 2010], MCMC Data Association [Oh et al., 2004], and the Resample-Move algorithm [Gilks and Berzuini, 2001] to explore the posterior over data associations, target states, and static parameters given sets of detections over time. By employing a Resample-Move Particle Filter, designing explicit MCMC proposal distributions for the latter two can be avoided and instead be constructed automatically during Filtering. In showing correctness of PMCMCDA, we contribute an extension of the proof presented in Andrieu et al. [2010] to allow for Resample-Move Particle Filters instead of vanilla Particle Filters.

Particle Filter Data Association combines the Resample-Move algorithm [Gilks and Berzuini, 2001] with a new Multiple Target Tracking proposal distribution for independently moving targets. Unlike PMCMCDA, PFDA can take target dynamics into account when constructing new data associations by maintaining a distinct set of target states and data associations in each particle. This has resulted in an online filtering algorithm capable of performing inference for scenarios up to twenty-five times longer than PMCMCDA is capable of.

Appendix A

Proofs

A.1 Resample Move gives unbiased estimates for observation likelihood

In this section, we consider the Resample-Move particle filter of Gilks and Berzuini [2001]. We aim to show, as Pitt et al. [2010] did for Auxiliary Particle Filter, that the Resample-Move particle filter gives an unbiased estimate for the likelihood of the observations for any number of particles. We use the notation given in Table A.1 and at each time step Algorithm 13. Our goal is to prove the following,

Theorem 3. *Let* $\hat{P}(y_{1:T}) = \prod_{t=1}^{T} \hat{P}(y_t|y_{1:t-1})$. *Then* $\hat{P}(y_{1:T})$ *is an unbiased estimator for* $P(y_{1:T})$ *– that is,* $\mathbb{E}[\hat{P}(y_{1:T})] = P(y_{1:T})$

x_t	Unknown state at time t
y_t	Known observation at time t
$x_{1:t}^{(t-1),p}$	Particle p containing states for times 1 to t inclusive where x_1 has been resampled through MCMC kernel K $(t-1)$ times.
$K(x_{1:t}^{(t),p}\|x_{1:t}^{(t-1),p})$	MCMC kernel applied to each particle after each iteration. Has stationary distribution $P(x_{1:t}\|y_{1:t})$
$w(x_{1:t+1}^{(t),p})$	Unnormalized weight for particle $x_{1:t+1}^{(t),p}$
π_t^p	Normalized weight of particle $x_{1:t}^{(t-1),p}$
$\hat{P}(y_t\|y_{1:t-1})$	Approximation to the probability of y_t given all previous observations.
a_t^p	Index of the parent of particle p at time t. See Figure A.1.

Table A.1: Notation

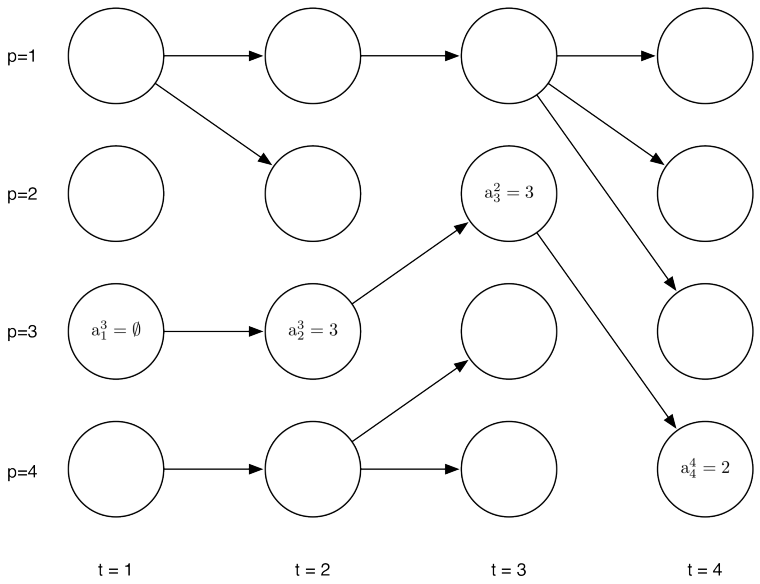


Figure A.1: The parents of one lineage of particles in a Particle Filter

Algorithm 13 An iteration of a Resample-Move Particle Filter

Input: A set of N weighted particles $\{x_{1:t}^{(t-1),p}, \pi_t^p\} \sim P(x_{1:t}|y_{1:t-1})$, observation y_{t+1}

Output: A set of N weighted particles $\{x_{1:t+1|1:t}^{(t),p}, \pi_{t+1}^p\} \sim P(x_{1:t+1}|y_{1:t})$, unbiased estimate of $\hat{P}(y_t|y_{1:t-1})$

