
TU Delft
THALES

Jump Markov Nonlinear System Identification in Multi-Sensor Target Tracking

A Novel Approach for Multiple Model Joint Tracking and Behavior Classification

By

Carlos Eduardo Richa

in partial fulfilment of the requirements for the degree of

Master of Science
in Electrical Engineering

at the Delft University of Technology,
to be defended publicly on Friday August 24th, 2018 at 10:30 AM.

Daily Supervisors (Thales):

Dr. ir. Rienk Bakker
Dr. ir. Martin Podt

Academic Supervisor:

Dr. ir. Hans Driessen

Thesis committee:

Prof. DSc. A. Yarovoy	TU Delft
Dr. ir. Hans Driessen	TU Delft
Dr. ir. Martin Podt	Thales Nederland BV
Dr. David Tax	TU Delft

TU Delft THALES

THALES

Approval Internship report/Thesis of:

Eduardo Richa

Title: Multiple model behavioral classification based on parameter estimation

Educational institution: TU Delft

Internship/Graduation period: 21-8-2017 to 31-5-2018

Location/Department: System engineering

Thales Supervisor: Martin Podt, Rienk Bakker

This report (both the paper and electronic version) has been read and commented on by the supervisor of Thales Netherlands B.V. In doing so, the supervisor has reviewed the contents and considering their sensitivity, also information included therein such as floor plans, technical specifications, commercial confidential information and organizational charts that contain names. Based on this, the supervisor has decided the following:

- This report is **publicly available (Open).** Any defence may take place publicly and the report may be included in public libraries and/or published in knowledge bases.

Approved: Martin Podt

(Thales Supervisor)

Approved:

(Educational institution)

Hengelo, 29-05-2018

(city/date)

(copy security)

In memory of Armando Quirós, Fritz Schilling, and Fernando Lugo
for their impact on my life and the world

C.E.R

Acknowledgements

I would like to begin by expressing my gratitude for all the help and support I received throughout my time at Thales. To my three supervisors, Hans Driessen, Martin Podt, and Rienk Bakker. Hans, thank you for the opportunity you gave me to work at a defense company, it had always been a dream of mine since I was young. I appreciate your continuous advice and support both academically and personally throughout this thesis. Martin, having you as a supervisor at Thales was a delightful experience. Thanks for all the coffee breaks to brainstorm and bounce ideas around. Rienk, thanks for the repetitive tours around the facilities and all the coffee breaks as well. I won't forget the most important lesson you taught me: take small steps when tackling large problems. The roll the three of you had on me as mentors will without a doubt have a lasting impact both on my life and my career. I was fortunate to have learned from three world-class engineers.

Next, I would like to thank my parents Eduardo and Lucila for their relentless support throughout my academic career that lasted nearly a decade. Words cannot express how blessed I have been. Dad, thanks for showing me how to build computers before the ripe old age of 6 and how to read math books to teach myself at 13 years old. You prepared me far before I arrived in Holland to do research on my own. Mom, I can sum up the most important thing you taught me since I was a baby that got me through all these challenges—resilience—I owe that one to you. To my brother Andres, thanks for putting up with me for all these years. I won't forget all the coffee rants about random stuff and a bright future we had all those years we lived together at Bull Creek in Austin.

After three incredible years in The Netherlands I managed to meet a lot of fantastic new people, whom without, this would have not been the same experience. Levar, that year on Wagenstraat was definitely one for the books, and you bet that there is more to come. Kiran, thanks for being an awesome trench mate through this signals and systems track. That was terrific preparation for the work to follow. Harriet, Sophie, Hannah, and Alex, I will miss seeing you guys. We had some pretty good times, and I hope we will continue to do so every time I come to Europe. Jaap, Esther, Bori, David, and Jesse, that trip to Ibiza was one of the highlights of my time here. There is indeed no replacement for sunshine, wine, great food, and good company. To Daniel and all my friends in the mastermind group, you will be missed. Thanks for all the good times and for helping me grow as a person.

Lastly, I would like to thank Maria, for all her love and support throughout my thesis. Words cannot express my gratitude. I won't forget all the times you harassed me from across the world to make sure I was eating and sleeping. I'm glad you visited my a couple of times while I was here and were part of my experience in Holland. You truly are my best friend.

Contents

List of Figures

List of Symbols

$\mathbb{R}^n$	The n-dimensional Euclidian space
$\mathbb{R}_+$	The non negative real numbers
$\mathbb{N}$	The natural numbers
$\mathbb{S}_+^n$	The set of $n_x \times n_x$ symmetric positive semi-definite matrices
$\subseteq$	Subset
$\uplus$	Disjoint Union
$\mathbb{E}[\cdot]$	Expectation operator
$\mathbb{E}[\cdot \mid \cdot]$	Conditional expectation operator
∇_θ	Gradient with respect to a parameter θ
$\Lambda(\cdot)$	Closed-form maximizer
$p_\theta(r_{t+1}\|r_t)$	Transition density for the discrete (modes) Markov process
$p_\theta(x_{t+1}\|x_t, r_{t+1})$	State transition density
$p_\theta(y_t\|x_t, r_{t+1})$	Measurement likelihood function
$\xrightarrow{p}$	Convergence in probability
$\xrightarrow{d}$	Convergence in distribution
$\mathcal{D}_{\mathrm{KL}}(\cdot \,\|\, \cdot)$	Kullback-Liebler divergence
$L_\theta(\cdot)$	Log-likelihood function
$var(\cdot)$	Variance operator
$cov(\cdot,\cdot)$	Covariance operator
$\mathcal{X}$	The state space
$\mathcal{Y}$	The measurement space
$\mathcal{Y}_0$	The joint measurement space
ξ_0	The joint sensor state space
$\mathcal{S}$	The set of modes
$\mathcal{R}(\cdot)$	The Bayes' risk function
$\mathcal{C}$	Cost function
Θ	The parameter Space
Φ	The class space
$\mathcal{N}(\mu, \Sigma)$	Multivariate Gaussian distribution with mean μ and covariance matrix Σ
$\mathcal{U}(a, b)$	Uniform distribution on the interval $[a, b]$
$\delta(\cdot)$	The Dirac delta function
$\hat{N}_{eff}$	Estimated number of effective Particles
N_{thresh}	Threshold for resampling
N_p	Number of Particles
Ω	The sample Space
$\mathcal{F}$	The event space (a σ-algebra)
$\mathbb{P}$	A probability measure
$\mathcal{O}(\cdot)$	Computational complexity (big O notation)
w_t^i	The i^{th} particle weight at time t
$w_{t\|N}^i$	The i^{th} particle weight at time t for the marginal smoothed density over N samples

$w_t^{(i,k)}$	The i^{th} particle weight, from the k^{th} class, at time t
x_t	The state variable at time t
x_t^i	The i^{th} particle, at time t
$x_t^{(i,k)}$	The i^{th} state particle, from the k^{th} class, at time t
$\hat{x}_{\varphi_k \mid t}$	The combined state (point) estimate at time t for the k^{th} class.
r_t	The mode at time t
r_t^i	The i^{th} mode particle at time t
$\hat{r}_{\varphi_k \mid t}$	The combined mode (point) estimate at time t for the k^{th} class.
$r_t^{(i,k)}$	The i^{th} mode particle, from the k^{th} class, at time t
y_t	The measurement at time t
$y_{0:N}$	The set of all measurements up to time N
z_t	The augmented state variable
φ_k	The k^{th} class
$s(\varphi_k)$	The number of modes in the k^{th} class.
$\overset{j}{y_t}$	Measurement from the j^{th} sensor
$\overset{j}{\mathcal{Y}_0}$	Measurement space for the j^{th} sensor
$\overset{j}{x^*}$	Sensor state for the j^{th} sensor
$\overset{j}{\xi_0}$	Sensor state space for the j^{th} sensor
$\overset{j}{h_\theta}$	Measurement function for the j^{th} sensor

Abbreviations

ASR	Airport surveillance radar
CLT	Central limit theorem
CRLB	Cramér-Rao lower bound
CRW	Correlated random walks
CT	Coordinated turn
CV	Constant velocity
DLM	Dynamic linear model
EAP	Expected a posteriori
EM	Expectation maximization
FFBSi	Forward filtering backward simulation
FFBSm	Forward filtering backward smoothing
GPB	Generalized pseudo Bayesian
HMM	Hidden Markov model
IMM	Interacting multiple model
IR	Infrared
IRST	Infrared search and track
JMLS	Jump Markov linear system
JMNLS	Jump Markov nonlinear system
JMS	Jump Markov system
JTC	Joint tracking and classification
KL	Kullback-Liebler
MAP	Maximum a posteriori
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
ML	Maximum likelihood
MLE	Maximum likelihood estimator
MM	Multiple model
MM–FFBSm	Multiple model forward filter backwards smoother
MTT	Multi-target tracking
PaRIS	Particle based, rapid incremental smoother
PDF	Probability distribution function
PSD	Power spectral density
RBPF	Rao-Blackwellized particle filter
RCS	Radar cross section
RW	Random walk
SIS	Sequential importance sampling
SLLN	Strong law of large numbers
SMC	Sequential Monte Carlo
SSR	Secondary surveillance radar
STT	Single target tracking
SCFG	Stochastic context-free grammars
SVM	Support vector machines
TBD	Track-before-detect

TPM	Transition probability matrix
UAV	Unmanned aerial vehicle

1 Introduction

1.1 Motivation

Data processing is a central and key concept to modern remote sensing systems. Beginning with the introduction of the Kalman filter in the early 1960s, the possibilities to track moving targets using radar, sonar, infrared (IR) or optical sensors, became even more promising, and since then a growing amount of research has been devoted to the topic. This thesis focuses on how to design an algorithm, by augmenting a tracking system with a classifier capable of recognizing the dynamic behavior of moving targets using data from multiple sensors. Much work has been done in classifying targets by object type [1] [2] [3] [4] in remote sensing applications, but little research exists on the classification of dynamic behavior in the context of target tracking using multi-sensor data. The detection of unusual behavior plays a crucial role in the prevention of illegal and harmful activities such as smuggling, piracy, arms trading, human trafficking and illegal immigration [5]. Also for military applications, it is useful to detect anomalous behavior to provide an alert for potential threats, especially with the more recent widespread use of drones for terrorist activities [6]. In order to provide a solution for these emerging needs, in this work, we present a novel method for target behavior classification by analyzing trajectories using data gathered from multiple sensors.

The primary goal of tracking is to make statistical inferences about the state of one or more unknown objects, such as their speed and position. The former is referred to as single-target tracking (STT) while the latter is referred to as multi-target tracking (MTT). The most basic function of any tracking algorithm is called *measurement-to-track data association*, and as the name implies, it involves assigning a measurement to an existing track or creating a new one when necessary. This track, in turn, is assigned to a single detected source (or group of sources) while differentiating it from other targets and reducing unwanted background noise (such as clutter) and false targets. Creating the proper statistical models and algorithms to make accurate predictions of future states and classification characteristics can then be carried out for the individual tracks [7]. These types of data processing problems are not only limited to remote sensing applications. In finance and economics for example, “tracking” and predicting future prices and characteristics bonds, interest rates, commodities, currencies, or options are similar problems in nature. Here we may be interested in other quantities such as the derivative prices, volatility, asset risk, market regimes, or exchange rates. The asset log-returns above the risk-free rate, or other econometric variables such as a nation's output (GDP) for example, can be seen as the data or "measurements" [8] [9] [10].

Once a track has been assigned to an object, there are multiple ways to classify the target's characteristics. Typically, objects are classified into sets that represent what type of object they are, such as an airliner, boat, fighter plane, UAV, or a bird [11]. Before measurement data can be processed, a sensor system usually follows a signal processing chain composed of a detection and discrimination phase where some information sequences are accepted or rejected in order to minimize the probability of false alarms. The standard hierarchy of a

signal and data processing chain is illustrated in Figure 1.1. Different authors use various definitions of classification, recognition, and identification depending on the application (e.g., see [1] for a discussion of NATO definitions). In this work, classification and recognition will be used interchangeably. Identification in the context of the work done here is used to describe the process of estimating system parameters.

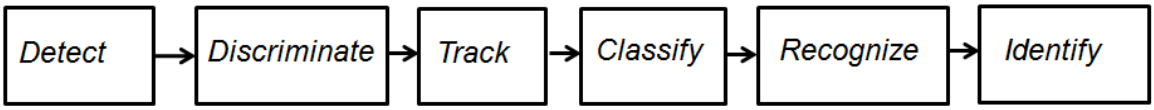


Fig. 1.1. Hierarchy of signal and data processing chain

An important note is that there are systems such as *track-before-detect* (TBD), which are a class of algorithms designed to operate in low SNR (signal-to-noise ratio) environments. Under these conditions, the detector may discard valuable information and the system could be made more robust by tracking an object before declaring it as a target [12]. In this situation, the chain in figure 1.1 is not accurate.

The acquisition of data can come from a single (just one sensor) or a mixture of sensors such as an array of radars. Due to the increased availability of computing power available today and advances in statistics and machine learning over the past half a century, processing data from multiple sensors, earning the name *sensor or information fusion*, has also become an area of growing interest. The need for multiple heterogeneous sensors comes from the fact that single sensors generally can only provide limited or partial information to make accurate inferences. In order to make full use of all these collected data, *multi-sensor management* techniques are becoming increasingly important, as the increased agility of sensors and increasing amounts of data are usually more than what a human operator is capable of processing. Essentially, multi-sensor management deals with the process of coordinating data from multiple sensors to improve performance and perception [13].

A perfect example is the Lockheed f-35, which is arguably the most advanced fighter jet at the date of this writing, in terms of sensor fusion technology. It was designed to process all the information received from all the sensors on the aircraft and to display the information in an easy to read manner, giving the pilot a clear picture of his threats and surrounding environment with minimal effort for interpretation. In light of this growing demand, it is of interest to develop solutions for distributed sensor fusion systems where individual sensors could have different characteristics, and where sensors could be added or replaced with ease. There is definite economic value in designing algorithms that are robust against these changes in system configuration without costly consequences. Modularity, in other words, is a critical requirement that will be addressed in this thesis.

1.2 Problem Statement and Thesis Objective

Maneuvering targets rarely undergo motions that can be accurately captured by a single model, and therefore to accurately model dynamic motion, it should be assumed that targets

can traverse several dynamic behavior modes. A multiple (dynamic) model-based approach is proposed here which requires little training data, and can easily adapt to the addition or replacement of sensors. Multiple model (MM) algorithms allow for targets to "switch" or "jump" between different modes to more accurately capture their more complicated dynamics. This sort of dynamic behavior usually cannot be described accurately by a single maneuver model—especially if the target is highly maneuverable. It will be assumed that each (multiple) model contains its own set of individual unknown parameters including transition probabilities that govern the mode switching, and the random process noise which represents the uncertainty in the dynamic motion model for the case of target tracking. Typically, these parameters are tuned manually, which is not easily done by a human with high accuracy. In this work, it is proposed that these parameters be estimated, using a Bayesian framework with a multi-sensor data configuration.

Since the true state of the system described above is unknown, data is gathered through observations from multiple sensors, and thus, a measurement or observation model is also needed to make a probabilistic relation between the data and the true state. The dynamic and measurement models together form a *state space model*. A well-known class of these types of multiple model systems is referred to in the literature by many different titles such as jump Markov systems (JMS), hybrid systems or regime switching models. This class of models, in the context of Bayesian state estimation and tracking, is widely explored in many fields and applications such as target tracking, biological time series [14], ecology [15], finance and econometrics [16], and audio signal processing [17].

With this understanding, the problem addressed in this work can now be stated more formally. The question to be answered is: *how to develop a context-free (i.e., sensor indifferent) method to robustly classify a selected set of anomalous trajectories and present those results to a human operator?* In this context, robustness is aimed at correctly classifying trajectories with a low false alarm rate. For example, to reduce the error of a human radar operator, it is desirable to be able to distinguish between a fishing boat, an attack pattern such as a highly maneuverable and weaving aircraft, or to classify different behavior of drones which are being used for a vast number of purposes, including terrorism. Since MM algorithms are widely used in tracking, they are a natural and convenient choice. Now that a problem has been clearly defined, the following are the three primary objectives of this thesis:

1. The first objective of this thesis is to estimate the parameters that govern the dynamics of object behavior described by multiple model (hybrid) systems. These parameters will be learned from multi-sensor data.

2. The second objective will be used to jointly track and classify dynamic object behavior based on trajectory analysis using the these trained hybrid models. The idea behind this is that the estimation of the transition probabilities, which describe the tendency for the object to switch to a different mode, can capture the information needed for classification by distinguishing trajectories. This will be done by fitting future data to a number of trained models (in parallel) and then using a Bayes classifier.

3. The third objective is to address the need for modularity. Another advantage of using a hybrid model is that they are very suitable for information fusion. As will be shown in later chapters, the addition or removal of sensors requires one to merely make small

adjustments in the sequential processing of measurements—thus reducing or eliminating the need for retraining.

1.3 Related work

Here we cover existing work in the realm of dynamic behavior classification. An overview of current methods in the literature at the time of this writing, and their shortcomings (for our purposes) is given here. The literature presented here is by no means exhaustive but is meant merely to give the reader an idea of the need and interest for the application of behavior classification and anomaly detection in various tracking scenarios and a vast range of other applications.

To begin, there is a large body of research in the area of machine vision for behavior recognition (see [18][19][20]). Recognition of moving vehicle trajectories is one application that is receiving much attention [21]. These methods are not appropriate for our application since they use large amounts of image data and are not context-free solutions regarding sensor modularity. A distinction must be made that we are tracking objects using remote sensors such as radars, sonar, infrared search and track (IRST), and optical cameras which typically give only bearing information about the target. Data-driven techniques aimed at analyzing trajectories using artificial neural networks (ANN) [22] and deep neural networks (DNN) [23] also exist. So-called data-driven "black box" methods like neural network approaches were also considered as a potential candidate for modeling and classifying dynamic behavior. In order to train neural networks, which have a large number of parameters, deep learning could be applied, but this has four main drawbacks. The first foreseeable issue is that this approach typically requires enormous amounts of data which is not always available, and typically implies very long training times. second issue would be that if the data is gathered from multiple sensors, then the system would have to be retrained if sensors are added or replaced, thus not meeting the requirement of modularity mentioned earlier. The third drawback to mention is that prior knowledge which is typically available cannot be easily incorporated. Lastly, the solutions corresponding to the training data can be quite variable [24].