1: Resample indices a_t^p according to weights $\pi_t^{p'}$
2: Set $\tilde{x}_{1:t}^{(t-1),p} = x_{1:t}^{(t-1),a_t^p}$
3: Move samples $x_{1:t}^{(t),p} \sim K(x_{1:t}^{(t),p}|\tilde{x}_{1:t}^{(t-1),p})$
4: Propagate samples $x_{t+1}^{(t),p} \sim q(x_{t+1}^{(t),p}|x_{1:t}^{(t),p})$
5: Reweight samples $w(x_{1:t+1}^{(t),p}) = \frac{P(x_{1:t+1}^{(t),p}, y_{1:t+1})}{P(x_{1:t}^{(t),p}, y_{1:t}) q(x_{t+1}^{(t),p}|x_{1:t}^{(t),p})} = \frac{P(y_{t+1}|x_{t+1}^{(t),p}) P(x_{t+1}^{(t),p}|x_t^{(t),p})}{q(x_{t+1}^{(t),p}|x_{1:t}^{(t),p})}$
6: Set $\pi_{t+1}^p = \frac{w(x_{1:t+1}^{(t),p})}{\sum_l w(x_{1:t+1}^{(t),l})}$
7: Estimate $\hat{P}(y_t|y_{1:t-1}) = \frac{1}{N} \sum_p w(x_{1:t+1}^{(t),p})$

Proof. The following proof is in the spirit of Pitt et al. [2010] with modifications made appropriately for Resample-Move. Note that although there are 3 rather long sequences of equations, each of them uses nearly the same logic and is only given for completeness of the argument. After each sequence of equations are the logical steps performed between each line. In addition to the variables already defined, let

- $\mathcal{A}_t = \{(x_{1:t}^{(t-1),p}, \pi_t^p)\}$ be the particles emitted at time t.
- $g(\tilde{x}_{1:t-1}^{(t-2),p}) = \frac{1}{N} \sum_{p=1}^N \pi_t^p \delta(\tilde{x}_{1:t-1}^{(t-2),p} = x_{1:t-1}^{(t-2),p})$ be the distribution from which $\tilde{x}_{1:t-1}^{(t-2),p}$ is drawn.

- $I(x_{1:t-1}^{(t-2)}) = \begin{cases} 1 & t \geq T \\ \int K(x_{1:t-1}^{(t-1)}|x_{1:t-1}^{(t-2)})P(x_t^{(t-1)}|x_{1:t-1}^{(t-1)})P(y_t|x_{1:t}^{(t-1)})I(x_{1:t}^{(t-1)})dx_{1:t}^{(t-1)} & \text{else} \end{cases}$,
be an intermediate integral used to make a concise, recursive form for the following statements without any particular intuitive meaning

We begin by forming a base case at time T,

$$\mathbb{E}\left[\hat{P}(y_T|y_{1:T-1})|\mathcal{A}_{T-1}\right] \tag{A.1}$$

$$=\mathbb{E}\left[\frac{1}{N}\sum_{p=1}^{N} w(x_{1:T}^{(T-1),p})|\mathcal{A}_{T-1}\right] \tag{A.2}$$

$$=\frac{1}{N}\sum_{p=1}^{N}\int \underbrace{g(\tilde{x}_{1:T-1}^{(T-2),p})}_{\text{resampling}}\underbrace{K(x_{1:T-1}^{(T-1),p}|\tilde{x}_{1:T-1}^{(T-2),p})}_{\text{moving}}\underbrace{q(x_T^{(T-1),p}|x_{1:T-1}^{(T-1),p})}_{\text{propagating}}\underbrace{w(x_{1:T}^{(T-1),p})}_{\text{reweighting}} dx_{1:T}^{(T-1),p} d\tilde{x}_{1:T-1}^{(T-2),p} \tag{A.3}$$

$$=\sum_{p=1}^{N}\pi_{T-1}^{p}\int K(x_{1:T-1}^{(T-1)}|x_{1:T-1}^{(T-2),p})q(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})w(x_{1:T}^{(T-1)})dx_{1:T}^{(T-1)} \tag{A.4}$$

$$=\sum_{p=1}^{N}\pi_{T-1}^{p}\int K(x_{1:T-1}^{(T-1)}|x_{1:T-1}^{(T-2),p})q(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})\frac{P(y_T|x_{1:T}^{(T-1)})P(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})}{q(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})}dx_{1:T}^{(T-1)} \tag{A.5}$$

$$=\sum_{p=1}^{N}\pi_{T-1}^{p}\int K(x_{1:T-1}^{(T-1)}|x_{1:T-1}^{(T-2),p})P(y_T|x_{1:T}^{(T-1)})P(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})dx_{1:T}^{(T-1)} \tag{A.6}$$

$$=\sum_{p=1}^{N}\pi_{T-1}^{p}I(x_{1:T-1}^{(T-2),p}) \tag{A.7}$$

$$\tag{A.8}$$

where we use,

1. Definition of $\hat{P}(y_T|y_{1:T-1})$
2. Integral over all intermediate variables
3. Definition of $g(\tilde{x}_{1:T-1}^{(T-2),p})$ in terms of δs and the fact that we can pull out $\frac{1}{N}\sum_{p=1}^{N}$ and notice that all terms in the sum have the same value, thus we can remove it entirely. We thus relabel $\tilde{x}_{1:T-1}^{(T-2),p}$ with $x_{1:T-1}^{(T-2),p}$
4. Definition of $w(x_{1:T}^{(T-1)})$

5. Cancelling out $q(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})$

6. $I(x_{1:T-1}^{(T-2),p}) = \int K(x_{1:T-1}^{(T-1)}|x_{1:T-1}^{(T-2),p})P(y_T|x_{1:T}^{(T-1)})P(x_T^{(T-1)}|x_{1:T-1}^{(T-1)})dx_{1:T}^{(T-1)}$

Next, we perform a recursive step resulting in the same form 1 step backwards in time,

$$=\mathbb{E}\left[\mathbb{E}\left[\hat{P}(y_t|y_{1:t-1})|\mathcal{A}_{t-1}\right]\hat{P}(y_{t-1}|y_{1:t-2})|\mathcal{A}_{t-2}\right] \tag{A.1}$$

$$=\mathbb{E}\left[\left(\sum_{p=1}^{N}\pi_{t-1}^{p}I(x_{1:t-1}^{(t-2),p})\right)\left(\frac{1}{N}\sum_{p=1}^{N}w(x_{1:t-1}^{(t-2),p})\right)|\mathcal{A}_{t-2}\right] \tag{A.2}$$

$$=\mathbb{E}\left[\left(\sum_{p=1}^{N}\frac{w(x_{1:t-1}^{(t-2),p})}{\sum_l w(x_{1:t-1}^{(t-2),l})}I(x_{1:t-1}^{(t-2),p})\right)\left(\frac{1}{N}\sum_{p=1}^{N}w(x_{1:t-1}^{(t-2),p})\right)|\mathcal{A}_{t-2}\right] \tag{A.3}$$

$$=\mathbb{E}\left[\frac{1}{N}\sum_{p=1}^{N}w(x_{1:t-1}^{(t-2),p})I(x_{1:t-1}^{(t-2),p})|\mathcal{A}_{t-2}\right] \tag{A.4}$$

$$=\frac{1}{N}\sum_{p=1}^{N}\int \underbrace{g(\tilde{x}_{1:t-2}^{(t-3),p})}_{\text{resampling}}\underbrace{K(x_{1:t-2}^{(t-2),p}|\tilde{x}_{1:t-2}^{(t-3),p})}_{\text{moving}}\underbrace{q(x_{t-1}^{(t-2),p}|x_{1:t-2}^{(t-2),p})}_{\text{propagating}}\underbrace{w(x_{1:t-1}^{(t-2),p})}_{\text{reweighting}}I(x_{1:t-1}^{(t-2),p})dx_{1:t-1}^{(t-2),p}d\tilde{x}_{1:t-2}^{(t-3),p} \tag{A.5}$$

$$=\sum_{p=1}^{N}\pi_{t-2}^{p}\int K(x_{1:t-2}^{(t-2)}|x_{1:t-2}^{(t-3),p})q(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})w(x_{1:t-1}^{(t-2)})I(x_{1:t-1}^{(t-2)})dx_{1:t-1}^{(t-2)} \tag{A.6}$$

$$=\sum_{p=1}^{N}\pi_{t-2}^{p}\int K(x_{1:t-2}^{(t-2)}|x_{1:t-2}^{(t-3),p})q(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})\frac{P(y_{t-1}|x_{1:t-1}^{(t-2)})P(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})}{q(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})}I(x_{1:t-1}^{(t-2)})dx_{1:t-1}^{(t-2)} \tag{A.7}$$

$$=\sum_{p=1}^{N}\pi_{t-2}^{p}\int K(x_{1:t-2}^{(t-2)}|x_{1:t-2}^{(t-3),p})P(y_{t-1}|x_{1:t-1}^{(t-2)})P(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})I(x_{1:t-1}^{(t-2)})dx_{1:t-1}^{(t-2)} \tag{A.8}$$

$$=\sum_{p=1}^{N}\pi_{t-2}^{p}I(x_{1:t-2}^{(t-3),p}) \tag{A.9}$$

where we used,

1. The previous result and the definition of $\hat{P}(y_{t-1}|y_{1:t-2})$

2. The definition of π_{t-1}^{p}

3. Canceling out $\sum_l w(x_{1:t-1}^{(t-2),l})$

4. Integral over all intermediate variables

5. Definition of $g(\tilde{x}_{1:t-2}^{(t-3),p})$ in terms of δs and the fact that we can pull out $\frac{1}{N}\sum_{p=1}^{N}$ and notice that all terms in the sum have the same value, thus we can remove it entirely.

6. Definition of $w(x_{1:t-1}^{(t-2)})$

7. Cancelling out $q(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})$

8. $I(x_{1:t-2}^{(t-3),p}) = \int K(x_{1:t-2}^{(t-2)}|x_{1:t-2}^{(t-3),p})P(y_{t-1}|x_{1:t-1}^{(t-2)})P(x_{t-1}^{(t-2)}|x_{1:t-2}^{(t-2)})I(x_{1:t-1}^{(t-2)})dx_{1:t-1}^{(t-2)}$