Given the discussion mentioned above about data-driven techniques, it would seem like a more natural choice for the application in this thesis to use a model-based approach since they are already widely used in tracking systems. To avoid confusion, the term "model-based approaches" implies that explicit models and expert knowledge are used to carefully design the models, as opposed to black box modeling. For airport surveillance radar (ASR), it is vital to classify tracks to distinguish aircraft from non-aircraft tracks, such as weather or biological tracks. Furthermore, for air traffic safety it is crucial to classify different targets such as unmanned aerial vehicles (UAV), helicopters, or aircraft that do not have Secondary Surveillance Radar (SSR) because it has been purposely disabled or a transmitter has failed. The authors in [25] recognize the limitations of neural network techniques due to the massive amounts of training data required, and aim to address this problem using a single source model and support vector machines (SVM) to discriminate between aircraft and non-aircraft targets by analyzing trajectories, radar cross section (RCS), and velocities. This method does not address the use of multiple sensors—nor does it say anything about the behavior of the classified object types— the latter of which can be beneficial in predicting potential safety hazards.

Hidden Markov models (HMM), initially introduced by Leonard E. Baum, Ted Petrie, and their colleagues in the late 1960's to early 1970's in a series of papers [26][27][28] at the Institute for Defense Analyses, have become a popular choice to model stochastic systems. An HMM can be seen as a specific case of JMS where the unknown states are discrete. Baum was also the co-inventor the Baum-Welch algorithm, which is a method of computing ML estimates of the parameters that govern the HMM. The use of discrete hidden Markov models (DHMM) for trajectory classification is not new. Beginning in the 1980's DHMMs became widely used in speech [29] and handwriting [30] recognition and became a core modeling technique in genomic sequencing [31], the most famous appearance being in the Human Genome Project [32]. More recently DHMMs have been used in seismology in the early 2000's and have also been shown to outperform many methods, especially in low SNR scenarios, for earthquake detection and classification due to their ability explicitly model time dependence [19]. In the context of radar tracking, DHMMs require a sufficiently dense discretization of the continuous state space, which can make these methods susceptible to the *curse of dimensionality* (i.e., computationally expensive in high dimensions). It also requires the state space to be predefined and therefore makes it difficult to achieve high resolution in areas of vital importance [33].

DHMMs have been used in ecology to estimate behavioral states based on movement paths of using telemetry and GPS data. It is of interest to deduce the influence of landscape features and conditions on animal behavior in different habitats, such as foraging and resting. One proposed method to model such behavior is to use a mixture of random walks (RW) and correlated random walks (CRW) where each mode differs in step length and turning angle, and with unknown transition probabilities between behavioral modes. The authors in [34], which initially proposed this method, use a Markov chain Monte Carlo (MCMC) approach, namely a Gibbs sampler, for inference on data of the movement of elk. A number of behavior modes, composed of a mix of CRW are first defined. They aimed to classify dynamic animal behavior by analyzing their trajectories and movement patterns. Each GPS measurement is composed of a step length and turning angle, which are assumed to be random variables from a Weibull distribution and wrapped Cauchy distribution, respectively, both with unknown parameters. This work was expanded upon in [35], and it was found that classification accuracy depends strongly on the degree of separation between the distributions in each behavior mode, as well as the amount of time spent in each mode for a given data set. One of the significant shortcomings in these papers is that the authors explicitly ignored measurement error and hence assumed the position of the animals to be known exactly. The measurement inaccuracy from the GPS systems claimed to be negligible. This eliminates the need for a measurement model and dramatically simplifies the problems and limits its findings to other applications where these assumptions cannot be made. A significant contribution of this thesis will be to expand upon DHMM methods mentioned, where the entire state space is discrete, to include a hybrid state space composed of a continuous part in conjunction with a discrete mode.

In the context of surveillance, in [36] a method is proposed for using MTIR (moving target indicator radar) using UAVs for behavior recognition and anomaly detection for assisting human operators to recognize potential threats. The authors in this paper also recognized the burden of data-driven approaches, and point out that on top of the requirement for large amounts of data, they suffer from high computational loads which can pose issues for real-time applications. The authors propose a classification method by applying string matching theory, which has had some success in text-processing applications. They then combine this

method with a fuzzy expert rule-based decision-making process to avoid excess false alarms. The dynamics of the system assume a single model for ground traffic vehicles, and therefore would not be feasible for tracking highly maneuverable targets whose behavior is challenging to capture accurately through a single model, and typically requires multiple models.

Perhaps one of the closest works related to this topic would be in [37] where the author proposes the use of a semi-Markov model to classify targets based on their dynamic behavior. No trajectory analysis was done in this work. The approach presented was to classify targets based on their maneuverability, which in this case was defined by the sojourn time distributions that govern the switching between the current mode governing the system dynamics. This is in contrast to a typical HMM where transition distributions all have the same exponentially distributed sojourn time. The joint tracking and classification of the targets and their dynamic behavior is carried out using a robust Rao-Blackwelized particle filter to track the maneuvers while a Kalman filter is used to track the target.

1.4 Structure and Contributions

Here we present a brief overview of the rest of this document and contributions with more detail. Recall, the goal of the work here is to present a novel method for classifying target behavior based on trajectory analysis. Recall, the primary research question in focus is to investigate if it is possible to learn the parameters in hybrid (multiple model) systems from multi-sensor data, and whether this class of models is suitable for jointly tracking and classifying target behavior. Since we are dealing with multiple sensors, this implies that we require using data fusion techniques to process the measurements. This, in turn, requires the use of multiple measurement models, and the configuration can be done so with ease in JMS. For estimation of the unknown state in linear Gaussian Jump systems, the well-known suboptimal Interacting Multiple Model (IMM) algorithm [38] and generalized pseudo Bayesian (GPB) schemes which use a bank of optimal Kalman Filter running in parallel, are a widely adopted method that has shown to perform well [39] [40]. Unfortunately though, for many applications, it is desirable to handle a wide range of nonlinear/non-Gaussian models. Measurement models for example, which are typically nonlinear, must be handled in another way as the Kalman filter is only suitable for linear models. Approximate solutions such as the extended Kalman filters (EKF) or unscented Kalman filters (UKF) do not have these constraints but have been shown to have other limitations such as poor performance in highly nonlinear systems. Particle filters, or sequential Monte Carlo (SMC), methods are an alternative choice that does not suffer from these drawbacks. It was shown in [41] that the particle filter increased performance for nonlinear/non-Gaussian Bayesian tracking over both the EKF and UKF. Due to their ability to deal with these highly nonlinear/non-Gaussian systems, they will be the chosen method for accurate state estimation in this work.

To estimate the system's parameters (system identification) which are needed to form a classifier, a maximum likelihood (ML) approach is a popular choice for many applications and is widely used and understood. More specifically, the expectation maximization (EM) algorithm will be employed for the numerical calculation ML estimates due to many desirable properties such as numerical robustness and a guaranteed convergence to a (local) maximum. It was shown in [42] that and [43] that the EM algorithm in combination with the SMC methods previously mentioned is capable of learning parameters in Jump Markov non-linear Systems (JMNLS), and will be the class of methods used here. Although there are other

methods for nonlinear system identification, we choose this one, due to the fact that is has shown promising results for estimating transition probabilities in systems with switching (jumps).

This thesis is organized as follows, including a description of this introduction for completeness.

- **Chapter 1**: An introduction and motivation behind the problem are provided. A brief literature review is also given.
- **Chapter 2**: The reader will be given some context and preliminaries needed for the work to follow. A brief treatment of Bayesian methods for statistical inference, including ML estimation, which are the core principles of our approach, will be presented. Included in this discussion are sequential Monte Carlo (SMC) methods or particle methods for state space systems which are central to the techniques used later.
- **Chapter 3**: This chapter will present the core novel algorithm design composed of two main parts. In the first part, the system identification techniques will be covered. First, the EM algorithm will be discussed in more detail. This will be done by first extending SMC methods for multiple model systems and explain how to use them to calculate expectations in the EM algorithm. Then the maximization step of the EM algorithm will be discussed. We finish this chapter by discussing the second part of the algorithm: the Joint tracking and classification (JTC) algorithm, for classifying targets behavior (as defined by their trajectory) in real time through trained models. This is essentially classification by model selection using a Bayesian classifier, where a unique set of learned parameters defines each model.
- **Chapter 4**: Here the gap will be bridged between the generic algorithms of the previous chapter and specific kinematic and measurement models that are used in remote sensing applications. Also, one of the main novel contributions of this work will be explained here, namely to incorporate data fusion techniques for parameter learning. The measurement models will, therefore, be extended for multiple sensor scenarios.
- **Chapter 5**: Covers simulations in MATLAB. Basic assumptions will be described, including parameters to be estimated and the configuration of the sensor system. An example of the learning portion of the proposed novel algorithm for system identification is first examined for estimating a set of parameters including the transition probabilities and noise parameters for the individual dynamic models (process noise). Finally, simulations of the JTC algorithm are carried out over multiple trajectories. A discussion of the results will take place within each example.
- **Chapter 6**: A brief summary of the work and contributions provided by this work is given. Finally, recommendations, proposed improvements, and direction for future work is touched on.

2 Preliminaries

2.1 Overview

The purpose of this chapter is to give the reader a brief overview of Bayesian methods for inference in time series models. The methods described here are found in a vast number of fields ranging from machine learning, statistics, financial econometrics to engineering. We begin by introducing the reader to state space models, which were initially proposed by Kalman and Bucy in 1960. Following this and the emersion of the Kalman filter shortly after, linear dynamical systems with Gaussian noise became a popular model for aerospace and control systems applications. These models and state estimation techniques were quickly adopted in many other applications, but their limitations became abundantly clear. They are not suitable for more complicated dynamical systems that are nonlinear in nature. Suboptimal methods such as the extended Kalman filter were developed to tackle the challenge of nonlinearities in system dynamics. Sequential Monte Carlo methods were proposed as early as the 1950's, although the lack of computational power of computers at the time rendered them infeasible. It wasn't until 1993 that these methods resurfaced when Gordon et al. [44] proposed a practical implementation of what is referred to as the bootstrap particle filter to tackle to problem of nonlinear and non-Gaussian Bayesian state estimation.

This chapter is organized as follows. We begin by introducing state space hidden Markov models in section 2.2. Then in section 2.3 maximum likelihood estimation will be treated. The Bayesian framework for statistical inference, which is a core idea of the work to follow, will be presented in section 2.4, along with MAP (maximum a posteriori) estimators. In this section, the reader will see the link between the Bayesian and maximum likelihood estimation. SMC filtering and smoothing methods for state estimation will be discussed in sections 2.5-2.6. Although much effort was put into explaining these building blocks, it is assumed the reader has a basic foundation in probability theory, and many topics will be expected to be known a priori.

2.2 Time Series Analysis

The analysis of data, gathered from observations or measurements at discrete time steps and the statistical properties and relations of these sequences is known as time series analysis. Any discrete-time data or signal such as daily stock prices, blood pressure measurements, brain wave patterns in functional MRI imaging, data samples in digital communication systems, and the amount of rainfall or weather patterns are all examples of time series data [45].

2.2.1 State-Space Model

A state-space model, which can be viewed as a generalization of a *hidden Markov model* (HMM), is one in which the data in question is not known directly. Instead, a transformed version of it in the form of an observation or measurement is known. To avoid confusion, from here on, we refer to a state-space hidden Markov model as a state-space model. Therefore, we require a model which describes the temporal evolution of the state, which is known as the *evolution* or *system model*. Another equation is then needed to model the relationship between the state and the measurements, which we will refer to as the *measurement model*. Both the system and measurement equations are assumed to be in stochastic form. The most general structure of a state space system is defined by two stochastic processes $\{x_t\}_{t\geq 0}$ and $\{y_t\}_{t\geq 0}$, $t \in \{0,1,2,3 \dots\}$ as

$$x_{t+1} = f_\theta(x_t, u_t, v_t) \quad (2.1a)$$
$$y_t = h_\theta(x_t, u_t, e_t) \quad (2.1b)$$

where f_θ and h_θ are known possibly nonlinear and time-varying mapping functions that depend on a set of parameters $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$. The central assumptions here are that the continuous state vector $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is a latent Markov process and is only observed indirectly through noisy measurements $y_t \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ which are conditionally independent. It is important to note that although the state variable and measurements are continuous variables, the systems presented here are their discrete-time equivalents and therefore can be seen as sampled values. We will assume $v_t \sim p_v(\cdot)$ and $e_t \sim p_e(\cdot)$ are mutually independent white noise processes with known probability density functions, which may also be parameterized by the parameter θ. The variable u_t is the exogenous system input and is assumed to be known and can be ignored without loss of generality. Further, we assume that all random variables are defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Here we assume, Ω is the sample space, and let 2^Ω be the power set of Ω. Then, let $\mathcal{F} \subseteq 2^\Omega$ be the σ-algebra containing the event space and $\mathbb{P}$ is an appropriate probability measure.

2.2.2 Jump Markov Systems

In many applications where structural uncertainty occurs, such as navigation, telecommunications, control theory, target tracking, and financial risk management and economics [46]. it is necessary to form more adaptive models. The state can be expanded to be comprised of a continuous and discrete component, and these types of systems are referred to in the literature as jump Markov systems (JMS) or hybrid systems. JMS assume that the dynamic system can abruptly change between different modes according to a Markovian switching scheme. The underlying assumption is that a single model is not sufficient to describe the system dynamics and therefore *multiple model* (MM) descriptions are needed to account for the uncertainty. The discrete component of the state is referred to as the *mode* $r_t \in \mathcal{S} \subseteq \mathbb{N}^{n_r}$, where $\mathcal{S} = \{1, 2, 3, \dots, s\}$ is a finite set. The mode evolves according to an *s*-state (discrete-time) Markov chain, governed by *transition probability matrix* (TPM) $\Pi_\theta = [\pi_{ij}]$, which is time-invariant and therefore can be considered a parameter of the system, defined as

$$\pi_{ij} \triangleq P(r_{t+1} = j \,|r_t = i\,), \quad \forall i,j \in S\,, t \geq 0 \tag{2.2}$$

The augmented state vector can now be defined as $z_t = [x_t^T, r_t]^T$, and it is referred to as a *hybrid process* [39]. A linear jump Markov system (LJMS), is a state space model where both the mode dependent state and measurements evolve according to a mode dependent dynamic linear model (DLM):

$$x_{t+1} = A(r_{t+1})x_t + B(r_{t+1})v_t \tag{2.3a}$$
$$y_t = C(r_t)x_t + D(r_t)e_t \tag{2.3b}$$

where the system matrices $A(\cdot), B(\cdot), C(\cdot), D(\cdot)$ evolve over time according to the finite state Markov chain r_t. These models are also referred to as dynamic linear models with switching [45]. A jump Markov nonlinear system (JMNLS) is similar to (2.1), except now it is mode dependent as well:

$$x_{t+1} = f_\theta(x_t, r_{t+1}, v_t) \tag{2.4a}$$
$$y_t = h_\theta(x_t, r_t, e_t) \tag{2.4b}$$

With a slight abuse of notation, and due to the stochastic nature of the noise components in the above descriptions, we can also express the above random variables in terms of their transition densities $p_\theta(\cdot \mid \cdot)$, where the parameter subscript indicates what is intended, as

$$r_{t+1} \sim p_\theta(r_{t+1}|r_t) \tag{2.5a}$$
$$x_{t+1} \sim p_\theta(x_{t+1}|x_t, r_t) \tag{2.5b}$$
$$y_t \sim p_\theta(y_t|x_t, r_t) \tag{2.5c}$$

One last remark the reader should keep in mind is that although the state variable is continuous by nature, here we will focus on the discrete-time formulation of these problems. The transition or difference equations will be used to make inferences on the state of the system by processing the measurements at discrete time steps.

2.3 Maximum Likelihood Estimation

The maximum likelihood (ML) approach is one of the central principles in modern statistics, and although it is different on the surface, it is ultimately linked to Bayesian estimation. Therefore we present it here, and the similarity between the two will become clear in the next section. Suppose we have N conditionally independent measurements (observed data) $y_{0:N} = [y_0, \dots, y_N]$, then the *likelihood function* $\mathcal{L}(\theta; y_{0:N}) = p_\theta(y_{0:N})$ is the joint density probability distribution function (PDF) of *all the measurements* as a function of the *unknown parameters* θ. The goal of ML estimation is then to maximize the likelihood function with respect to the unknown parameters. In many applications, especially those where the data comes from a family of exponential distributions, it is more convenient to work with the *log-likelihood function* $L_\theta(y_{1:N})$, which is defined by merely taking the natural logarithm of the likelihood function

$$L_\theta(y_{0:N}) \triangleq \log \mathcal{L}(\theta; y_{0:N}) = \log\, p_\theta(y_{0:N}) = \prod_{t=1}^{N} p_\theta(y_t|y_{0:t-1})\ . \tag{2.6}$$

The maximum likelihood estimator (MLE) can then be written as

$$\hat{\theta} = \underset{\theta}{argmax}\ L_\theta(y_{0:N}) = \underset{\theta}{argmin}\ -L_\theta(y_{0:N}) \tag{2.7}$$

where the last part follows from the fact that that the likelihood function is monotonic.

The MLE is a popular choice for practical applications due to a number of desirable large sample properties. Typically it is regarded as an approximately optimal estimator, or approximately the minimum variance unbiased (MVU) estimator [47]. The most important properties of the MLE will be presented here without proof: (see [48][49][50]):

1. *Consistency*: An estimator $\hat{\theta}$ is said to be (weakly) consistent if $\hat{\theta} \xrightarrow{p} \theta$, where $\xrightarrow{p}$ denotes convergence in probability.

2. *Asymptotic efficiency:* Even though we cannot guarantee that any estimator can attain the Cramér-Rao Lower Bound (CRLB), which gives a lower bound on the variance of the estimator, for a finite amount of data the MLE will reach it asymptotically as $n \to \infty$. Furthermore, the MLE is unbiased and can be stated as

$$\mathbb{E}\left(\hat{\theta}_{ML}\right) = \theta \tag{2.8}$$
$$\lim_{n\to\infty} var\left(\hat{\theta}_{ML}\right) = CRLB. \tag{2.9}$$

3. *Asymptotic normality:* Using the central limit theorem (CLT) it can be shown that this can be stated more formally as

$$\sqrt{n}\left(\hat{\theta} - \theta\right) \xrightarrow{d} \mathcal{N}\left(0, I^{-1}(\theta)\right) \tag{2.10}$$

Where $I(\theta)$ is the *Fisher information matrix:*

$$\mathbb{E}_\theta\left[\nabla_\theta L_\theta(y_{0:N}) \nabla_\theta L(y_{0:N})^T\right] \tag{2.11}$$

and $\xrightarrow{d}$ denotes convergence in distribution as $n \to \infty$.