We assume the reader can infer the full recursion going backwards in time. Finally, for $t = 1$

$$\mathbb{E}[\mathbb{E}[\hat{P}(y_{2:t}|y_1)|\mathcal{A}_1]\hat{P}(y_1)] \tag{A.1}$$

$$=\mathbb{E}\left[\left(\sum_{p=1}^{N}\pi_1^p I(x_1^{(0),p})\right)\left(\frac{1}{N}\sum_{p=1}^{N}w(x_1^{(0),p})\right)\right] \tag{A.2}$$

$$=\mathbb{E}\left[\left(\sum_{p=1}^{N}\frac{w(x_1^{(0),p})}{\sum_l w(x_1^{(0),l})}I(x_1^{(0),p})\right)\left(\frac{1}{N}\sum_{p=1}^{N}w(x_1^{(0),p})\right)\right] \tag{A.3}$$

$$=\mathbb{E}\left[\frac{1}{N}\sum_{p=1}^{N}w(x_1^{(0),p})I(x_1^{(0),p})\right] \tag{A.4}$$

$$=\frac{1}{N}\sum_{p=1}^{N}\int \underbrace{q(x_1^{(0),p})}_{\text{propagating}}\ \underbrace{w(x_1^{(0),p})}_{\text{reweighting}}\ I(x_1^{(0),p})dx_1^{(0),p} \tag{A.5}$$

$$=\int q(x_1^{(0)})\frac{P(y_1|x_1^{(0)})P(x_1^{(0)})}{q(x_1^{(0)})}I(x_1^{(0)})dx_1^{(0)} \tag{A.6}$$

$$=\int P(y_1, x_1^{(0)})I(x_1^{(0)})dx_1^{(0)} \tag{A.7}$$

$$=\int P(y_1)P(x_1^{(0)}|y_1)\left(\int K(x_1^{(1)}|x_1^{(0)})P(y_2|x_{1:2}^{(1)})P(x_2^{(1)}|x_1^{(1)})I(x_{1:2}^{(1)})dx_{1:2}^{(1)}\right)dx_1^{(0)} \tag{A.8}$$

$$=\int\left(\int P(x_1^{(0)}|y_1)K(x_1^{(1)}|x_1^{(0)})dx_1^{(0)}\right)P(y_1)P(y_2|x_{1:2}^{(1)})P(x_2^{(1)}|x_1^{(1)})I(x_{1:2}^{(1)})dx_{1:2}^{(1)} \tag{A.9}$$

$$=\int P(x_1^{(1)}|y_1)P(y_1)P(y_2|x_{1:2}^{(1)})P(x_2^{(1)}|x_1^{(1)})I(x_{1:2}^{(1)})dx_{1:2}^{(1)} \tag{A.10}$$

$$=\int P(x_{1:2}^{(1)}, y_{1:2})I(x_{1:2}^{(1)})dx_{1:2}^{(1)} \tag{A.11}$$

$$\vdots \tag{A.12}$$

$$=\int P(x_{1:t}^{(t-1)}, y_{1:t})dx_{1:t}^{(t-1)} \tag{A.13}$$

$$=P(y_{1:t}) \tag{A.14}$$

Where we use,

1. The previous result and the definition of $\hat{P}(y_1)$
2. The definition of π_1^p
3. Canceling out $\sum_l w(x_1^{(0),l})$
4. Integral over all intermediate variables
5. Definition of $w(x_1^{(0)})$
6. Cancelling out $q(x_1^{(0)})$ and using $P(y_1|x_1^{(0)})P(x_1^{(0)}) = P(x_1^{(0)}, y_1)$
7. The definition of $I(x_1^0) = \int K(x_1^{(1)}|x_1^{(0)})P(y_2|x_{1:2}^{(1)})P(x_2^{(1)}|x_1^{(1)})I(x_2^{(1)})dx_{1:2}^{(1)}$ and $P(x_1^{(0)}, y_1) = P(x_1^{(0)}|y_1)P(y_1)$
8. Regrouping terms
9. Detailed balance for K: $K(x_1^{(1)}|x_1^{(0)})P(x_1^{(0)}|y_1) = K(x_1^{(0)}|x_1^{(1)})P(x_1^{(1)}|y_1)$
10. $P(x_{1:2}^{(1)}, y_{1:2}) = P(x_1^{(1)}|y_1)P(y_1)P(y_2|x_{1:2}^{(1)})P(x_2^{(1)}|x_1^{(1)})$
11. repeating the same logic
12. Integrating out $x_{1:t}^{(t-1)}$

□

Thus our goal is proven. Finally, it is important to note that nothing special in the above proof relies on K targeting $P(x_{1:t}|y_{1:t})$ starting at time 1. A quick look affirms that K may only target $P(x_{t-L:T}|y_{1:T}, x_{t-L-1})$ and the theorem will still hold. This allows one to design K such that only the final portion of particle $x_{1:t}^{(t-1),p}$ need be stored in memory at any given time.

A.2 Particle MCMC Data Association targets the true posterior

In Andrieu et al. [2010], it is shown that we may replace $P(y_{1:T}|\theta)$ with its approximation in the Metropolis-Hastings acceptance ratio and still maintain target distribution $P(x_{1:T}, \theta|y_{1:T})$. We

reuse the notation from the previous section with the exception of θ which denotes the time-invariant parameters of the model. In the sequel, we show that we may use a Resample-Move Particle Filter and maintain the same result. Suppose we use Algorithm 14 to generate samples from the outer Particle MCMC loop (distinct from the inner MCMC loop used within the Resample-Move Particle Filter). We aim to prove the following,

Theorem 4. *Particle MCMC with Resample Move is a normal Metropolis Hastings algorithm over all variables generated by the Resample Move Particle Filter with a target distribution marginalizing out to* $P(\theta, x_{1:T}|y_{1:T})$.

Algorithm 14 Single Iteration of Particle MCMC

Input: $\hat{Z}_i = \hat{P}(y_{1:t}|\theta_i)$ and $(x_{1:T}, \theta)_i$

Output: $\hat{Z}_{i+1} = \hat{P}(y_{1:t}|\theta_{i+1})$ and $(x_{1:T}, \theta)_{i+1}$ such that $(x_{1:T}, \theta)_1, (x_{1:T}, \theta)_2, \ldots$ is distributed approximately according to $P(x_{1:T}, \theta|y_{1:T})$

1: Sample $\theta' \sim q(\theta'|\theta)$

2: Run a Resample-Move Particle Filter to calculate $\hat{Z}' = \hat{P}(y_{1:T}|\theta)$, then sample 1 index K according to the final weights generated, π_T^p

3: Set $x'_{1:T} = x_{1:T}^K$

4: Set $(\hat{Z}, x_{1:T}, \theta)_{i+1} = (\hat{Z}', x'_{1:T}, \theta')$ with probability $\min(1, \alpha)$ where $\alpha = \frac{\hat{P}(y_{1:T}|\theta')P(\theta')q(\theta_i|\theta')}{\hat{P}(y_{1:T}|\theta_i)P(\theta)q(\theta'|\theta_i)}$, else $(\hat{Z}, x_{1:T}, \theta)_i$

Proof. Before we begin, let us define the following:

We will prove a (notationally) simpler case where θ is assumed known and fixed, and we only wish to target $P(x_{1:T}|y_{1:T})$. While this isn't of much interest on its own, its intuition and proof are almost identical to the full Particle MCMC case and will serve as a basis for discussion.

Algorithm 15 Particle Independent Metropolis Hastings with Resample-Move

Input: Current MCMC state $(x_{1:T}, \hat{Z})_i$

Output: New MCMC state $(x_{1:T}, \hat{Z})_{i+1}$ such that $(x_{1:T}, \hat{Z})_1, (x_{1:T}, \hat{Z})_2, \ldots$ approximate sampled from $P(x_{1:T}|y_{1:T})$

1: Run a Resample-Move Particle Filter to calculate $\hat{Z}' = \hat{P}(y_{1:t})$, then sample $K \sim \mathcal{M}(\vec{\pi}_T)$

2: Set $x'_{1:T} = x_{1:T}^K$

3: Calculate acceptance ratio $\alpha = \min\left(\frac{\hat{Z}'}{\hat{Z}_i}, 1\right)$

4: Set $(x_{1:T}, \hat{Z})_{i+1} = (x'_{1:T}, \hat{Z}')$ with probability α else keep $(x_{1:T}, \hat{Z})_i$ otherwise

Consider Algorithm 15. We will show that it is a normal MCMC chain with stationary distribution,

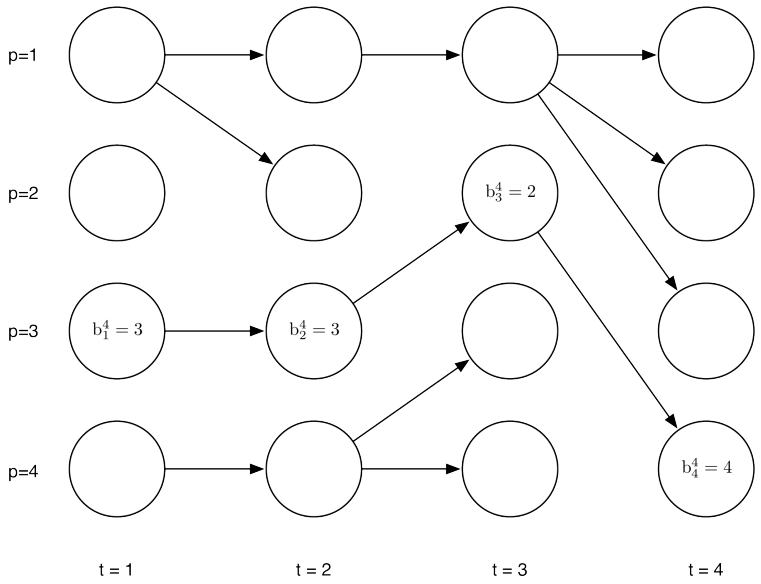


Figure A.2: The history of a single particle in a Particle Filter. Arrows denote parents of particles from the previous time step.

$L(x_{1:t}^{(t-1)}\|x_{1:t}^{(t)})$	The 'reverse' of kernel K defined in the previous section. Defined to be, $$L(x_{1:t}^{(t-1)}\|x_{1:t}^{(t)}) = K(x_{1:t}^{(t)}\|x_{1:t}^{(t-1)})\frac{P(x_{1:t}^{(t-1)}\|y_{1:t})}{P(x_{1:t}^{(t)}\|y_{1:t})} \quad \text{(A.15)}$$
b_t^p	Suppose we traced the 'history' of indices of particle p at time T through the particle. Then b_t^p is the index of that particle at time t. Refer to Figure A.2.
$\mathcal{M}(a_t^p\|\vec{\pi}_t)$	The probability of drawing index a_t^p from a Multinomial distribution with $P(a_t^p = j) = \pi_t^j$
ψ	The likelihood of the whole particle filtering process, including propagating, resampling, and moving. Given by, $$\psi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}) = \left(\prod_{p=1}^{K} q(x_1^{(0),p})\right) \prod_{t=2}^{T}\prod_{p=1}^{N} \underbrace{\mathcal{M}(a_{t-1}^p\|\vec{\pi}_{t-1})}_{\text{resampling}} \times \underbrace{K(x_{1:t-1}^{(t-1),p}\|x_{1:t-1}^{(t-2),a_{t-1}^p})}_{\text{moving}} \underbrace{q(x_t^{(t-1),p}\|x_{1:t-1}^{(t-1),p})}_{\text{propagating}}$$
$\vec{x}_{1:t}^{(t-1)}, \vec{a}_t, \vec{\pi}_t$	The vector of all particles, resample indices, and normalized weights at time t. See Table A.1.