4. *Invariance:* The MLE is preserved by parametrization $\lambda = q(\theta)$ where q is a known function. Under the condition that q is non-invertible, then λ maximizes the modified likelihood:

$$L^*(\lambda|x) \triangleq \sup_{\theta:q(\theta)=\lambda} L(\theta|x). \tag{2.12}$$

5. The MLE for a set of data is equivalent to the MLE of the *sufficient statistics.*

It is important to note that the asymptotic properties (1-3) require certain regularity conditions on the family conditions as smoothness, the existence of the derivatives of L_θ, and non-zero Fisher information (see [47] and [50] for more detail).

2.4 Bayesian Inference

Here the Bayesian approach to statistical inference provided to give the reader the necessary background for understanding the state and parameter estimation in the following chapter. This can be achieved by using previous information about an unknown parameter (in this case the state can be viewed as the unknown) contained in the *prior* distribution and using the received measurement data to update it to the *posterior* distribution by using Bayes' law. The two distributions are related by proportionality as

$$posterior \propto prior \times likelihood. \tag{2.13}$$

It is through this combination of using the observed data and the prior knowledge that Bayesian analysis allows us to model the uncertainty in the outcomes of an underlying process. From (2.13) we can already see a hint that ML estimation is indeed connected to the Bayesian framework, and this will become clear shortly. The discussion herein will be in the context of state space models. Our focus will be on the posterior distribution of the state because all the necessary information to describe the system is contained within it [16]. Our focus will now turn to *recursive* Bayesian estimation. The goal is to sequentially estimate the posterior distribution of the unobserved data $p_\theta(x_t|y_{0:t})$ given all available measurements at time t. The Markov property of (2.1a) and the independence of the observations are key assumptions in the explanations to follow. Assuming we have a set of observations $y_{0:t}$ then by repeated use of Bayes' theorem, the posterior densities can be computed in a two-stage iterative process:

Prediction stage:

$$p_\theta(x_{t+1}|y_{0:t}) = \int p_\theta(x_{t+1}|x_t)p_\theta(x_t|y_{0:t})dx_t \tag{2.14}$$

Update/Correction stage:

$$p_\theta(x_t|y_{0:t}) = \frac{p_\theta(y_t|x_t)p_\theta(x_t|y_{0:t-1})}{p_\theta(y_t|y_{0:t-1})} \tag{2.15}$$

with

$$p_\theta(y_t|y_{0:t-1}) = \int p_\theta(y_t|x_t)p_\theta(x_t|y_{0:t-1})dx_t \tag{2.16}$$

where the normalizing factor (2.15) is known as the *evidence*. An important thing to notice is that (2.15) is consistent with the relation in (2.13). The procedure for arriving at the *prediction equation* (2.14) involves the use of the *Chapman-Kolmogorov equation* and the Markov nature of the state evolution distribution. The calculation of the three densities on the right-hand side of (2.15) is the primary focus in Bayesian inference. All together (2.14-2.16), including the measurement likelihood function $p_\theta(y_t|x_t)$, provide the basis to estimate the posterior state density recursively. The reader is referred to [48] for the details.

Lastly, we discuss the concept of optimality. An estimator can only be optimal in a specific sense [51]. The natural question that arises is how to define or measure optimality? To answer this let's first define the *Bayes risk* as

$$\mathcal{R}(x_t, \hat{x}_t) = \mathbb{E}[\mathcal{C}(\eta)] \tag{2.17}$$

where $\eta = (x_t - \hat{x}_t)$ is the *prediction (estimation) error*, and where $\mathcal{C}$ is referred to as the *cost* or *loss function*. The value of $\hat{x}_t$ which minimizes the Bayes risk is considered the optimal Bayes estimator [52]. Now that this has been established, we name three fundamental Bayesian optimality criterion [53]:

1. *Minimum mean square error (MMSE):* Here we aim to compute the conditional mean

$$\hat{x} = \mathbb{E}[x_t|y_{0:t}] = \int_{\mathcal{X}} x_t p_\theta(x_t|y_{0:t}) dx_t \tag{2.18}$$

   This is the equivalent of defining a quadratic cost function $\mathcal{C} = \eta^2$ and attempt to minimize the Bayes risk:

$$\mathbb{E}[\|x_t - \hat{x}_t\|^2|y_{0:t}] = \int_{\mathcal{X}} \|x_t - \hat{x}_t\|^2 p_\theta(x_t|y_{0:t}) dx_t \tag{2.19}$$

   And hence $\hat{x}$ is also called a *conditional mean estimator* or the expected a posteriori (EAP) estimator.

2. *Maximum a posteriori (MAP):* This estimator does exactly what the name says—it maximizes the posterior density $p_\theta(x_t|y_{0:t})$. Another way to say this is that it finds the largest mode of the posterior, and a major drawback of this approach is that for multi-modal distributions, it can lead to poor estimations. In terms of the Bayes risk, the MAP estimator minimizes a "hit or miss" [54] loss function:

$$\mathcal{C} = 1 - \mathbb{1}_{x_{t:(\|\eta\|)\leq\zeta}}(x_t) \tag{2.20}$$

   Where ζ is a small scalar and $\mathbb{1}_{\mathcal{A}}(\cdot)$ is the indicator function over some set $\mathcal{A}$.

3. *Maximum Likelihood:* We revisit the MLE to show its connection to the Bayesian philosophy. If we define the prior density in the posterior distribution to be a uniform distribution, then the prior plays no role in maximizing the posterior, and therefore, the MLE can be seen as a particular case of the MAP estimator.

Any estimator that minimizes the Bayes risk is referred to as *Bayes-optimal*. It is important to note that the MLE is typically only Bayes-optimal in the case of a uniform prior, which is rarely the case [55].

2.5 Sequential Monte Carlo Methods

Now that we have established the general Bayesian inference scheme, we can move on to methods for sequential Bayesian estimation appropriate for nonlinear non-Gaussian systems, based on *Monte Carlo* (MC) techniques. MC techniques are a class of methods for approximate inference based on numerical sampling method when exact inference is infeasible. We, therefore, turn our attention to so-called *Sequential Monte Carlo* (SMC) approaches; also known as *particle methods*. The aim is to recursively approximate the sequence of posterior probability distributions defined on a sequence of probability spaces. This can be achieved by a combination of the *sequential importance sampling* (SIS) and *sampling importance resampling* (SIR) algorithms, which will be discussed shortly. The result is a sequence of posterior distributions that are represented by a set of particles with associated nonnegative particle weights.

2.5.1 Sequential Importance Sampling

We begin our discussion about SMC by introducing the concept of *importance sampling*. As mentioned earlier, for nonlinear non-Gaussian problems, not always feasible to sample from the posterior distribution. Since it is usually not possible to draw samples directly from the posterior distribution, except in special cases where there exists a random number generator for that distribution (e.g., Gaussian), a workaround is to draw samples from another (known) distribution $q_\theta(x)$ called a *proposal distribution* or *importance density*. Samples drawn from the importance density, for which a well-constructed random number generator exists, can then be used to compute an approximation of the *target distribution* $p_\theta(x)$. The choice of importance density is a crucial design parameter, and should be as close as possible to the target density, and should have the same support. As the name implies, the goal of the sequential importance sampling (SIS) algorithm is to estimate the filtered posterior distributions recursively, through a Bayesian filter, and represent them as a histogram of *point masses* or *particles* x_t^i with associated *importance weights* w_t^i. Therefore, to facilitate the computation of integration in Bayesian estimation, the posterior distribution can be characterized as a *weighted particle system* $\{x_t^i, w_t^i\}_{i=1}^{N_p}$, where the weights for each time step sum to one. With this being said, an approximation of the target posterior density can be expressed as an empirical point-mass distribution

$$p_\theta(x_t|y_{0:t}) \approx \hat{p}_\theta^{N_p}(x_t|y_{0:t}) \triangleq \sum_{i=1}^{N_p} w_t^i \, \delta(x_t - x_t^i) \tag{2.21}$$

where $\delta(\cdot)$ is the Dirac delta function. Now suppose the importance density can be factorized using Bayes' law as

$$q_\theta(x_{0:t}|y_{0:t}) = q(x_t|x_{0:t-1}\, y_{0:t}) q(x_{0:t-1}|y_{0:t-1}) \tag{2.22}$$

$$= q(x_0) \prod\nolimits_{n=1}^{t} q(x_n|x_{0:n-1}, y_{0:n}) \tag{2.23}$$

where we have used the chain rule of probability to arrive at (2.23). If we further assume that the posterior can be factorized as

$$p(x_{0:t}|y_{0:t}) = p(x_{0:t-1}|y_{0:t-1}) \frac{p_\theta(y_t|x_t)p_\theta(x_t|x_{t-1})}{p(y_t|y_{0:t-1})} \tag{2.24}$$

we can now define the weights as (see [53])

$$\begin{aligned} w_t^i &= \frac{p_\theta(x_{0:t}^i|y_{0:t})}{q_\theta(x_{0:t}^i|y_{0:t})} \\ &\propto \frac{p_\theta(y_t|x_t^i)p_\theta(x_t^i|x_{t-1}^i)p_\theta(x_{0:t}^i|y_{0:t-1})}{q_\theta(x_t^i|x_{0:t-1}^i, y_{0:t})q_\theta(x_{0:t-1}^i|y_{0:t-1})} \\ &= w_{t-1}^i \frac{p_\theta(y_t|x_t^i)p_\theta(x_t^i|x_{t-1}^i)}{q_\theta(x_t^i|x_{0:t-1}^i, y_{0:t})} \end{aligned} \tag{2.25}$$

If we now impose the Markov and measurement independence assumptions on the importance density such that $q_\theta(x_t^i|x_{0:t-1}^i, y_{0:t})$ =$q_\theta(x_t^i|x_{t-1}^i, y_t)$ then the update recursion (2.25) reduces to

$$w_t^i \propto w_{t-1}^i \frac{p_\theta(y_t|x_t)p_\theta(x_t|x_{t-1})}{q_\theta(x_t^i|x_{t-1}^i, y_t)}. \tag{2.26}$$

This will usually be the case in most applications where only the current filtered estimate of the posterior is required. To ensure that (2.21) is a properly defined probability measure the weights must be normalized

$$w_t^i = \frac{w_t^i}{\sum_{j=1}^{N_p} w_t^j} \tag{2.27}$$

As $N_s \to \infty$ the approximation (2.22) approaches the true posterior density. One time step of the SIS filter is presented in here in Algorithm 1.

Algorithm 1: SIS Particle filter

INPUTS: $\left\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N_p}, y_t$

1. For $i = 1, \dots, N_p$ draw the samples from the proposal distribution as $x_t^{(i)} \sim q(x_t|x_{t-1}^{(i)}, y_t)$
2. For $i = 1, \dots, N_p$ calculate the importance weights $w_t^{(i)}$ according to (2.26)
3. For $i = 1, \dots, N_p$ normalize the importance weights according to (2.27)
4. Set $t \mapsto t + 1$ go back to step 2.

OUTPUTS: $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N_p}$

Importance sampling provides a framework for producing approximations of expectations with respect to a distribution $p_\theta(x)$. Suppose we seek to produce the expectation of a nonlinear measurable function $f: \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ of a random variable x, which can be written as

$$\mathbb{E}[f(x)] = \int f(x)p_\theta(x)dx = \int f(x)\frac{p_\theta(x)}{q_\theta(x)}q_\theta(x)dx. \tag{2.28}$$

To form an approximation of (2.28), we begin drawing N_s i.i.d samples $x^i \sim q_\theta(\cdot)$ and using a *Monte Carlo estimator* [56] while making the appropriate substitutions we arrive at the following expression:

$$\mathbb{E}[f(x)] \approx \frac{1}{N_s}\sum_{i=1}^{N_s} w(x^i)f(x^i) \tag{2.29}$$

where $w(x^i)$ are the normalized importance weights (2.27). This process is also referred to as *Monte Carlo integration*. These techniques will be used extensively in the next chapter, as they are a critical part of the computing the expectations in E-step of the EM algorithm.

2.5.2 Sampling Importance Resampling

In practice, the SIS filter suffers from some severe drawbacks, which leads us to discuss an improved version known as the sampling importance resampling (SIR) algorithm. The first issue faced when implementing an SIS filter is known as the *degeneracy problem* [33][48]. This phenomenon that occurs after a few iterations in which all the weight tends to a single particle, while the rest of them have negligible influence. Furthermore, the unconditional variance of the importance weights is guaranteed to increase with time, and therefore it is impossible to avoid this problem [57]. One way to measure degeneracy is known as the *effective sample size* [58] defined as

$$N_{eff} = \frac{N_p}{1 + var(w_t^{*i})} \leq N_p \tag{2.30}$$

where $w_t^{*i} = p_\theta(x_t^i|y_{0:t})/q_\theta(x_t^i|x_{t-1}^i, y_t)$ is called the "true weight" [33]. Since this cannot be evaluated, a common approach is to use an approximation using normalized weights:

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p}(w_t^i)^2}. \tag{2.31}$$

A simple approach to combat the degeneracy problem is to increase the number of particles one uses, but this is inefficient as it leads to a significant increase in computation complexity. One of the most common ways to deal with the degeneracy problem is by *resampling*. The basic concept is to remove particles with negligible weight and concentrate more particles in areas of higher importance, to more accurately capture regions of high probability in the true posterior. One of the most basic ways to achieve this is by replacing each particle x_t^j with a new particle $\tilde{x}_t^i$ according to $P(x_t^j = \tilde{x}_t^i) = w_t^i$ if the effective sample size falls below some threshold N_{thresh}. The SIS filter which resamples each time the effective sample size falls

below N_{thresh} is known as the *sampling importance resampling* (SIR) particle filter. A conventional algorithm to perform resampling task is known as *systematic* resampling and is presented below in Algorithm 2. Although there are many other methods for resampling (see [48] [53] [58]) each with different properties, we choose the systemic approach due to its ease of implementation.

Algorithm 2: Systemic Resampling

INPUTS: $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N_p}$

1. Initialize CDF: $c_1 = 0$.
2. For $i = 2, \dots, N_p$ construct the CDF as: $c_i = c_{i-1} + w_t^i$
3. Begin by drawing a starting point $u_1 \sim \mathbb{U}\,[0\,, N_p^{-1}]$
4. For $j = 1, \dots, N_p$ let $u_j = u_1 + N_p^{-1}(j-1)$ and do:
    - WHILE $u_j > c_i$
        - $i = i + 1$
    - END WHILE
    - Set $\tilde{x}_t^{(i)} = x_t^{(i)}$
    - Set $\widetilde{w}_t^{(i)} = N_s^{-1}$

OUTPUTS: $\left\{\tilde{x}_t^{(i)}, \widetilde{w}_t^{(i)}\right\}_{i=1}^{N_p}$

Resampling helps alleviate the degeneracy problem but also produces other undesirable by-products. The most important one is known as *path degeneracy* [59] or *sample impoverishment*, which is especially an issue when there is little to no process noise and is a consequence of the strong law of large number (SLLN). The problem arises because after resampling the sequences of particles are no longer statistically independent [60]. Other issues can be the limited possibility of parallelization and a reduction in diversity, the latter of which can lead to inaccurate estimates of statistics [33]. There are a variety of methods to handle these issues (see [53]) but are beyond the scope of this document and do not concern us for the work to be done here.

2.5.3 Bootstrap Particle Filters

The *bootstrap particle filter*, first introduced by Gordon, Salmond, and Smith [44] in 1993, is widely recognized as being the first practical implementation of the particle methods previously mentioned. The algorithm is also referred to by different names in the literature such as the *condensation* algorithm or the *survival of the fittest* algorithm [61][48]. As mentioned before one of the most crucial design parameters in the design of a particle filter is the choice of importance density $q_\theta\left(x_t^i \middle| x_{t-1}^i, y_t\right)$. One standard approach is to choose a density that maximizes the effective sample size by minimizing the variance. The second approach is more complicated but is shown in [62] to be

$$q_\theta\left(x_t^i|x_{t-1}^i,y_t\right)_{opt} = \frac{p_\theta(y_t|x_t)p_\theta\left(x_t|x_{t-1}^i\right)}{p_\theta\left(y_t|x_{t-1}^i\right)}. \tag{2.32}$$

Unfortunately, this choice faces a lot of practical problems such as the need to generate samples which may not always be so straightforward, and the computation of an integral for the normalizing constant $q_\theta\left(y_t|x_{t-1}^i\right) = \int p_\theta(y_t|x_t)p_\theta(x_t|x_{t-1}^i)dx_t$, which generally has no closed-form solution except in certain special cases. A good compromise is to choose the suboptimal state transition density $p_\theta(x_t|x_{t-1})$ as the importance density, which leads to the bootstrap filter. This is one of the most widely used particle filters due to its simplicity and ease of implementation. One iteration of the SIR and bootstrap filter is presented below in Algorithm 3.

Algorithm 3: SIR/Bootstrap Particle Filter

INPUTS: $\left\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N_p}, y_t$

1. Initialize particles according to prior density, $\left\{x_0^{(i)}\right\}_{i=1}^{N_p} \sim p_\theta(x_0)$ and set $t = 1$.
2. For $i = 1, \dots, N_p$ predict particles forward by drawing M i.i.d. particles sampled as $x_t^{(i)} \sim q_\theta(x_t|x_t^{(i)}, y_t)$ for optimal SIR filter or use $q_\theta(x_t|x_{t-1})$ for bootstrap filter
3. Evaluate the importance weights $\left\{w_t^{(i)}\right\}_{i=1}^{N_p}$ as,

$$w_t^{(i)} = p_\theta(y_t|x_t^{(i)})$$

4. For $i = 1, \dots, N_p$, normalize the importance weighs:

$$w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N_p} w_t^{(j)}}$$

5. Compute $\widehat{N_{eff}}$ according to (insert equation number)
6. If $\widehat{N_{eff}} \leq N_{THR}$ resample particles and reset weights according to ***Algorithm 2***.
7. Set $t \mapsto t + 1$ go back to step 2.

OUTPUTS: $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N_p}$

2.6 Smoothing

The process of using past, present, and future data to make an inference on the state at time t is referred to as smoothing. In the Bayesian context, the goal is to recursively estimate the sequence (smoothed) marginal posterior densities $p_\theta(x_t|y_{0:t+\tau})$, $\forall\, x_t \in [x_0, \dots, x_N]$, $\tau \geq 1$ and $t < N$. There are many different flavors of smoothing, and the former is referred to as (off-line) fixed interval smoothing, but there are other types as well. In fixed-point smoothing,

one is interested in estimating the posterior $p_\theta(x_t|y_{0:N})$ for fixed t as N increases. In fixed-lag smoothing the goal is to estimate $p_\theta(x_t|y_{0:t+\Delta t})$ for fixed Δt as t increases, and is used for recursive (on-line) implementations. Smoothing can be thought of as two filters: a forward filter that makes a state estimate based on previous data and a backward filter that uses only future data. Under white noise assumptions for both the process and measurements, the errors from the two filters are uncorrelated [51]. In the work to follow we will focus on *fixed-interval* smoothing. More specifically, for a particle smoother, we aim to obtain an approximation of the smoothed marginal posterior distributions as

$$p_\theta(x_t|y_{0:N}) \approx \hat{p}_\theta^{N_p}(x_t|y_{0:N}) \triangleq \sum_{i=1}^{N_p} w_{t|N}^i \, \delta(x_t - x_t^i) \tag{2.33}$$

Where $N > t$ and $w_{t|N}^i$ are the smoothed weights at time t taking into account all the data up till time $t = N$. To be explicit, we aim to arrive at a weighted particle system $\{x_t^i, w_{t|N}^i\}_{i=1}^{N_p}$. As originally proposed in [62] and [63], the *forward filter/backward smoother* (FFBSm), where the "m" stands for marginal, is an approximate method to compute the marginal smoothing distributions $\hat{p}_\theta^{N_p}(x_t|y_{0:N})$. In this procedure, the particles from the forward filtering pass remain unchanged, weights of the filtered particle system are updated in a backward pass. The weights can be updated according to the following relation:

$$w_{t|N}^i = \sum_{j=1}^{N_p} w_{t+1|N}^j \frac{w_t^i \, p_\theta(x_{t+1}^j|x_t^i)}{\sum_{l=1}^{N_p} w_t^l \, p_\theta(x_{t+1}^j|x_t^i)}. \tag{2.34}$$

The algorithm is presented below in Algorithm 4, and the reader is referred to [53] for more details.

Algorithm 4: Fixed-interval smoother (FFBSm)

INPUTS: $\{x_t^i, w_t^i\}_{i=1}^{N_p}$ (Forward filter particle systems for $t = 1, \dots, N$)

1. Initialize at time $t = N$,
   - For $i = 1, \dots, N_p$ set $w_{N|N}^i = w_t^i$
2. For $t = N - 1, \dots, 1$
   - For $i = 1, \dots, N_p$ evaluate importance weights according to (2.34)

OUTPUTS: $\{w_{t|N}^i\}_{i=1}^{N_p}$ for $t = 1, \dots, N$

Smoothing is critical for the system identification techniques that will be discussed in the next chapter. It is important to note that there exist various algorithms for implementation, all with different computational complexity and ease of implementation (see [58] [59] [60]).

Some of the more sophisticated algorithms include the *and forward filtering backward simulation* (FFBSi) and the *particle-based, rapid incremental smoother* (PaRIS) to name a few.

3 Algorithm Design

This chapter contains the general mathematical formulation of the behavior classification algorithm, while the next two chapters will contain the explicit system modeling and simulations with remote sensing applications. We take an *unsupervised learning* approach using the state-space representation to model the dynamics of the system. Recall, the state space approach is a way of modeling the relationship between the unknown state and mode, to the measurements. The parameters that govern the dynamics, including the TPM, must be estimated. The estimation of the parameters can be seen as a "training" step. We investigate the system identification problem for JMNLS using the EM algorithm for ML estimation of the unknown parameters. To compute the expectations, we must expand on the particle filtering methods of the previous chapter to handle hybrid (multiple model) systems. We then extend existing methods for particle smoothing and tailor them to form a multiple model particle smoother, which is crucial for the E-step of the EM algorithm. We follow by discussing the subsequent the M-step. The last section culminates with the final proposed algorithm with the aim of being able to both track targets and classify their behavior using a Bayes' classifier that will be implemented via a multiple model particle filter.

3.1 System Identification overview

Nonlinear system identification with Markovian switching is known to be a challenging problem (see [60] and references therein). To be explicit, by system identification, we mean the learning of the unknown parameters that define the system. Since the method of estimating these system parameters, in the context of state-space models, is one of the core novel contributions of this work, the topic deserves a separate discussion of prior work. Here a concise yet thorough overview of system identification techniques for general state space models is presented.

State estimation in state-space models received much attention after the invention of the Kalman filter. Although this was a great stride forward, a new issue arrived due to the fact that in order to use a Kalman filter the system must be linear and driven by a Gaussian white noise process. Another issue is that the system parameters are assumed to be known, which is rare in practice [42]. An algorithm for smoothing and forecasting in DLM with unknown parameters using a Kalman smoother in combination with the EM algorithm was first proposed by Shumway and Stoffer in 1982 [67], where they apply their proposed method to economic data. In the early 2000's, a great deal of effort began towards system identification in more general settings such as nonlinear systems with Markovian switching (jumps). For the linear case, in [68] a method was proposed to recursively MMSE estimates of transition probabilities in an IMM setting. Shortly after, an alternative Bayesian approach for estimating these transition probabilities for linear systems was proposed in [69], and later a ML approaching using the EM algorithm was presented in [70].

The rise of stochastic sampling methods in systems with jumps such as particle methods (see, e.g. [71][72]), led to more general system identification methods that could estimate all system parameters, as well an handle nonlinear systems. In the first decade or so of the 2000's, there was a large body of work devoted to using SMC methods in conjunction with the EM algorithm to estimate parameters of nonlinear systems without jumps in both an online and offline fashion [73] [74] [75] [76] [60]. There was also some work proposing the use of particle filters to compute Jacobians needed for a direct ML approach via gradient ascent methods [77].

A framework for the estimation of all parameters of a JMNLS using SMC methods first appeared in 2012 in [43], where an online EM-based fashion using fixed-lag particle smoothers to compute approximate expectations in the E-step. This formulation was explicitly for systems where both the measurement and process noise are mixtures of members from the exponential family. Two years later in 2014, the authors in [42] expanded upon the work in [60] for systems without jumps in order to generalize the work done in [43] to more generic JMNLS, which could be done in an offline fashion as well. Shortly after, the authors in [78] expanded on their previous work in [43], to present a recursive approach using Rao-Blackwellized particle filters (RBPF) for joint-state and parameter estimation, resulting in a more efficient parameter inference scheme. Recently this work was further expanded upon in [79] where a general method for system identification of JMNLS was proposed using the PaRIS smoothing algorithm which reduced the computational complexity of and variance of the parameter estimates. Lastly, in 2017, a recursive gradient-based approach for ML estimation was proposed as an alternative to the EM approaches in [80].

Another noteworthy mention is that much work was also done in other fields such as mathematical finance. In 2006 for example, the authors in [80] estimated the model parameters, including the transition probabilities, for a Markov switching stochastic volatility model. The reader is referred to [16],[81] and the references therein for an overview for existing work done in parallel in the areas of finance and economics.

Lastly, we have only covered model-approaches here. There are also black box deep learning methods for nonlinear system identification, but as mentioned previously, these approaches typically involve estimating a huge amount of parameters, and hence require a lot of data (see, e.g., [82]). For these reasons, we will not consider these methods further.

3.1.1 Mathematical Formulation

Let's assume that the target dynamics evolve according to a JMNLS (2.4a-2.4b) which is shown again here for convenience:

$$x_{t+1} = f(x_t, r_{t+1}, v_t, \theta) \quad (3.1a)$$
$$y_t = h(x_t, r_t, e_t, \theta) \quad (3.1b)$$

We again assume $t \in \mathbb{N} = \{0,1,2,\dots\}$, $v_t \sim p_v(\cdot)$ and $e_t \sim p_e(\cdot)$. The initial state x_0 has a known prior $p_\theta(x_0|r_0)$ and the initial mode probabilities $\pi_i \sim p_\theta(r_0)$ are available.

The TPM $\Pi_\theta = [\pi_{ij}]$ is defined as in (2.2) and the transition densities using (2.5a-2.5c). Our first challenge is to estimate a set of deterministic and time-independent parameters contained in a set $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$, which includes the transition probabilities π_{ij}. The goal is to do this

based on all available independent measurements $y_{0:N}$. If we denote θ_n as the parameters corresponding to the n^{th} mode can express the unknown parameters as

$$\theta = (\{\theta_n\}_{n=1}^{s}, \{\Pi_\theta\}). \tag{3.2}$$

To learn these parameters, we will incorporate the ML approach for reasons such as the fact that it does not require a prior and the other favorable statistical properties already mentioned in section 2.3. With a clear direction in place, we now explicitly state our system identification problem mathematically as:

$$\hat{\theta}^{ML} = \underset{\theta}{argmax}\ L_\theta(y_{0:N}) \tag{3.3}$$

A typical direct approach to maximizing (3.3) would be to employ a gradient-ascent algorithm. Potential challenges with this approach that one could encounter are that direct knowledge of $p_\theta(y_{0:N})$ is scarce, and when it is available, analytical closed-form solutions typically do not exist, and direct maximization requires the numerical computation of high-dimensional integrals [43]. In [80], the authors propose a gradient-based recursive maximum likelihood (RML) approach and acknowledge other potential drawbacks such as the need for explicitly dealing with parameters constraints such as those of the Markov transition matrix by re-parameterization or using constrained optimization methods. The re-parameterization will cause a change in the gradient which can cause a change in convergence results, and therefore the algorithm must be carefully designed. The EM algorithm is on the other hand, is well known for its numerical stability, ease of implementation, and the ability to often handle parameter constraints explicitly [84]. It also guarantees an increase in likelihood every iteration, although to be fair, its rate of convergence is known to be very slow at best. Two main drawbacks are the lack of a built-in method for computing covariance parameter estimates and potentially slow convergence [85]. For JMNLS, if the noise sequences are members of the exponential family, then very efficient implementations for online estimation make the EM algorithm an attractive choice as well [78]. There is merit in both the EM and gradient-based RML approaches. The former converges linearly, but the latter can reach quadratic convergence rates. The EM algorithm has another significant advantage in that it does not require the calculation of the gradient of the likelihood function. This is one of the main reasons it is the chosen candidate since it simplifies computations.

3.2 Expectation Maximization

The EM algorithm [56][86], is a two-stage iterative optimization process for computing maximum likelihood estimates in stochastic models that have latent (hidden) variables. In our case, the unobserved or hidden data is the augmented state variable comprised of the state and the mode $z_t = [x_t^T, r_t]^T$. The algorithm will produce a sequence of estimates for the unknown parameters $\{\theta_k\}_{k\geq 0}, k \in \mathbb{N}$. One of the properties that makes it a popular choice is its ability to guarantee an increase in (log) likelihood $L_{\theta_k}(y_{0:N}) > L_{\theta_{k-1}}(y_{0:N})$ at every iteration. The key is to instead approach the maximization problem (3.3) from a different angle, and consider the joint log-likelihood of the observations and the hidden data (also referred to as the *complete data log-likelihood*) $L_\theta(y_{0:N}, z_{0:N})$. The critical assumption is that it is (usually) easier to optimize than the (*incomplete*) log-likelihood $L_\theta(y_{0:N})$. In the case of the hybrid system (3.11), the complete data likelihood can be expressed as

$$p_\theta(y_{0:N}, z_{0:N}) = \prod_{t=0}^{N} p_\theta(z_t, y_t|z_{t-1}) \tag{3.4}$$

$$= p_\theta(r_0)p_\theta(x_0|r_0)\prod_{t=1}^{N} p_\theta(z_t, y_t|z_{t-1}) \tag{3.5}$$

where we assume we have a known priors $p_\theta(r_0)$ and $p_\theta(x_0|r_0)$. We define the augmented state transition density $p_\theta(z_t|z_{t-1}) = p_\theta(x_{t+1}|x_t, r_{t+1})p_\theta(r_{t+1}|r_t)$ and subsequently let $p_\theta(z_t, y_t|z_{t-1}) = p_\theta(y_t|z_t)p_\theta(z_t|z_{t-1})$. After substituting the appropriate densities and taking the logarithm of both sides of (3.5) we arrive at the complete data log-likelihood as:

$$L_\theta(y_{0:N}, z_{0:N}) = \log p_\theta(r_0) + \log p_\theta(x_0|r_0) + \sum_{t=0}^{N-1} \log p_\theta(r_{t+1}|r_t) + \sum_{t=0}^{N} \log p_\theta(y_t|x_t, r_t) + \sum_{t=1}^{N-1} \log p_\theta(x_{t+1}|x_t, r_{t+1}). \tag{3.6}$$

Since $z_{0:N}$ are unknown or "hidden," we must form an approximation $\mathcal{Q}(\theta, \theta_k)$ where we assume a true parameter value θ_k. The approximation is then formed by averaging over the unobserved variables by evaluating the conditional mean estimate (MMSE estimator)

$$\mathcal{Q}(\theta, \theta_k) \triangleq \mathbb{E}_{\theta_k}[\, L_\theta(y_{0:N}, z_{0:N})|y_{0:N}) = \int L_\theta(y_{0:N}, z_{0:N})p_{\theta_k}(z_{0:N}|y_{0:N})dz_{0:N} \tag{3.7}$$

Therefore, in the case of the system (3.1a-3.1b), we can form $\mathcal{Q}(\theta, \theta_k)$ by applying the conditional mean operator $\mathbb{E}_{\theta_k}[\,\cdot\,|y_{1:N}\,]$ to both sides of (3.6) we arrive at:

$$\mathcal{Q}(\theta, \theta_k) = \Upsilon_1 + \Upsilon_2 + \Upsilon_3 + \Upsilon_4 + \Upsilon_5 \tag{3.8}$$

where we have broken down the function into:

$$\Upsilon_1 = \sum_{r_0 \in \Omega} \log p_\theta(r_0)\, p_{\theta_k}(r_0) \tag{3.9a}$$

$$\Upsilon_2 = \sum_{r_0 \in \Omega} \int_{\mathcal{X}} \log p_\theta(x_0|r_0)p_{\theta_k}(x_0|r_0, y_{0:N})dx_0 \tag{3.9b}$$

$$\Upsilon_3 = \sum_{t=0}^{N-1} \sum_{r_{t+1} \in \Omega} \sum_{r_t \in \Omega} \int_{\mathcal{X}} \log p_\theta(r_{t+1}|r_t)p_{\theta_k}(x_{t+1}, r_{t+1}, r_t|y_{0:N})dx_{t+1} \tag{3.9c}$$

$$\Upsilon_5 = \sum_{t=0}^{N-1} \sum_{r_{t+1} \in \Omega} \iint_{\mathcal{X}} \log p_\theta(x_{t+1}|r_{t+1}, x_t) p_{\theta_k}(x_{t+1}, x_t, r_{t+1}|y_{0:N}) dx_{t+1} dx_t \tag{3.9d}$$

$$\Upsilon_5 = \sum_{t=0}^{N} \sum_{r_t \in \Omega} \int_{\mathcal{X}} \log p_\theta(y_t|x_t, r_t) p_{\theta_k}(x_t, r_t\,|y_{0:N}) dx_t \tag{3.9e}$$

Looking at (3.9a-3.9e) we can see that the computation of the approximate complete data log-likelihood $\mathcal{Q}(\theta, \theta_k)$ requires the computation of multi-dimensional integrals. As will be shown in section 3.2.3, this typically requires numerical approximations, where MC integration will prove to be a crucial tool.

3.2.1 Computing State Filtered Densities

To accommodate the extension to a JMNLS, we must extend the particle filtering methods of the previous chapter to accommodate for the system dynamics being able to switch between modes. Initially proposed by McGinnity and Irwin [87], the *multiple model bootstrap filter* is an alternative to the IMM algorithm, and it can also deal with nonlinear and non-Gaussian systems. The process aims to represent the mode dependent state posterior densities as

$$p_\theta(x_t, r_t|y_{0:t}) \approx \hat{p}_\theta^{N_p}(x_t, r_t|y_{0:t}) \triangleq \sum_{i=1}^{N_p} w_t^i \, \delta(x_t - x_t^i) \mathbb{1}(r_t = r_t^i) \tag{3.10}$$

The algorithm is presented below, and the reader is referred to [88] for more details.

Algorithm 5: Multiple model bootstrap particle filter

INPUTS: $\{y_{0:t}\}$

1. Initialize particles according to prior density $\{x_0^i, r_0^i\}_{i=1}^{N_p} \sim p_\theta(x_0)$ and set $t = 1$.
2. Predict mode particles $\{r_{t+1}^i\}_{i=1}^{N_p}$ forward based on $\{r_t^i\}_{i=1}^{N_p}$ and Π_θ
3. Draw process noise samples $\{v_t^i\}_{i=1}^{N_p} \sim p_{r_t}(v)$ and predict particles x_{t+1}^iforward using the Markov transition density:

$$x_{t+1} = f_\theta(x_t, r_{t+1}, v_t)$$

4. Evaluate importance weights for the augmented state particles $\{z_{t+1}^i\}_{i=1}^{N_p} = \{x_{t+1}^i, r_{t+1}^i\}_{i=1}^{N_p}$ as

$$w_{t+1}^i = p_\theta(y_{t+1}|\, x_{t+1}^i, r_{t+1}^i)$$

5. For $i = 1, \dots, N_p$, normalize the importance weighs:

$$w_t^i = \frac{w_t^i}{\sum_{j=1}^{N_p} w_t^j}$$

6. Compute $\widehat{N_{eff}}$ according to (2.31)
7. If $\widehat{N_{eff}} \leq N_{THR}$ resample particles and reset weights according to ***Algorithm 2***.
8. Set $t \mapsto t+1$ go back to step 2.

OUTPUTS: $\{z_{0:t}^i, w_{0:t}^i\}_{i=1}^{N_p}$

The use of multiple model particle filtering for target tracking is not new, and a large volume of literature is devoted to the topic (see, e.g. [89] [91][92]).