Table A.2: Notation used in Particle MCMC proof

$$\Pi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}, K) \tag{A.16}$$

$$= \underbrace{\left(\frac{P(x_{1:T}^{(T-1),K}|y_{1:T})}{N^T}\right)}_{\text{actual goal}} \left(\frac{\overbrace{\psi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1})}^{\text{particle filter}} \overbrace{\prod_{t=1}^{T-1} L(x_{1:t}^{(t-1),b_{t-1}^K}|x_{1:t}^{(t),b_t^K})}^{\text{doesn't affect actual goal}}}{\underbrace{q(x_1^{(0),b_1^K}) \prod_{t=2}^{T} \pi_{t-1}^{b_{t-1}^K} K(x_{1:t-1}^{(t-1),b_t^K}|x_{1:t-1}^{(t-2),b_{t-1}^K}) q(x_t^{(t-1),b_t^K}|x_{1:t-1}^{(t-1),b_t^K})}_{\text{particle filter history for } x_{1:T}^{(T-1),K}}} \right) \tag{A.17}$$

using the following proposal distribution,

$$q(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}, K) = \underbrace{\psi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1})}_{\text{particle filter}} \underbrace{\pi_T^K}_{\text{choosing K}} \tag{A.18}$$

The most important thing to notice is that **the right hand side of the target distribution is independent of the full history of** $x_{1:T}^{(T-1),K}$, and can thus be integrated away leaving only the left hand side. Algorithmically, this means simply ignoring their value during MCMC. This is the key reason the algorithm works.

To make things notationally simpler, let's relabel the following

$$L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) = \prod_{t=1}^{T-1} L(x_{1:t}^{(t-1),b_{t-1}^K}|x_{1:t}^{(t),b_t^K}) \tag{A.19}$$

$$q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) = q(x_1^{(0),b_1^K}) \prod_{t=2}^{T} K(x_{1:t-1}^{(t-1),b_t^K}|x_{1:t-1}^{(t-2),b_{t-1}^K}) q(x_t^{(t-1),b_t^K}|x_{1:t-1}^{(t-1),b_t^K}) \tag{A.20}$$

$$Z = P(y_{1:T}) \tag{A.21}$$

$$\hat{Z} = \hat{P}(y_{1:T}) = \prod_{t=1}^{T} \hat{P}(y_t|y_{1:t-1}) = \prod_{t=1}^{T} \left(\frac{1}{N} \sum_{p=1}^{N} w(x_{1:t}^{(t-1),p}) \right) \tag{A.22}$$

Let's consider the acceptance ratio $\frac{\hat{Z}'}{\hat{Z}_i}$. We will show that,

$$\frac{\hat{Z}}{Z} = \frac{\Pi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}, K)}{q(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}, K)} \tag{A.23}$$

Assuming this is indeed the case, we see that

$$\frac{\hat{Z}}{\hat{Z}_i} = \frac{\hat{Z}/Z}{\hat{Z}_i/Z} = \frac{\Pi'(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}, K) q_i(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}, K)}{\Pi_i(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}, K) q'(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}, K)} \tag{A.24}$$

Here, Π' denotes (for lack of better notation) the density of the target distribution with respect to all random variables sampled in calculating $\hat{Z}'$, q' the corresponding proposal, and similarly for Π_i and q_i. If this holds true, we do indeed have an MCMC chain with the desired target distribution and an independent proposal distribution.

$$\frac{\Pi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}, K)}{q(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}, K)} \tag{A.1}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{N^T} \right) \left(\frac{\psi(\vec{x}_1^{(0)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}) L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K})}{q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \prod_{t=1}^{T-1} \pi_t^{b_t^K}} \right) \tag{A.2}$$

$$\times \frac{1}{\psi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}) \pi_T^{b_T^K}} \tag{A.3}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{N^T} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K})}{q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \prod_{t=1}^{T} \pi_t^{b_t^K}} \right) \tag{A.4}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{N^T} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K})}{q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \prod_{t=1}^{T} \frac{w(x_{1:t}^{(t-1),b_t^K})}{\sum_l w(x_{1:t}^{(t-1),l})}} \right) \tag{A.5}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{1} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \left(\frac{1}{N^T} \prod_{t=1}^{T} \sum_l w(x_{1:t}^{(t-1),l}) \right)}{q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \prod_{t=1}^{T} w(x_{1:t}^{(t-1),b_t^K})} \right) \tag{A.6}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{1} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \hat{Z}}{q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \prod_{t=1}^{T} w(x_{1:t}^{(t-1),b_t^K})} \right) \tag{A.7}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{1} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \hat{Z}}{q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \prod_{t=1}^{T} \frac{P(x_{1:t}^{(t-1),b_t^K}, y_{1:t})}{P(x_{1:t-1}^{(t-1),b_t^K}, y_{1:t-1}) q(x_t^{(t-1),b_t^K} | x_{1:t-1}^{(t-1),b_t^K})}} \right) \tag{A.8}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K} | y_{1:T})}{1} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K}) \hat{Z}}{\left(\prod_{t=1}^{T-1} K(x_{1:t}^{(t),b_{t+1}^K} | x_{1:t}^{(t-1),b_t^K}) \right) \left(\prod_{t=1}^{T} \frac{P(x_{1:t}^{(t-1),b_t^K}, y_{1:t})}{P(x_{1:t-1}^{(t-1),b_t^K}, y_{1:t-1})} \right)} \right) \tag{A.9}$$