3.2.2 Computing Smoothed Marginal Densities

By taking a closer look at (3.7) and (3.9a-3.9e), one can see that the calculate the expectations of the complete data log-likelihood with respect to the conditional distribution $p_\theta(z_{0:N}|y_{0:N})$, requires knowledge of the smoothed densities $p_\theta(x_0|r_0, y_{0:N})$, $p_\theta(x_{t+1}, r_{t+1}, r_t|y_{0:N})$, $p_\theta(x_{t+1}, x_t, r_{t+1}|y_{0:N})$, and $p_\theta(x_t, r_t|y_{0:N})$. Furthermore, to combat the particle degeneracy problem that arises when $t \ll N$ in approximating (3.10), where a single particle represents the posterior density, we require similar empirical approximations of the form

$$p_\theta(x_t, r_t|y_{0:N}) \approx \hat{p}_\theta^{N_p}(x_t, r_t|y_{0:N}) \triangleq \sum_{i=1}^{N_p} w_{t|N}^i \, \delta(x_t - x_t^i)\mathbb{1}(r_t = r_t^i) \tag{3.11}$$

This approximation does not suffer from the degeneracy problem needed for accurate parameter estimation [92]. Doucet, Godsill and Andriu [57] first proposed a fixed interval particle smoother; recall from section 2.7, this algorithm, referred to as, the FFSBm algorithm, where again "m" stands for marginal, and can be extended to the multiple model case. In practice, due to its high computational complexity $\mathcal{O}(NN_P^2)$, the FFSBm algorithm is only suitable for situations where the data sets are relatively small. Given that for this thesis, we aim to develop methods for situations where data is limited this is not seen as an obstacle. To be clear, the method for computing the sequence of smoothed marginals $\{\hat{p}_\theta^{N_p}(x_t, r_t|y_{0:N})\}_{t=0}^{N}$ is a design choice, and the general procedure for computing expectations does not change. For applications where the computational load is a limitation, the smoothing technique presented here can be replaced by other alternatives such as a fixed-lag smoother [93]; other alternatives will also be suggested in Chapter 6.

With this in mind, an expression for the smoothed weights $w_{t|N}^i$ in (3.11) can be calculated recursively by extending the work done in [57], and again, the particles will remain the unchanged. A proof of this extension is provided in Appendix C. The smoother will,

therefore, be fed the augmented state particles $\{z_t^i\}_{i=1}^{N_p} \triangleq \{x_t^i, r_t^i\}_{i=1}^{N_p}$ and will update their corresponding weights to form the approximated smoothed state density (3.11) as:

$$w_{t|N}^i = \sum_{j=1}^{N_p} w_{t+1|N}^j \frac{w_t^i \, p_\theta(x_{t+1}^j | x_t^i, r_{t+1}^j) p_\theta(r_{t+1}^j | r_t^i)}{\sum_{l=1}^{N_p} w_t^l \, p_\theta(x_{t+1}^j | x_t^l, r_{t+1}^j) p_\theta(r_{t+1}^j | r_t^l)} \tag{3.12}$$

We will name this algorithm the *multiple model forward filter backward smoother* (MM-FFBSm), where once again "m" stands for marginal.

Algorithm 6: Fixed-interval multiple model particle smoother (MM-FFBSm)

INPUTS: $\{z_t^i, w_t^i\}_{i=1}^{N_p}$ (Forward filter particle systems for $t = 1, \dots, N$).

1. Initialize at time $t = N$,
   - For $i = 1, \dots, N_p$ set $w_{N|N}^i = w_t^i$
2. For $t = N-1, \dots, 1$
   - For $i = 1, \dots, N_p$ update the importance weights according to (3.12)

OUTPUTS: $\{w_{t|N}^i\}_{i=1}^{N_p}$ for $t = 1, \dots, N$

3.2.3 Computing Expectations (E-Step)

We now turn to the computations of the expectations (3.15a-3.15e) which require the computation of integrals with respect to smoothed densities. The reader may now understand the importance of MC estimators introduced in the previous chapter since these integrals generally have no closed-form solutions except when dealing with linear systems [60]. We can express an approximation to (3.8) as

$$Q(\theta, \theta_k) \approx \hat{Q}(\theta, \theta_k) = \widehat{\Upsilon_1} + \widehat{\Upsilon_2} + \widehat{\Upsilon_3} + \widehat{\Upsilon_4} + \widehat{\Upsilon_5} \tag{3.13}$$

with

$$\Upsilon_1 \approx \widehat{\Upsilon_1} = \sum_{i=1}^{N_p} w_{0|N}^i \log p_\theta(r_0^i) \tag{3.14a}$$

$$\Upsilon_2 \approx \widehat{\Upsilon_2} = \sum_{i=1}^{N_p} w_{0|N}^i \log p_\theta(x_0^i | r_0^i) \tag{3.14b}$$

$$\Upsilon_3 \approx \widehat{\Upsilon_3} = \sum_{t=0}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \log p_\theta(r_{t+1}^i | r_t^i) \tag{3.14c}$$

$$\Upsilon_4 \approx \widehat{\Upsilon_4} = \sum_{t=0}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \log p_\theta\left(x_{t+1}^j | x_t^i, r_{t+1}^j\right) \tag{3.14d}$$

$$\Upsilon_5 \approx \widehat{\Upsilon_5} = \sum_{t=0}^{N} \sum_{i=1}^{N_p} w_{t|N}^{i} \log p_\theta\left(y_t | x_t^i, r_t^i\right) \tag{3.14e}$$

where the weights $w_{t|N}^{ij}$ are defined as

$$w_{t|N}^{i} = \frac{w_t^i w_{t|N}^j p_{\theta_k}\left(x_{t+1}^j | r_{t+1}^j, x_t^i\right) p_{\theta_k}\left(r_{t+1}^j | r_t^i\right)}{\sum_l^{N_p} w_t^l p_{\theta_k}\left(x_{t+1}^j | r_{t+1}^j, x_t^l\right) p_{\theta_k}\left(r_{t+1}^j | r_t^l\right)}. \tag{3.15}$$

The derivation of this is a straightforward use of importance sampling and Bayes law (see [42]).

3.2.4 Maximization (M-Step)

To maximize $\hat{\mathcal{Q}}(\theta, \theta_k)$, we can do this via numerical methods, or in some cases via a closed-form maximizer $\Lambda(\cdot)$. Both approaches typically require the calculation of the gradient, which must vanish to find an inflection point that yields a maximizer $\hat{\theta}$. By taking the gradient of (3.13), we can compute the gradient of the particle representations of $\mathcal{Q}(\theta, \theta_k)$ by

$$\nabla_\theta \hat{\mathcal{Q}}(\theta, \theta_k) = \nabla_\theta \widehat{\Upsilon_1} + \nabla_\theta \widehat{\Upsilon_2} + \nabla_\theta \widehat{\Upsilon_3} + \nabla_\theta \widehat{\Upsilon_4} + \nabla_\theta \widehat{\Upsilon_5} \tag{3.16}$$

with

$$\nabla_\theta \widehat{\Upsilon_1} = \sum_{i=1}^{N_p} w_{0|N}^i \nabla_\theta \log p_\theta\left(r_0^i\right) \tag{3.17a}$$

$$\nabla_\theta \widehat{\Upsilon_2} = \sum_{i=1}^{N_p} w_{0|N}^i \nabla_\theta \log p_\theta\left(x_0^i | r_0^i\right) \tag{3.17b}$$

$$\nabla_\theta \widehat{\Upsilon_3} = \sum_{t=0}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \nabla_\theta \log p_\theta\left(r_{t+1}^i | r_t^i\right) \tag{3.17c}$$

$$\nabla_\theta \widehat{\Upsilon_4} = \sum_{t=0}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \nabla_\theta \log p_\theta\left(x_{t+1}^j | x_t^i, r_{t+1}^j\right) \tag{3.17d}$$

$$\nabla_\theta \widehat{\Upsilon_5} = \sum_{t=0}^{N} \sum_{i=1}^{N_p} w_{t|N}^i \nabla_\theta \log p_\theta\left(y_t | x_t^i, r_t^i\right) \tag{3.17e}$$

For many problems, no closed-form maximizer exists. Therefore, as mentioned before, a numerical approximation can be employed for more general models. Following the suggestion in [84], a gradient-based search technique could be employed. See [94] for a treatment of such numerical schemes.

3.3 Learning Phase

In this section, we put together the different components of the system identification approach to be used in this work for JMNLS, or to put it another way, the unsupervised learning phase of the hybrid state HMMs defined by (3.1a-3.1b). A common stopping criterion for the algorithm is $\zeta = \hat{Q}(\theta_{k+1}, \theta_k) - \hat{Q}(\theta_k, \theta_k) \leq \epsilon$ and the algorithm is terminated once this value is below some user-defined tolerance $\epsilon > 0$. Since the focus of this work is classification, the process here is a batch algorithm due to its simplicity. The algorithm can be modified to an online implementation by using fixed-lag smoothers if needed in practice. In Chapter 5, we will use training data (measurements) generated from kinematic motion models to train the system for recognizing trajectories based on their dynamic behavior using multiple sensors.

Algorithm 7: Multiple Model Particle EM Algorithm for Jump Markov Nonlinear System Identification

INPUTS: $\left\{z_{0:N}^i, w_{0:N}^i\right\}_{i=1}^{N_p}, y_{0:N}$

1. Initialize $\{\theta_k | \, L_{\theta_k}(y_{0:N}) < \infty\}$ and set $k = 1$.
2. E-Step:
   - Run Algorithms 5 and 6 to obtain the filtered and smoothed particle representations (3.10) and (3.11)
   - Use these approximate densities to compute (3.7) via (3.9a-3.9e)
3. M-Step:
   - Compute $\theta_{k+1} = \underset{\theta \in \Theta}{argmax}\, \hat{Q}(\theta, \theta_k)$
4. Evaluate the stopping criterion $\zeta = \hat{Q}(\theta_{k+1}, \theta_k) - \hat{Q}(\theta_k, \theta_k)$ for some chosen tolerance $\epsilon > 0$ and do :
   - If $\zeta > \epsilon$
     - Set $k \longmapsto k + 1$ go back to step 2.
   - Else
     - Terminate loop and continue to step 5.
5. Set $\hat{\theta}_{ML} = \theta_{k+1}$

OUTPUTS:$\{\hat{\theta}_{ML}\}$

3.4 Joint Tracking and Classification

We finalize this chapter by formalizing one of the main contributions in this work, namely to build a classification algorithm for targets based on their behavior. This can be inferred from the information of their temporal behavior contained in the estimated transition matrix and process noise parameters based on their trajectories.

This process is referred to in the literature as *joint tracking and classification* (JTC) [95]. The task of classification (or model selection) will be carried out by using a bank of multiple model particle filters in parallel to both track and classify targets, as illustrated in figure 3.1. The outputs of these filters will be used at each time step to calculate the posterior class probabilities. Typically, JTC problems include a class measurement such as RCS, in addition to kinematic measurements, used in classifying object type (e.g., military aircraft or commercial airliner) [96]. This will be ignored as we have no measurement for behavior in our problem formulation since we instead distinguish between trajectories with the use of different estimated transition probabilities. Now instead suppose each target belongs to one of m (behavior) classes $\varphi_k \in \Phi = \{\varphi_1, \dots, \varphi_m\}$. Our goal in this section is to estimate the state $x_t \in \mathcal{X}$ and posterior class probabilities $P(\varphi_k|y_{0:t})$ for each $k \in \{1, \dots, m\} = \mathcal{K}$. We assume, that for each class, an initial prior $P_0(\varphi_k)$, such that $\sum_{\varphi_k \in \Phi} P_0(\varphi_k) = 1$, is available. The parameters for all the classes are denoted by

$$\theta = \left(\{\theta_{\varphi_k}\}_{k=1}^{m}, \{\Pi_{\varphi_k}\}_{k=1}^{m}\right) \subseteq \Theta \tag{3.18}$$

where θ_{φ_k} are the system parameters and $\Pi_{\varphi_k} = [\pi_{ij,k}]$ are the transition probabilities for the k^{th} class. A diagram of the algorithm to be derived is shown in Figure 3.1.

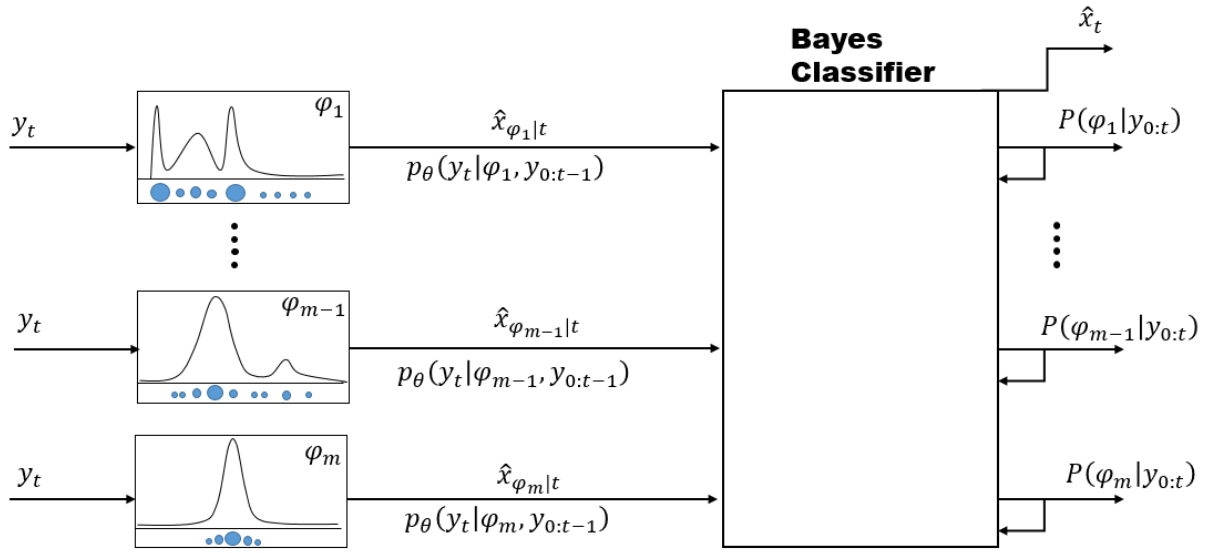


Fig. 3.1. Diagram of proposed JTC system

The number of modes in each class is not restricted to be the same, and we define $s(\varphi_k)$ as the number of modes per class. The likelihood for each class will be expressed more compactly for convenience as

$$\lambda_{\varphi_k}(y_t|x_t) = p_\theta(y_t|\{x_t, \varphi_k\}, y_{0:t-1}) \tag{3.19}$$

Prediction and measurement updates for each class can be calculated in the usual manner using the Bayesian recursions (2.10-2.11):

$$\begin{aligned} p_\theta(\{x_{t+1}, \varphi_k\}|y_{1:t-1}) &= \int_{x_t \in \mathcal{X}} p_\theta(x_{t+1}, \{x_t, \varphi_k\}|y_{0:t}) dx_t \\ &= \int_{x_t \in \mathcal{X}} p_\theta(x_{t+1}|\{x_t, \varphi_k\}, y_{0:t}) p_\theta(\{x_t, \varphi_k\}|y_{0:t}) dx_t \\ &= \int_{x_t \in \mathcal{X}} p_\theta(x_{t+1}|\{x_t, \varphi_k\}) p_\theta(\{x_t, \varphi_k\}|y_{0:t}) dx_t \end{aligned} \tag{3.20}$$

with

$$\begin{aligned} p_\theta(x_{t+1}|\{x_t, \varphi_k\}, y_{0:t}) &= \sum_{j=1}^{s(\varphi_k)} p_\theta(x_{t+1}|x_t, r_{t+1} = j, y_{0:t}) \cdot P(r_{t+1} = j \,|x_t, \varphi_k, y_{0:t}) \\ &= \sum_{j=1}^{s(\varphi_k)} p_\theta(x_{t+1}|x_t, r_{t+1} = j, y_{0:t}) \cdot \sum_{i=1}^{s(\varphi_k)} \pi_{ij,k} \, P(r_t = i \mid \varphi_k \,, y_{0:t}) \end{aligned} \tag{3.21}$$

We can evaluate the posterior mode probabilities for a given class φ_k by

$$P(r_t = j \,|\varphi_k, y_{0:t}) = \frac{1}{a_t} p_\theta(y_t|r_t = j, \varphi_k \,, y_{0:t-1}) \cdot \sum_{i=1}^{s(\varphi_k)} \pi_{ij} \, P(r_{t-1} = i \mid \varphi_k \,, y_{0:t-1}) \tag{3.22}$$

where a_t is a normalizing constant. Similarly, the measurement updates for each class can be carried out by the following recursion:

$$p_\theta(\{x_t, \varphi_k\}|y_{0:t}) = \frac{p_\theta(y_t|\{x_t, \varphi_k\}, y_{0:t-1}) p_\theta(\{x_t, \varphi_k\}|y_{0:t-1})}{p_\theta(y_t|y_{0:t-1})} \tag{3.23}$$

$$= \frac{1}{c_k} \lambda_{\varphi_k}(y_t|x_t) p_\theta(\{x_t, \varphi_k\}|y_{0:t-1}) \tag{3.24}$$

with,

$$c_k = \sum_{i=1}^{s(\varphi_k)} \int_{x_t \in \mathcal{X}} \lambda_{\varphi_k}(y_t|x_t) p_\theta(\{x_t, \varphi_k\}|y_{0:t-1}) \, dx_t \tag{3.25}$$

Finally, we arrive at an expression for the target class posterior probabilities as:

$$P(\varphi_k|y_{0:t}) = \frac{p_\theta(y_t|\varphi_k, y_{0:t-1})P(\varphi_k|y_{0:t-1})}{\sum_{\varphi_k \in \Phi} p_\theta(y_t|\varphi_k, y_{0:t-1})P(\varphi_k|y_{0:t-1})}. \tag{3.26}$$

The state estimates for each class $\varphi_k \in \Phi$ can be estimated via the MMSE estimator or MAP estimator. If we denote $x_t^{(i,k)}$ and $w_t^{(i,k)}$ as the particles and their corresponding weights for the k^{th} class respectively, we can form the MMSE approximation as

$$\hat{x}_{\varphi_k|t}^{MMSE} = \sum_{r_t \in \mathcal{S}} \int_{x_t \in \mathcal{X}} x_t p_\theta(x_t, r_t, \varphi_k|y_{0:N})\, dx_t \approx \sum_{i=1}^{N_p} w_t^{(i,k)}\, x_t^{(i,k)} \tag{3.27}$$

and the corresponding estimates for the modes $\hat{r}_{\varphi_k|t}$ are given by

$$\hat{r}_{\varphi_k|t}^{\text{MMSE}} = \sum_{i=1}^{N_p} w_t^{(i,k)}\, r_t^{(i,k)} \tag{3.28}$$

For non-linear non-Gaussian systems, sometimes the MMSE estimator can perform poorly. In target tracking, multi-modal posteriors are common, and an MMSE estimator might be a poor choice since it can give a point estimate in areas of low probability. In these situations, a MAP estimate might perform better. A detailed discussion of this is outside the scope of this text; see [97][98] for more details on the implementation of the MAP estimator using particle filters, and a comparison of its performance with the MMSE estimator. The final JTC algorithm is now presented in Algorithm 8 below.