$$= \left(\frac{P(x_{1:T}^{(t-1),K}|y_{1:T})}{P(x_{1:T}^{(T-1),b_T^K}, y_{1:T})} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K})\hat{Z}}{\left(\prod_{t=1}^{T-1} K(x_{1:t}^{(t),b_{t+1}^K}|x_{1:t}^{(t-1),b_t^K})\right)\left(\prod_{t=1}^{T-1} \frac{P(x_{1:t}^{(t-1),b_t^K}, y_{1:t})}{P(x_{1:t}^{(t),b_{t+1}^K}, y_{1:t})}\right)} \right) \tag{A.10}$$

$$= \left(\frac{\hat{Z}}{Z} \right) \left(\frac{L(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K})}{\prod_{t=1}^{T-1} K(x_{1:t}^{(t),b_{t+1}^K}|x_{1:t}^{(t-1),b_t^K}) \frac{P(x_{1:t}^{(t-1),b_t^K}, y_{1:t})}{P(x_{1:t}^{(t),b_{t+1}^K}, y_{1:t})}} \right) \tag{A.11}$$

$$= \left(\frac{\hat{Z}}{Z} \right) \left(\frac{\prod_{t=1}^{T-1} L(x_{1:t}^{(t-1),b_t^K}|x_{1:t}^{(t),b_t^K})}{\prod_{t=1}^{T-1} L(x_{1:t}^{(t-1),b_t^K}|x_{1:t}^{(t),b_t^K})} \right) \tag{A.12}$$

$$= \frac{\hat{Z}}{Z} \tag{A.13}$$

where we have used

1. The definition of $\Pi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{T-1}, K)$ and $q(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1}, K)$
2. Canceling out $\psi(\vec{x}_1^{(0)}, \vec{x}_{1:2}^{(1)}, \ldots, \vec{x}_{1:T}^{(T-1)}, \vec{a}_1, \ldots \vec{a}_{t-1})$ and pushing $\pi_T^{b_T^K}$ into $\prod_{t=1}^{T-1} \pi_t^{b_t^K}$.
3. The definition of $\pi_t^{b_t^K}$
4. Reorganizing $\sum_l w(x_{1:t}^{(t-1),l}$ and $\frac{1}{N^T}$
5. The definition of $\hat{Z}$
6. The definition of $w(x_{1:t}^{(t-1),b_t^K})$
7. The definition of $q(x_1^{(0),b_1^K}, \ldots, x_{1:T}^{(T-1),b_T^K})$ and cancelling out all partcile filter proposal distributions $q(x_t^{(t-1),b_t^K}|x_{1:t-1}^{(t-1),b_t^K})$
8. Reorganizing $\prod_{t=1}^{T} \frac{P(x_{1:t}^{(t-1),b_t^K}, y_{1:t})}{P(x_{1:t-1}^{(t-1),b_t^K}, y_{1:t-1})}$
9. The definition of Z and merging the two $\Pi_{t=1}^{T-1}$
10. The definition of $L(x_{1:t}^{(t-1),b_t^K}|x_{1:t}^{(t),b_{t+1}^K})$

Thus we have established that Algorithm 15 is a proper MCMC algorithm with independent proposals and a target distribution that marginalizes down to $P(x_{1:T}|y_{1:T})$.

Let us now consider our original goal of correctness for Algorithm 7. Looking at $\frac{\hat{Z}'}{Z_i}$ in terms of Π', Π_i, q', q_i we see that the only change would be a multiplication with $\frac{P(\theta')q(\theta_i|\theta')}{P(\theta_i)q(\theta'|\theta_i)}$ and a dependence on θ' (θ_i respectively) when sampling from ψ. As the acceptance ratio is changed to $\frac{\hat{Z}'P(\theta')q(\theta_i|\theta')}{\hat{Z}_iP(\theta_i)q(\theta'|\theta_i)}$, we need only add a notational dependence of θ to the above proof to achieve the exact same result.

□

Bibliography

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Dave Andrzejewski and Shuchi Chawla. CS880: Approximation Algorithms Notes, 2007.

B. Benfold and I. Reid. Stable Multi-Target Tracking in Real-Time Surveillance Video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3457–3464. IEEE, 2011.

K. Bernardin and R. Stiefelhagen. Evaluating Multiple Object Tracking Performance: the CLEAR MOT Metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

S.S. Blackman. *Multiple-Target Tracking with Radar Applications*. Artech House, 1986.

S.S. Blackman. Multiple Hypothesis Tracking for Multiple Target Tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, 2004.

M.D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust Tracking-by-Detection Using a Detector Confidence Particle Filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1515–1522. IEEE, 2009.

T. Dean and K. Kanazawa. A Model for Reasoning about Persistence and Causation. *Computational Intelligence*, 5(2):142–150, 1989.

P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer Verlag, 2004.

P. Diaconis. The Markov Chain Monte Carlo Revolution. *Bull. Amer. Math. Soc.(NS)*, 46(2): 179–205, 2009.

Randal Douc. Comparison of Resampling Schemes for Particle Filtering. In *In 4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 64–69, 2005.

A. Doucet and A.M. Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. *Handbook of Nonlinear Filtering*, pages 656–704, 2009.

P. Fearnhead. Markov Chain Monte Carlo, Sufficient Statistics, and Particle Filters. *Journal of Computational and Graphical Statistics*, 11(4):848–862, 2002.

T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar Tracking of Multiple Targets using Joint Probabilistic Data Association. *Oceanic Engineering, IEEE Journal of*, 8(3):173–184, 1983.

D. Fox. Adapting the Sample Size in Particle Filters through KLD-Sampling. *The International Journal of Robotics Research*, 22(12):985, 2003.

W.R. Gilks and C. Berzuini. Following a Moving Target - Monte Carlo Inference for Dynamic Bayesian Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63 (1):127–146, 2001.

N.D. Goodman, V.K. Mansinghka, D. Roy, K. Bonawitz, and J.B. Tenenbaum. Church: a Language for Generative Models. In *Uncertainty in Artificial Intelligence*, volume 22, page 23, 2008.

P.J. Green. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4):711–732, 1995.

R. Hess and A. Fern. Discriminatively Trained Particle Filters for Complex Multi-Object Tracking. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 240–247. IEEE, 2009.

R. Karlsson and F. Gustafsson. Monte Carlo Data Association for Multiple Target Tracking. In *Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE*, volume 1, pages 13–1. IET, 2001.

Z. Khan, T. Balch, and F. Dellaert. An MCMC-based Particle Filter for Tracking Multiple Interacting Targets. *Computer Vision-ECCV 2004*, pages 279–290, 2004.

Mike Klaas, Nando de Freitas, and Arnaud Doucet. Toward Practical N^2 Monte Carlo: the Marginal Particle Filter. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 308–315, Arlington, Virginia, 2005. AUAI Press.

C. Kreucher, K. Kastella, and A.O. Hero III. Tracking Multiple Targets using a Particle Filter Representation of the Joint Multitarget Probability Density. In *Proc. of SPIE Vol*, volume 5204, page 259, 2004.

R.P.S. Mahler. Multitarget Bayes Filtering via First-Order Multitarget Moments. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1152–1178, 2003.

B. Marthi, H. Pasula, S. Russell, and Y. Peres. Decayed MCMC Filtering. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 319–326. Morgan Kaufmann Publishers Inc., 2002.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21:1087, 1953.

B.C. Milch. *Probabilistic Models with Unknown Objects*. PhD thesis, University of California, Berkeley, 2006.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the National Conference on Artificial Intelligence*, pages 593–598. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

K. Nummiaro, E. Koller-Meier, and L. Van Gool. An Adaptive Color-Based Particle Filter. *Image and Vision Computing*, 21(1):99–110, 2003.

S. Oh, S. Russell, and S. Sastry. Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 735–742. IEEE, 2004.

H. Pasula, S. Russell, M. Ostland, and Y. Ritov. Tracking Many Objects with Many Sensors. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1160–1171, 1999.

M. Pitt, R. Silva, P. Giordani, and R. Kohn. Auxiliary Particle Filtering within Adaptive Metropolis-Hastings Sampling. *Arxiv preprint arXiv:1006.1914*, 2010.

Michael K Pitt and Neil Shephard. Filtering via Simulation: Auxiliary Particle Filters. *Journal of the American Statistical Association*, 94(446):590, 1999.

S. Särkkä, A. Vehtari, and J. Lampinen. Rao-Blackwellized Particle Filter for Multiple Target Tracking. *Information Fusion*, 8(1):2–15, 2007.

B.N. Vo, S. Singh, and A. Doucet. Sequential Monte Carlo Methods for Multitarget Filtering with Random Finite Sets. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4):1224–1245, 2005.

Anh-Tuyet Vu, Ba-Ngu Vo, and R. Evans. Particle Markov Chain Monte Carlo for Bayesian Multi-Target Tracking. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1 –8, July 2011.

E.A. Wan and R. Van Der Merwe. The Unscented Kalman filter for Nonlinear Estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000 (AS-SPCC)*, pages 153–158. IEEE, 2000.

Nick Whiteley. Monte Carlo Methods: Lecture 3: Importance Sampling. `http://www.maths.bris.ac.uk/~manpw/teaching/folien3.pdf`, 2011.

C. Yang, R. Duraiswami, and L. Davis. Fast Multiple Object Tracking via a Hierarchical Particle Filter. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 212–219. IEEE, 2005.

A. Yilmaz, O. Javed, and M. Shah. Object Tracking: a Survey. *ACM Computing Surveys (CSUR)*, 38(4):13, 2006.

Index