Algorithm 8: Multiple Model Particle Filter Bayesian Classifier

INPUTS: $\{y_{0:N}\}$

1. Initialization at time $t = 0$,
   - For $\varphi_k = 1, 2, \ldots s(\varphi)$
     - set $P(\varphi_k) = P_0(\varphi_k)$
   - For $i = 1, \ldots, N_p$
     - draw particles $x_0^i \sim p_\theta(x_0, \varphi_k)$
     - draw particles $r_0^i \sim p_\theta(r_0, \varphi_k)$
2. For φ_k , $k = 1, 2, \ldots m$ (in parallel)

   Prediction:

   - Predict mode particles $\{r_{t+1}^i\}_{i=1}^{N_p}$ forward based on $\{r_t^i\}_{i=1}^{N_p}$ and $\Pi_{\{\varphi_k\}}$
   - Draw process noise samples $\{v_t^i\}_{i=1}^{N_p} \sim p_\theta\,(v\,|\theta_k, \varphi_k)$ and predict particles x_{t+1}^iforward using the Markov transition density:

$$x_{t+1} = f_\theta(x_t, r_{t+1}, v_t, \varphi_k)$$

   Measurement Update:

- For $i = 1, \dots, N_p$ evaluate importance weights for the augmented state particles $z_t = \{x_{t+1}^i, r_{t+1}^i\}_{i=1}^{N_p}$ as

$$w_{t+1}^i = p_\theta(y_{t+1}|x_{t+1}^i, r_{t+1}^i, \varphi_k)$$

- Evaluate $p_\theta(y_{t+1}|\varphi_k, y_{0:t}) = \sum_{i=1}^{N_p} w_{t+1}^i$
  - Set $\Gamma(\varphi_k) = \sum_{i=1}^{N_p} w_{t+1}^i$

Selection Step:

- Normalize importance weights as

$$w_t^i = \frac{w_t^i}{\sum_{j=1}^{N_p} w_t^j}$$

- Evaluate $\widehat{N}_{eff}$
- If $\widehat{N}_{eff} \leq N_{THR}$ resample particles $\{z_{t+1}^i\}_{i=1}^{N_p} = \{x_{t+1}^i, r_{t+1}^i\}_{i=1}^{N_p}$ and reset weights according to ***Algorithm 2***

Compute state MMSE (or substitute with MAP if desired) estimate and posterior mode probabilities:

- $\hat{x}_{\varphi_k|t}^{MMSE} = \sum_{i=1}^{N_p} w_t^{(i,k)} x_t^{(i,k)}$
- $\hat{r}_{\varphi_k|t}^{MMSE} = \sum_{i=1}^{N_p} w_t^{(i,k)} r_t^{(i,k)}$

3. Compute posterior class probabilities and combined state estimate:
   - For φ_k, $k = 1, 2, \dots m$ evaluate class posteriors according to (3.26) as

$$P(\varphi_{k|t}|y_{0:t}) = \frac{\Gamma(\varphi_k)P(\varphi_{k|t-1}|y_{0:t-1})}{\sum_{\varphi_k \in \Phi}\Gamma(\varphi_k)P(\varphi_{k|t-1}|y_{0:t-1})}$$

   - Evaluate combined state estimate:

$$\hat{x}_t = \sum_{\varphi_k \in \Phi} P(\varphi_k|y_{0:t}) \cdot \hat{x}_{\varphi_k|t}$$

4. Set $t \mapsto t + 1$ go back to step 2.

OUTPUTS: $\{P(\varphi_{k|t}|y_{0:t}), \hat{x}_t\}_{k=1}^m$ for $t = 1, \dots, N$

4 System Modeling and Data Fusion

We now turn our attention to explicit system modeling for target tracking and the incorporation of data fusion techniques. Sections 4.1 and 4.2 introduce some kinematic motion models to describe target motion, and measurement models, respectively for remote sensing applications. We build upon this in section 4.3 by tackling the multi-sensor problem showing how to process measurements from multiple sensors sequentially.

4.1 Dynamic and Measurement models

In dynamic target tracking, the choice of models can drastically alter the performance of tracking systems, and their importance cannot be stressed enough. The goal of target tracking is to extract information about the state of an object from available measurements or observations.

Here we will introduce two-dimensional kinematic motion models. These class of models generally fall into two categories: maneuver and nonmaneuver models. There are a vast number of motion models developed over the years such as constant velocity (CV), and coordinated turn (CT), and variable turn models to name a few. The models presented here are the discrete-time equivalents of their continuous time versions. For a more thorough and complete treatment of these kinematic motions, the reader is referred to [7] and [99].

It will be assumed that the state of an object will be described by the state vector at time t as $x_t = [x_t^c, vx_k, y_t^c, vy_k]'$, where $(x_t^c, y_t^c)' \in \mathbb{R}^2$ are the horizontal and vertical Cartesian position coordinates in Euclidian space, and $vx_t = \dot{x_t^c}$ and $vy_t = \dot{y_t^c}$ are the corresponding velocities in each direction. We assume each state variable is a member of state space of the system $x_t \in \mathcal{X} \subseteq \mathbb{R}^4$ and evolves according to the mode dependent Markov transition density $p_\theta(x_{t+1}|x_t, r_{t+1})$.

The exact description of target motion is never truly known, and therefore some uncertainty is always present due to model inaccuracy and external factors such as turbulence. Furthermore, in most target tracking applications, the dynamic state of real targets to be estimated is poorly described by a single kinematic model [89]; multiple model descriptions of target dynamics are desirable in these situations.

4.1.1 Constant Velocity Models

Constant velocity models are a class of nonmaneuver models. This means that the target travels at a constant velocity as well as straight and level. The constant velocity model can be expressed as

$$x_{t+1} = F_{cv}x_t + v_t \tag{4.1}$$

Where

$$F_{cv} = diag[F_2, F_2], \tag{4.2}$$

$$F_{cv} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{4.3}$$

T is the sampling time and $v_t \in \mathbb{R}^4$ is an additive white Gaussian noise (AWGN) process to account for uncertainties in the model with covariance matrix

$$cov(e_t) = diag\left[\frac{\sigma_x^2}{T} \cdot Q_2, \frac{\sigma_y^2}{T} \cdot Q_2\right], \tag{4.4}$$

$$Q_2 = \begin{bmatrix} \frac{T^4}{3} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix}. \tag{4.5}$$

where σ_x^2 and σ_y^2 are the *power spectral density* (PSD) of the acceleration for each directional component.

4.1.2 Coordinated Turn Models

A model that describes an object that performs a constant acceleration is known as a coordinated turn model. Let's assume the turn-rate ω is *known.* Then the CT model with *known turn-rate* can be described as follows

$$x_{t+1} = F_{ct}x_t + v_t \tag{4.6}$$

where F_{ct} is defined as

$$F_{ct} = \begin{bmatrix} 1 & \frac{\sin \omega T}{\omega} & 0 & -\frac{1 - \cos \omega T}{\omega} \\ 0 & \cos \omega T & 0 & -\sin \omega T \\ 0 & \frac{1 - \cos \omega T}{\omega} & 1 & \frac{\sin \omega T}{\omega} \\ 0 & \sin(\omega T) & 0 & \cos \omega T \end{bmatrix} \tag{4.7}$$

and $v_t \sim p_v(\cdot)$ is also an additive white Gaussian noise process with covariance (see [99]):

$$cov(v_t) = \sigma_\omega^2 \cdot \begin{bmatrix} \frac{2(\omega T - \sin \omega T)}{\omega^3} & \frac{1-\cos \omega T}{\omega^2} & 0 & \frac{\omega T - \sin \omega T}{\omega^2} \\ \frac{1-\cos \omega T}{\omega^2} & T & -\frac{\omega T - \sin \omega T}{\omega^2} & 0 \\ 0 & -\frac{\omega T - \sin(\omega T)}{\omega^2} & \frac{2(\omega T - \sin \omega T}{\omega^3} & \frac{1-\cos \omega T}{\omega^2} \\ \frac{\omega T - \sin \omega T}{\omega^2} & 0 & \frac{1-\cos(\omega T)}{\omega^2} & T \end{bmatrix} \quad (4.8)$$

It is important to note that in practice the turn-rate is typically not known. An alternative approach would be to assume ω is unknown and include it as a component of the state vector x_t. If a multiple model configuration is implemented with a fixed number of turn-rates $\{\omega_1, \dots, \omega_n\}$, then this is referred to as a *multiple turn-rate* model.

4.2 Radar and Optical Sensor Measurement Models

Since we do not know the actual state of a system but instead a transformed version of them we must now define these measurement equations. In the case of remote sensing, specifically in radar systems, measurements are usually in polar or spherical coordinates and must be transformed to Cartesian coordinates. All of the following models can be extended to three dimensions to include elevation, see [100] for more details. The measurement equation is defined by a nonlinear mapping function $h_\theta: \mathcal{X} \rightarrow \mathcal{M} \subseteq \mathbb{R}^{n_y}$, where $\mathcal{M}$ is the *measurement space*, and n_y is the dimension of the measurements y_t.

$$y_t = \begin{bmatrix} r_t \\ b_t \\ \dot{r}_t \end{bmatrix} = h_\theta(x_t) + e_t = \begin{bmatrix} h_r(x_t) \\ h_b(x_t) \\ h_d(x_t) \end{bmatrix} + \begin{bmatrix} e_{rt} \\ e_{bt} \\ e_{dt} \end{bmatrix} \quad (4.9)$$

where r_t, b_t, $\dot{r}_t$, are the measured range, bearing, and Doppler-derived range rate respectively at time t, and $e_t \sim p_e(\cdot)$ is a white noise multivariate Gaussian process that characterized the measurement noise or error with statistics:

$$cov(e_t) \triangleq R_s = diag(\sigma_r^2, \sigma_b^2, \sigma_d^2). \quad (4.10)$$

Also,

$$h_\theta(x_t) = \begin{bmatrix} h_r(x_t) \\ h_b(x_t) \\ h_d(x_t) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_t^c)^2 + (y_t^c)^2} \\ \arctan((y_t^c / x_t^c) \\ (\frac{x_t^c \cdot vx_k + y_t^c \cdot vy_k}{\sqrt{(x_t^c)^2 + (y_t^c)^2}}) \end{bmatrix}. \quad (4.11)$$

Care must be taken to ensure that a four-quadrant arctangent function must be used to avoid ambiguities and acquire a proper estimate of the target position. It is also important to note that not all sensors use all three kinematic measurements described above. For example, optical sensors only provide bearing information, but with much higher accuracy. Long-range surveillance radars only measure rage and bearing, and sometimes range rate, but cannot be extended to include elevation in 3D models as mentioned earlier [100]. The measurement equation at time t for an optical sensor therefore takes only the bearing component in $b_{opt,t}$ and can be expressed as [101]:

$$y_{opt,t} = b_{opt,t} = h_{opt,\theta}(x_t) + e_{opt,t} \quad (4.12)$$

$$h_{opt,\theta}(x_t) = \arctan((y_t^c / x_t^c) \quad (4.13)$$

where $e_{opt,t} \sim \mathcal{N}(0, \sigma_{opt}^2)$ is also AWGN with the distinction that $\sigma_{opt}^2 \ll \sigma_b^2$.

4.3 Sensor Data Fusion

When gathering data from multiple sensors, we must describe what are known as a *multi-source measurement models.* First, we will discuss basic assumptions and then specify the multi-sensor likelihood functions that will be used to describe the uncertainty in the fused data measurements. Suppose we have a total of s sensors each with its own sensor tag $j = \{1, \dots, s\}$, then the multi-sensor measurement space is the disjoint union

$$\mathcal{Y}_0 = \overset{1}{\mathcal{Y}}_0 \biguplus \cdots \biguplus \overset{s}{\mathcal{Y}}_0 \quad (4.14)$$

where $\overset{j}{\mathcal{Y}}_0$ corresponds to measurement space for jth sensor. In practice for tracking applications, the sensors themselves are described by a state vector. If for example, a sensor is on an airborne surveillance aircraft or UAV then we could describe the sensor state as $x^* = (x^s, y^s, v_x^s, v_y^s, \omega^s, \ell^s, \mu, \chi)$. Here (x^s, y^s) are the position parameters, (v_x^s, v_y^s) are the corresponding velocities, ω^s is the turn radius, ℓ^s is the fuel level, μ represents the sensor's current mode, and χ represents the currently selected datalink channel used for transmission for the particular sensor. Now that this has been established, we can define the joint state space for all sensors as

$$\xi_0 = \overset{1}{\xi}_0 \biguplus \cdots \biguplus \overset{s}{\xi}_0 \quad (4.15)$$

where $\overset{j}{\xi}_0 \ni \overset{j}{x}^*$ is the state space for the j^{th} sensor. In a multi-sensor multi-target system, it is common in control theory to regard the evolution of the targets and sensors as a joint stochastic process described in an augmented state variable $\varsigma = (x, x^*)$ [55].

Now that basic assumptions have been established, we can explore techniques for processing measurement from multiple sensors. For simplicity, let's assume we have two sensors that generate statistically independent observations $\overset{1}{y}_t \in \overset{1}{\mathcal{Y}}_0$ and $\overset{2}{y}_t \in \overset{2}{\mathcal{Y}}_0$ that have the same sampling time. Under the assumptions from the previous section (4.8-4.10) and expanding on the notation from 2.5c we can express the corresponding *sensor likelihood functions* for the j^{th}sensor as $p_\theta\left(\overset{j}{y}_t \middle| x_t\right) \sim \mathcal{N}(\overset{j}{h}_\theta(x_t), R_{s,j})$. Then the process of measurement fusion can be done recursively through the Bayesian update equation by processing one measurement and then the next:

$$p_\theta\left(x_{t+1} \middle| Y_{0:t}, \overset{1}{y}_{t+1}\right) \propto p_\theta\left(\overset{1}{y}_{t+1} \middle| x_{t+1}\right) \cdot p_\theta(x_{t+1} | Y_{0:t}) \tag{4.16a}$$

$$p_\theta\left(x_{t+1} \middle| Y_{0:t}, \overset{1}{y}_{t+1}, \overset{2}{y}_{t+1}\right) \propto p_\theta\left(\overset{2}{y}_{t+1} \middle| x_{t+1}\right) \cdot p_\theta\left(x_{t+1} \middle| Y_{0:t}, \overset{1}{y}_{t+1}\right) \tag{4.16b}$$

Where we have defined $Y_{0:t} = \{\overset{1}{y}_{0:t}, \overset{2}{y}_{0:t}\} \in \mathcal{Y}_0$ to be all the measurements available at time t from both sensors. This process can be repeated for additional sensors.

If the samples are gathered at the exact same time then another way this can be done is by updating the state using the joint measurement likelihood:

$$p_\theta\left(\overset{1}{y}_{t+1}, \overset{2}{y}_{t+1} \middle| x_{t+1}\right) \triangleq p_\theta\left(\overset{1}{y}_{t+1} \middle| x_{t+1}\right) \cdot p_\theta\left(\overset{2}{y}_{t+1} \middle| x_{t+1}\right) \tag{4.17}$$

To see how we can formulate this joint measurement model for our problem, let's first assume that samples from both sensors are gathered at the same time and consider the *joint measurement model*:

$$\begin{pmatrix} \overset{1}{y}_t \\ \overset{2}{y}_t \end{pmatrix} = \begin{pmatrix} \overset{1}{h}_\theta(x_t) \\ \overset{2}{h}_\theta(x_t) \end{pmatrix} + \begin{pmatrix} \overset{1}{e}_t \\ \overset{2}{e}_t \end{pmatrix} \tag{4.18}$$

where we assume that the joint correlation matrix R_{ss} of $\overset{1}{e}_t$ and $\overset{2}{e}_t$ is known. In our case since the sensors generate statistically independent variables (this must not always be the case) we have

$$R_{ss} = \begin{bmatrix} R_{s,1} & 0 \\ 0 & R_{s,2} \end{bmatrix} \tag{4.19}$$

which can be justified by using the mathematics of Gaussians. We can, therefore, express the joint measurement update equation as a single multivariate Gaussian as:

$$p_\theta\left(\overset{1}{y}_{t+1}, \overset{2}{y}_{t+1} \middle| x_{t+1}\right) \backsim \mathcal{N}(h_\theta, R_{ss}) \tag{4.20}$$

where

$$h_\theta = \begin{pmatrix} \overset{1}{h_\theta}(x_t) \\ \overset{2}{h_\theta}(x_t) \end{pmatrix} \tag{4.21}$$

These results can be generalized for situations where more sensors are added (or removed), even when they differ in their dimensions or measurement spaces. By repeating the process just explained we can express the fused measurement equation for a (finite) number of s sensors as

$$\begin{pmatrix} \overset{1}{y}_t \\ \vdots \\ \overset{j}{y}_t \\ \vdots \\ \overset{s}{y}_t \end{pmatrix} = \begin{pmatrix} \overset{1}{h_\theta}(x_t) \\ \vdots \\ \overset{j}{h_\theta}(x_t) \\ \vdots \\ \overset{s}{h_\theta}(x_t) \end{pmatrix} + \begin{pmatrix} \overset{1}{e}_t \\ \vdots \\ \overset{j}{e}_t \\ \vdots \\ \overset{s}{e}_t \end{pmatrix} \tag{4.22}$$

The covariance matrix (4.14) can also be expanded in a similar fashion:

$$R_{ss} = \begin{bmatrix} R_{s,1s} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & 0 & \vdots \\ 0 & 0 & R_{s,j} & 0 & 0 \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & R_{s,s} \end{bmatrix} \tag{4.23}$$

where we again assume that all sensors have the same sampling time.

In the simulations to follow we will neglect all elements of the sensor states $\overset{j}{x}^*$except the position elements (x^s, y^s) which will be held constant. It is straightforward to generalize the algorithms to incorporate all the information contained in each $\overset{j}{x}^*$ as needed. If for example, using an airborne radar or on a ship moving across the ocean. An example of a system where these techniques would be required is shown in Figure 4.1 which shows a naval vessel with an integrated mast containing multiple heterogeneous sensors.

Fig. 4.1. A photo of a Thales integrated mast, seen on top of the naval vessel, containing all major radars, sensors, and antennas.

5 Simulations and Results

This chapter provides the presentation and discussion of simulations done in MATLAB and their results. The purpose of these simulations is to illustrate the capabilities of the work in Chapters 3 and 4 to address the three primary objectives of this thesis described in the introduction. The first is the effectiveness of the system identification techniques, presented in the previous chapter for training JMNLS. The second thing is to show the suitability of these JMNLS for the classification of dynamic object behavior while providing modularity. Section 5.1 will give an example of the learning phase, while section 5.2 gives three examples of joint tracking and behavior classification with different sensor configurations.

5.1 Parameter Learning

In this section, an example will be given to examine and illustrate the capabilities of Algorithm 7 for system identification in target tracking.

5.1.1 Computing Closed-Form Maximizers

In the case of the dynamic motion models defined in the previous chapter, it is possible to find closed-form maximizers, and this will aid in speeding up simulations in the next section. We will assume that we are only estimating the transition probabilities and the process noise parameters since the noise characteristics of a sensor system are typically known in practice. Those parameters do not need to be estimated, but an extension to evaluate them is trivial.

Let us now take a special case of (3.1), and find a closed-form maximizer $\Lambda(\cdot)$. Assume that the system is of the separable form

$$x_{t+1} = f_\theta(x_t, r_{t+1}) + v_{\theta,t}(r_t) \quad (5.1a)$$
$$y_t = h_\theta(x_t) + e_t \quad (5.1b)$$

where $v_{\theta,t}(r_t) \sim \mathcal{N}\big(0, R_v(r_t)\big)$, $f_\theta: \mathbb{R}^{n_x} \longrightarrow \mathbb{R}^{n_x}$, $h_\theta: \mathbb{R}^{n_x} \longrightarrow \mathbb{R}^{n_y}$, and let $r_t \in \mathcal{S} = \{1, 2, 3, \dots, s\}$. More specifically, we will assume that the unknown process noise covariance matrix $R_v(r_t)$ can be expressed in the following form:

$$R_v(r_t) = \sigma^2_{(r_t)} R_Q \quad (5.2)$$

where $R_Q \in \mathbb{S}_+^{n_x}$ (symmetric positive semi-definite, $n_x \times n_x$) and $\sigma^2_{(r_t)}$ are the individual unknown noise variances. The measurement noise distribution $p_e(\cdot)$ and its parameters are assumed to be known, but this need not be the case and the derivations to come can easily be generalized to include them as unknowns in a straightforward manner. The set of unknown parameters, therefore, contains only the process noise parameters and the TPM, so we have

$\theta = \left(\{\theta(r_t)\}_{r_t=1}^{S}, \Pi\right) \subseteq \Theta$, where in this case $\theta(r_t) = \sigma_{(r_t)}^2$. In the case of this example, it is possible to find a closed form maximizer $\Lambda(\cdot)$ since the process noise $v_{\theta,t}(r_t)$ is Gaussian and because only (3.18) is dependent on the transition probabilities π_{ij}. Lastly, we assume that we have prior densities $p_\theta(r_1^i) \sim \mathcal{U}(0,1)$ and the initial state x_0 is fully known.

We proceed by first recognizing that the gradient for $\widehat{\Upsilon_1}$ vanishes and does not contribute to finding a maximizer since $p_\theta(r_1^i) \sim \mathcal{U}(0,1)$. The argument is similar for why the MLE is equal to the MAP estimator with a uniform prior. The second component $\widehat{\Upsilon_2}$ also vanishes since the state is fully known (i.e., let $p(x_t^i|r_t) \sim \mathcal{N}(x_0, 0_{(n_x \times n_x)})$.

Moving on to the third component, recall that $\mathcal{S}$ is the set of all modes. We can then find the maximizer for the transition probabilities $\hat{\pi}_{ij}$ by solving an equivalent constrained optimization problem:

$$\begin{aligned} \underset{p_{ij}}{minimize} \quad & \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \Psi_{ij} \log \pi_{ij} \\ subject\ to \quad & \sum_{j \in} \pi_{ij} = 1, \ \forall i \ \in \mathcal{S} \\ & \pi_{ij} \geq 0, \forall (i,j) \in \mathcal{S} \times \mathcal{S} \end{aligned} \tag{5.3}$$

with

$$\Psi_{ij} = \sum_{t=1}^{N-1} \sum_{k=1}^{N_p} \sum_{l=1}^{N_p} w_{t|N}^{kl} \, \mathbb{1}(r_t^k = j) \mathbb{1}(r_t^l = i) \tag{5.4}$$

It can be shown that if $\Psi_{ij} \in \mathbb{R}_+ \ \forall (i,j) \in (\mathcal{S} \times \mathcal{S})$ then

$$\hat{\pi}_{ij} = \frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}}, \forall (i,j) \in (\mathcal{S} \times \mathcal{S}) \tag{5.5}$$

is a maximizer for (5.3). This is the usual maximizer for HMMs [78]. The proof is included in Appendix C for completeness.

For calculating $\widehat{\Upsilon_4}$ will make use of the following two identities in the following derivations (see [47]). Suppose $\phi \in \mathbb{R}^p$ and $\mathrm{A}(\phi) \in \mathbb{R}^{n \times n}$, then

$$\frac{\partial}{\partial \phi_k} \log \det[\,\mathrm{A}(\phi)] = trace\left(\mathrm{A}^{-1}(\phi) \frac{\partial}{\partial \phi_k} \mathrm{A}(\phi)\right) \tag{5.6}$$

and

$$\frac{\partial}{\partial \phi_k} \mathrm{A}^{-1}(\phi) = -\mathrm{A}^{-1}(\phi) \frac{\partial}{\partial \phi_k} \mathrm{A}(\phi) \mathrm{A}^{-1}(\phi). \tag{5.7}$$

Where we define $\partial \mathrm{A}^{-1}(\phi)/\partial \phi_k \in \mathbb{R}^{n \times n}$ as the matrix with element $[i,j]$ as $\partial [\mathrm{A}^{-1}(\phi)]_{ij}/\partial \phi_k$. Also to simplify the notation, we will let $\mathbb{1}_{r_t}(r_{t+1}^j) = \mathbb{1}(r_{t+1}^j = r_t)$ for some $r_t \in \mathcal{S}$.

The fourth component $\widehat{\Upsilon_4}$ deals with finding the process noise parameters, since we have a Gaussian Markov transition density p_θ where we have assumed the process noise parameters $\theta(r_t) = \sigma_{r_t}^2$ are unknown. By construction, we have $p_\theta(x_{t+1}^j | x_t^i, r_{t+1}^j) \sim \mathcal{N}(f_\theta(x_t^i, r_{t+1}^j), R_v(r_{t+1}^j))$, so we can evaluate (3.17d) explicitly:

$$\nabla_\theta \widehat{\Upsilon_4} = \sum_{t=1}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \cdot \frac{\partial}{\partial \theta} \left[\log \frac{\exp\left(-\frac{1}{2} \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j) \right)^T R_v^{-1}(r_{t+1}^j) \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j) \right) \right)}{\sqrt{(2\pi)^{n_x} \det R_v(r_{t+1}^j)}} \right] \tag{5.8}$$

Each component of the gradient can be computed individually. To make the calculations easier, we exploit the structure of the covariance matrix (5.2) and factor out the noise parameter and maximize (3.17d) directly for each mode $r_t \in \mathcal{S}$:

$$\nabla_\theta \widehat{\Upsilon_4} = \sum_{t=1}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \mathbb{1}_{r_t}(r_{t+1}^j) \frac{\partial}{\partial \sigma_{r_t}^2} \Big[\Big(-\frac{1}{2} \sigma \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j) \right)^T R_v^{-1}(r_{t+1}^j) \cdot R_v^{-1}(r_{t+1}^j) \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j) \right) - \log \sqrt{(2\pi)^{n_x} \det\left(R_v(r_{t+1}^j) \right)} \Big]$$

$$= \sum_{t=1}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \mathbb{1}_{r_t}(r_{t+1}^j) \frac{\partial}{\partial \sigma_{r_t}^2} \Big[-\frac{1}{2} \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j) \right)^T \cdot R_v^{-1}(r_{t+1}^j) \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j) \right)$$

$$- \frac{1}{2}\left(\log\left(\det R_v(r_{t+1}^j)\right) - \log(2\pi)^{\frac{n_x}{2}}\right)\Bigg]$$

$$= \sum_{t=1}^{N-1}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} w_{t|N}^{ij}\, \mathbb{1}_{r_t}(r_{t+1}^j)\left[-\frac{1}{2}\left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right)^T\right.$$

$$\cdot \frac{\partial}{\partial \sigma_{r_t}} R_v^{-1}(r_{t+1}^j)\left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right)$$

$$\left. - \frac{1}{2}\frac{\partial}{\partial \sigma_{r_t}^2}\left(\log\left(\det R_v(r_{t+1}^j)\right)\right)\right]$$

$$= \sum_{t=1}^{N-1}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} w_{t|N}^{ij}\, \mathbb{1}_{r_t}(r_{t+1}^j)\left[\frac{1}{2}\left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right)^T R_v^{-1}(r_{t+1})\right.$$

$$\cdot \frac{\partial}{\partial \sigma_{r_t}^2} R_v(r_{t+1}^j) R_v^{-1}(r_{t+1}^j)\left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right)$$

$$\left. - \frac{1}{2} trace\left(R_v^{-1}(r_{t+1}^j)\frac{\partial}{\partial \sigma_{r_t}^2} R_v(r_{t+1}^j)\right)\right]$$

$$= \sum_{t=1}^{N-1}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} w_{t|N}^{ij}\, \mathbb{1}_{r_t}(r_{t+1}^j)$$

$$\cdot \left[\frac{1}{2}\left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right)^T R_v^{-1}(r_{t+1}^j) R_Q R_v^{-1}(r_{t+1}^j)\right.$$

$$\left. \cdot \left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right) - \frac{1}{2} trace\left(R_v^{-1}(r_{t+1}^j) R_Q\right)\right]$$

$$= \sum_{t=1}^{N-1}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} w_{t|N}^{ij}\, \mathbb{1}_{r_t}(r_{t+1}^j)$$

$$\cdot \left[\frac{1}{2}\left(x_{t+1}^j - f_\theta(x_t^i, r_{t+1}^j)\right)^T \frac{1}{\sigma_{r_t}^4} R_Q^{-1} R_Q R_Q^{-1}\right.$$

$$\cdot \left(x_{t+1}^{j} - f_\theta\left(x_t^i, r_{t+1}^j\right)\right) \frac{1}{2} trace\left(\frac{1}{\sigma_{r_t}^2} I_{(n_x)}\right)\Bigg]$$

$$= \sum_{t=1}^{N-1} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} w_{t|N}^{ij} \mathbb{1}_{r_t}\left(r_{t+1}^j\right)$$

$$\cdot \left[\frac{1}{2}\left(x_{t+1}^{j} - f_\theta(x_t^i, r_{t+1}^j)\right)^T \frac{1}{\sigma_{r_t}^4} R_Q^{-1}\left(x_{t+1}^{j} - f_\theta(x_t^i, r_{t+1}^j)\right) - \frac{n_x}{2}\sigma_{r_t}^{-2}\right]. \tag{5.9}$$

Where $I_{(n)}$ is the $n \times n$ identity matrix and have used identities (5.1) and (5.2) in the last two steps. Now by setting (5.10) equal to 0 and rearranging terms we arrive at a maximizer for $\sigma_{r_t}^2$:

$$\sigma_{r_t}^2 = \frac{\sum_{t=1}^{N-1}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} w_{t|N}^{ij} \mathbb{1}_{r_t}\left(r_{t+1}^j\right)\left[\left(x_{t+1}^{j} - f_\theta(x_t^i, r_{t+1}^j)\right)^T R_Q^{-1}\left(x_{t+1}^{j} - f_\theta(x_t^i, r_{t+1}^j)\right)\right]}{n_x \cdot \sum_{t=1}^{N-1}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} w_{t|N}^{ij} \mathbb{1}_{r_t}\left(r_{t+1}^j\right)}. \tag{5.10}$$

These closed-form maximizer $\Lambda(\cdot)$, composed of (5.5) and (5.10) will be used in the simulation to follow.

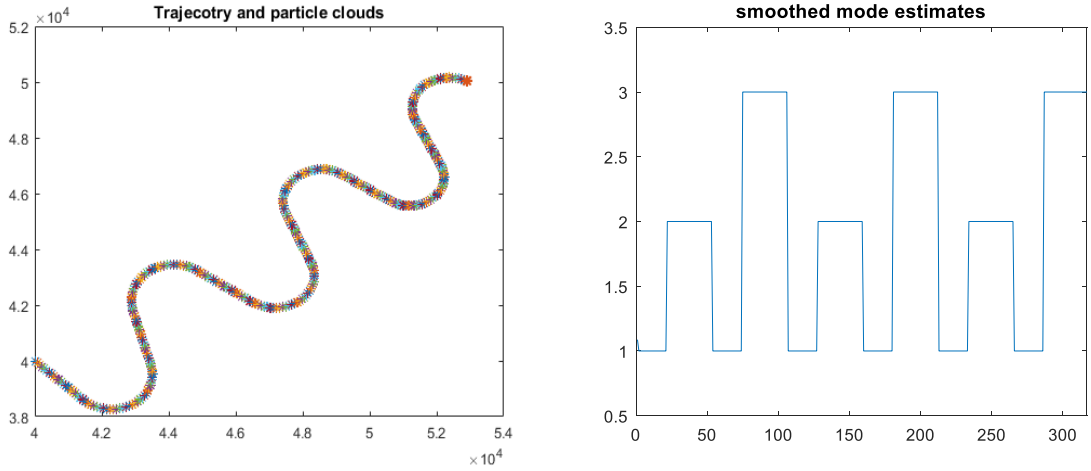
5.1.2 Example 1: Training a Zig-Zag

Now an example will be presented to examine training step provided by Algorithm 7. Due to constraints in computational power and memory, a linear measurement model $y_t = x_t + \xi_t$ where $\xi_t \sim \mathcal{N}(0, \Xi)$, where $\Xi = diag\left(\sigma_{xx}^2, \sigma_{vx}^2, \sigma_{yy}^2, \sigma_{vy}^2\right)$ will be used to for the training step. This relaxes the number of particles needed by reducing the variance for accurate parameter estimation, as opposed to using the measurement models of the previous chapter which are highly nonlinear.

For simplicity, assume the target is performing a "zig-zag" trajectory—the target will begin in a straight pattern and alternate between the three modes creating the pattern in figure 5.1. The first mode will correspond to a constant velocity model. The CT models to describe left and right turns with known turn rates will be represented by the second and third modes respectively. We assume that the system is coupled in the x and y directions for the CV model so $\sigma_x^2 = \sigma_y^2$. These assumptions are summarized in Table 5.1.

Table 5.1. Mode assignments

Mode $r_t \in \mathcal{S} = \{1,2,3\}$	Model	Turn rate (rad/s)	Process noise (m/s^2)
$r_t = 1$	CV	0	σ_1^2
$r_t = 2$	CT	ω_L	σ_2^2
$r_t = 3$	CT	ω_R	σ_3^2



(a) zig-zag trajectory with particle clouds (b) smoothed mode estimates

Fig. 5.1. Trajectory and modes for parameter learning example

In the simulations to follow the closed form maximizers derived in the previous section will be used for the M-step of the EM algorithm. For this simulation, a total of $N_p = 350$ particles were used. The measurement noise was set to $\Xi = diag(3\,m/s\,, 2\ m/s^2\,, 5\ m/s\,, 2\,m/s^2)$. The initial guesses for the true estimates were chosen to be farther than 50% away from the true value, and are listed in table 5.2 and table 5.3 at the end of this section along with the estimated results. For illustration purposes, the stopping criterion is not used in this example to show the convergence behavior of the EM algorithm.

Below in Figure 5.2 the results for the process noise estimates can be seen. It can be seen that estimates get quite close to the true value parameter values for both CT after 20 EM iterations. For the CV model, it takes longer, getting closer to the true value around 80 iterations. Once the estimated value of $\hat{Q}(\theta, \theta_k)$ gets close enough to the true value, the estimates start to hover around while fluctuating. The reason for this fluctuation is due to the sampling nature of SMC methods (particle filters). Since we are approximating $Q(\theta, \theta_k)$, convergence to the true maximum is not feasible unless we let $N_p \to \infty$. Since an approximation of is being calculated using SMC methods, this means the estimates θ_k can also be seen as a stochastic process, and there is no guarantee that the new estimate will lead to a local increase in approximate complete data log-likelihood $Q(\theta, \theta_k)$ for every iteration. Only if the exact value of $Q(\theta, \theta_k)$ can be calculated will the EM algorithm be guaranteed to converge to a local maximum [42]. Since every EM iteration new samples are generated, so is

a new estimate with the variance dependent on the type of SMC method employed. The error of these estimates is limited by the variance of the approximated expectations (3.14a-3.14e), which in turn are dependent on the empirical approximations of the marginal smoothed densities in (3.9a-3.9e). As explained in [74] there are a number of factors that control the variance of the Monte Carlo estimates. One of them being the choice of importance density. Since we did not use the optimal importance density, this already caused a reduction in estimation accuracy.

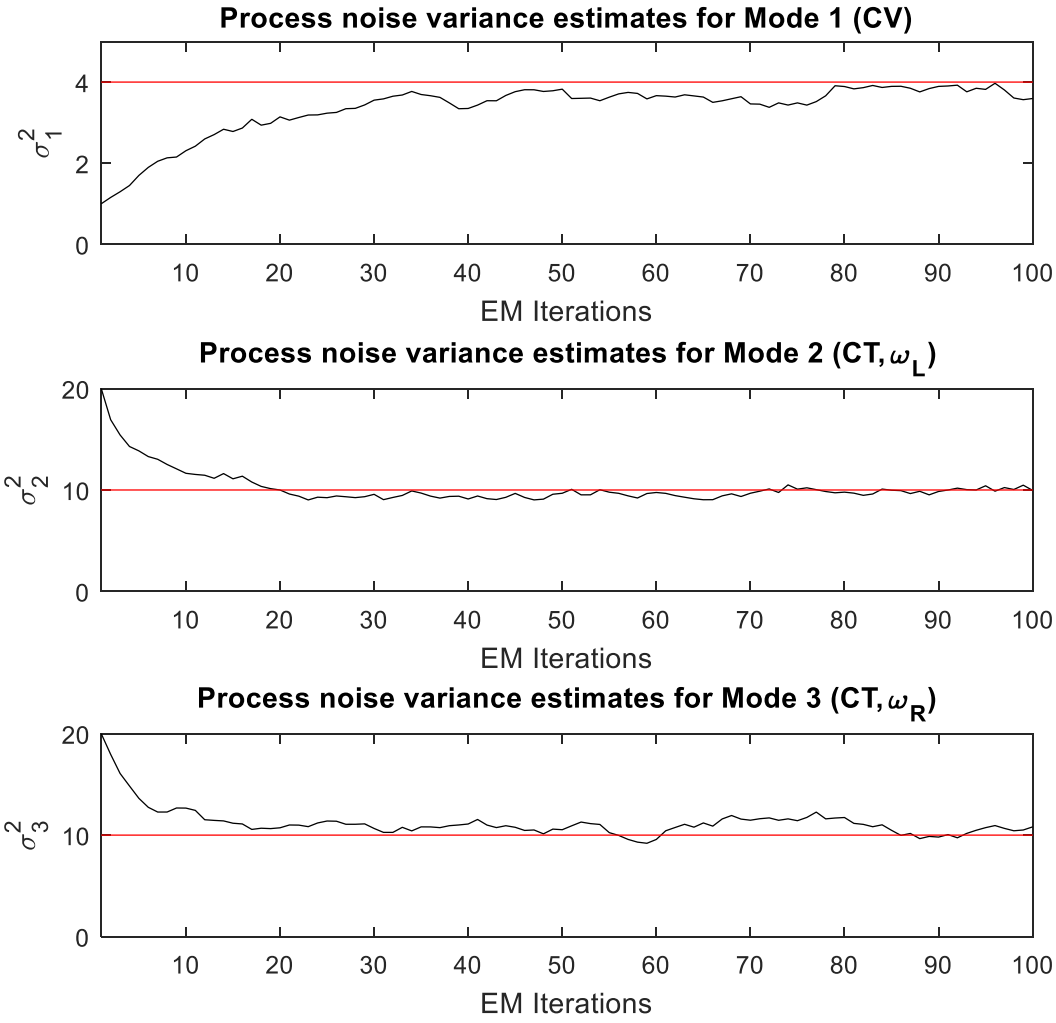


Fig. 5.2. Process noise estimates for 100 EM iterations

Table 5.2. Process noise estimates and true values (m/s^2)

Parameter	σ_1^2	σ_2^2	σ_3^2
Estimated	3.6443	9.8366	10.6993
True Value	4	10	10

The estimated transition probabilities π_{ij} can be seen in Figure 5.3. As can be seen, the EM algorithm arrived at an accurate estimate quite quickly after just a few iterations. These values are quite close to the ground truth, and by careful observation, one can see that they make sense intuitively for zig-zag trajectory which can go either straight or turn left or right.

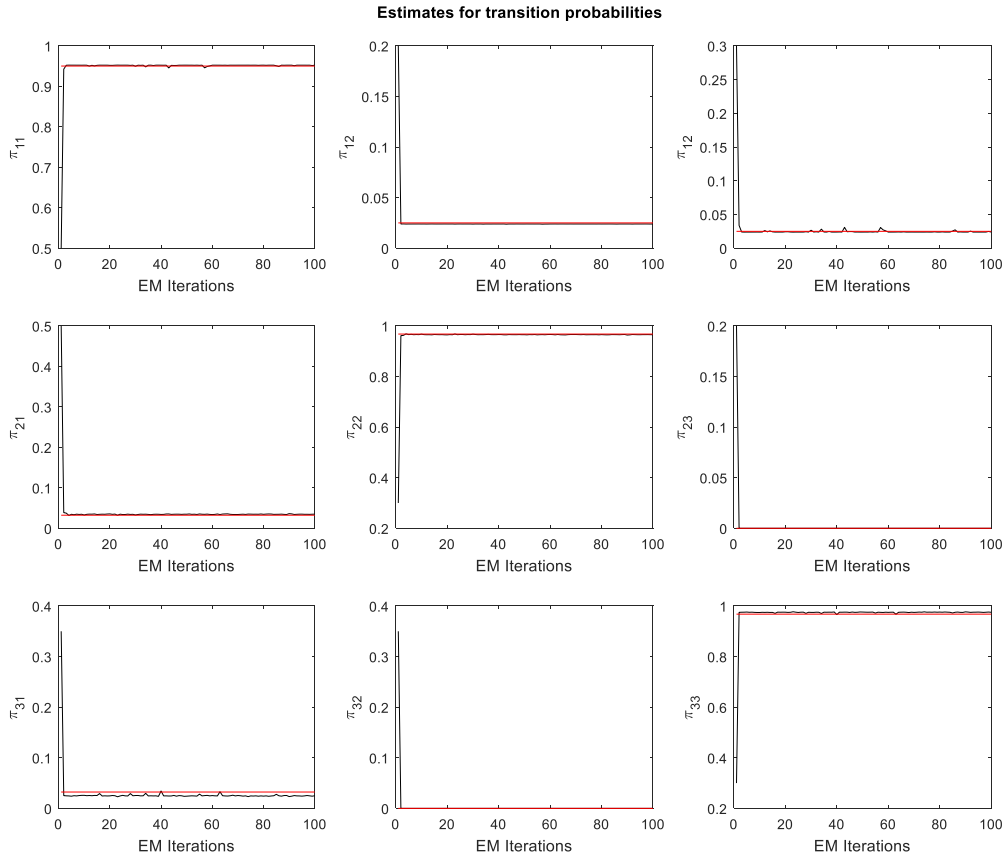


Fig. 5.3. Estimates of transition probabilities π_{ij} for 100 EM iterations. Red line indicates the true value and the black curves indicate the numerical approximations

Table 5.3: Estimates of transition Probabilities and the true values (ground truth)

Transition Probabilities									
	π_{11}	π_{12}	π_{13}	π_{21}	π_{22}	π_{23}	π_{31}	π_{32}	π_{33}
True value	.95	.0250	.0250	.0323	.9677	0	.0323	0	.9677
Estimated	.9521	.239	.240	.0355	.9645	0	.0242	0	.9758

As a final note in this section, it should be reiterated that although a linear measurement model was used, this does not mean the training step cannot be done using a nonlinear measurement model. This was in fact tested, but the results were quite inaccurate, using anything less than 500 particles. As the number of particles was increased, simulations showed estimates closer to their true value, but with limited memory, a cap was reached in how many particles could be used. This issue here was the smoother, which requires storing all the particles, their weights, and the transition probabilities $p_\theta\left(x_{t+1}^j \middle| x_t^i, r_{t+1}^i\right)$ that are necessary to compute the smoothed weights in both the numerator or denominator of (3.12). These values had to be either stored or recomputed, both having negative consequences on available memory storage or runtime. An example using a nonlinear measurement model can be found in Appendix B for illustration purposes.

5.2 Trajectory Classification

This section will cover the testing and evaluation of the joint tracking and classification algorithm (8) proposed in section 3.4 with three examples. In each example, the tracking of the object in motion will be illustrated to give a picture of the trajectory in question. Tracking performance is outside the scope of this thesis, and therefore will not be considered further. Afterward, the posterior class probabilities, the primary quantity of interest, will be carefully considered. In the first two examples, only one sensor will be assumed to be gathering measurements, and in the final example, two extra sensors will be added to the system to illustrate the ease of modularity without the need for retraining. The added sensors will have different noise characteristics, and one of them will be placed at a separate location.

In the work to follow, three classes φ_k of trajectories will be considered. These classes are shown in Figure 5.4. Trajectories (a) and (b) are relatively straightforward. Trajectory (c) must be studied with caution since this pattern consists of an alternating sequence of turn, straight and turn in the opposite direction. It is evident that this same sequence of modes, can also create a "figure eight," by increasing the turn rate or the amount of time spent in each turning mode. Using similar reasoning, by alternating the turn rate, a "weave" pattern can be formed such as those typical of fishing boats continuously sweeping a small area. For the work in this thesis, the trajectories have been chosen to be distinct enough that no overlap occurs as this would unnecessarily complicate the problem at hand–how to distinguish more closely related trajectories will be discussed in Chapter 6. The concept is that there is some ambiguity in specific trajectories, and Algorithm 7 will reflect this by estimating similar transition probabilities. In the simulations to follow, each model was trained using 350 particles and running 10 Monte Carlo averages. For the JTC portion, each one of the parallel filters will run with 15,000 particles each. The reason for the drastic increase in the number of

particles is that filtering performed without smoothing reduces the amount of memory required since it is an online process and storing the data is not necessary. The computational complexity is also significantly reduced allowing for shorter simulation times. This allowed for a more considerable amount of particles to be used in the JTC phase.

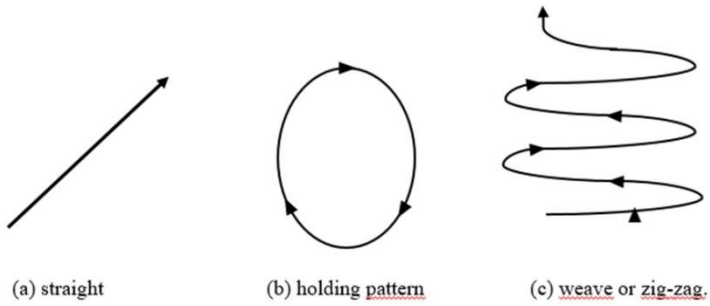


Fig. 5.4. Graphical representation of the classes of trajectories

5.2.1 Example 2: Straight Line

Let us first consider an example where we aim to classify the behavior of an object moving with a constant heading such as an airliner (class 1). The trajectory and filter output corresponding to class for the correct class and the filtered mode sequences are shown in Figure 5.6. Here we assume the target is being tracked by two sensors: one radar and one optical sensor (bearing only) placed at the origin. Their noise characteristics are $\sigma_{r,1}^2 = 15\ m,\ \sigma_{1,b}^2 = 10\ mrad\ , \sigma_{1,d}^2 = 5\ m/s$ for the radar, and $\sigma_{opt}^2 = 1\ mrad$ for the optical sensor. The object is first detected at a distance of approximately 56.5 km away heading northeast (away from the radar) at a speed of approximately 885 km/h (550 mph), the typical speed of a subsonic airliner at cruising altitude. In this example, the trajectory carried out by the airliner is assumed to belong to one of three classes defined in table 5.4. Assume a sampling interval of $T = .5\ s$. For simplicity, the initial state is assumed to be fully known, and the plane is first detected at $x_0 = [40 \times 10^3 m, 175\ m/s\ , 40 \times 10^3 m, 175\ m/s]^T$. As a reminder, each particle filter will run with $N_p = 15{,}000$ particles (per class). The initial mode and class probabilities will both be uniformly distributed so that: $P_0(\varphi_k) \sim \mathcal{U}(0,1)$ and $p_\theta(r_0, \varphi_k) \sim \mathcal{U}(0,1)$. Lastly, the plane will be constrained to turn rates $\omega_L = \omega_R = .25\ rad/s$.

Table 5.4. Class assignments φ_k

Class (k)	1	2	3
trajectory	straight	holding pattern	weave or zig-zag

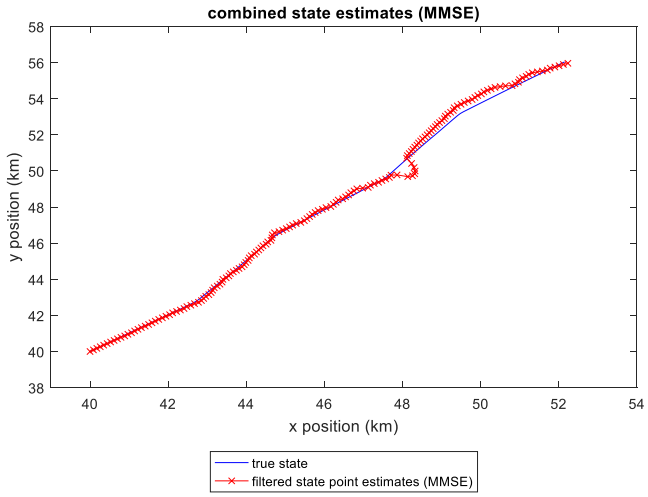
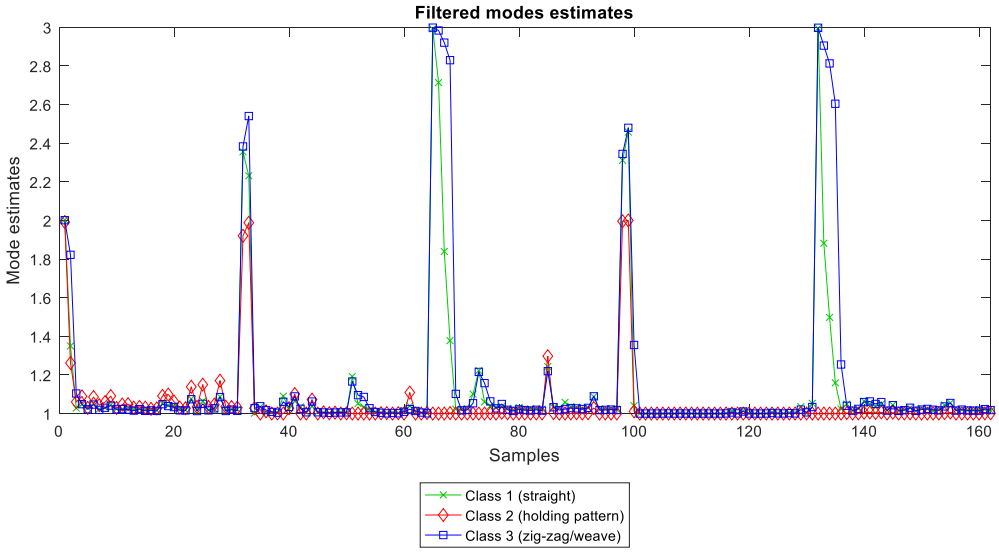
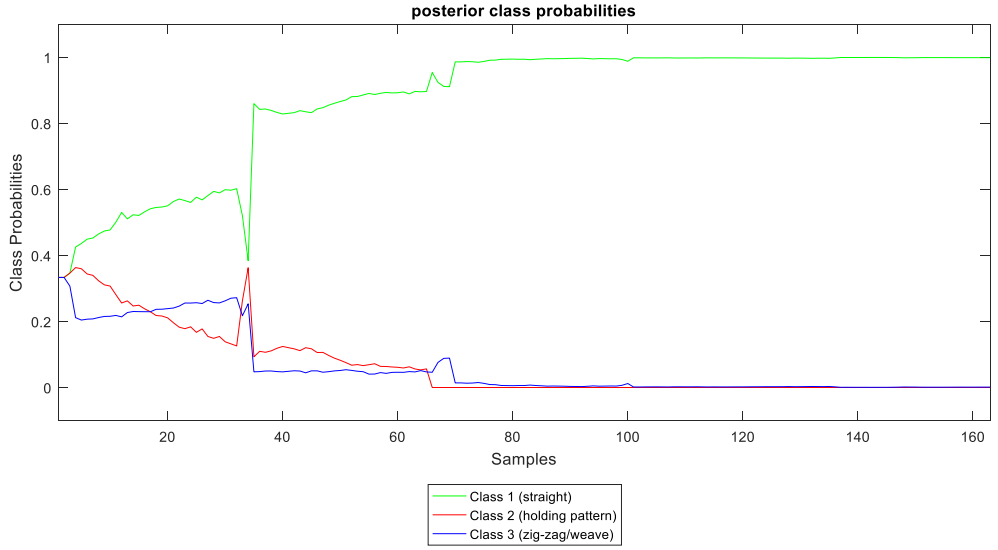


Fig. 5.5. Straight constant velocity trajectory

Looking at Figure 5.5, one will notice slight adjustments in heading. This is typical behavior of an airliner, as the inaccuracy of avionics, turbulence, crosswinds and other factors such as the curvature of the earth do not allow for a constant heading to reach a target destination. While in flight, corrections have to be made every so often to ensure a more direct path. Therefore, for the straight class φ_1, the model was trained using data of trajectories that made slight adjustments in heading while maintaining an overall straight path. Another advantage of this is that if a model is trained with no turns, then naturally all probabilities of mode transitions to left and right turns will be estimated to be zero, or close to it. This could cause potential issues depending on the type of particle filter being used especially during mode transitions. The lack of sufficient particles could cause all the particles to go into the straight mode when a turn is occurring due to the sampling nature of SMC methods. If few or no particles are in the turning modes, then the filter will diverge and fail to track the target. An alternative is to use a more efficient particle filter where the particles are fixed in each mode. This will be discussed further In Chapter 6. For now, we train the models allowing for infrequent small turns to overcome depletion issues typically encountered by the multiple model bootstrap filter [102].



(a)



(b)

Fig. 5.6. System outputs for a (nearly) straight trajectory. (a) filtered mode estimates (b) posterior class probabilities

By analyzing Figure 5.6 closely, we can see that the classifier initially goes through a period of confusion since the object initially begins in a straight line, which could potentially be any one of the three classes. As class 1 begins to rise in probability, there is a slight dip with a corresponding rise in class 2. This is expected since the first turn is slightly left and looking at the filtered modes around the first turn we can see that class two has a precise point estimate of nearly 2 (turn left), therefore making it the most accurate class. As more samples are processed, the posterior class probability rises steadily to its true class, namely a straight trajectory. Another important thing to note is that although both class 3 and class 1 accurately estimate the switch to mode three, we see that class 1 reacts more quickly to return to a straight trajectory as expected. This occurs because the transition probabilities for staying in a turn mode are much higher for zig-zag since it has longer turns.

5.2.2 Example 3: Zig-Zag

For further illustration, now consider the behavior of an airliner with the same assumptions, but this time let us analyze a zig-zag trajectory, pictured below in Figure 5.7.

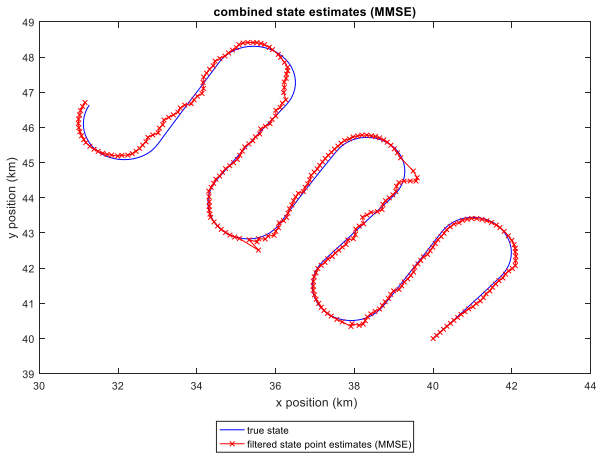
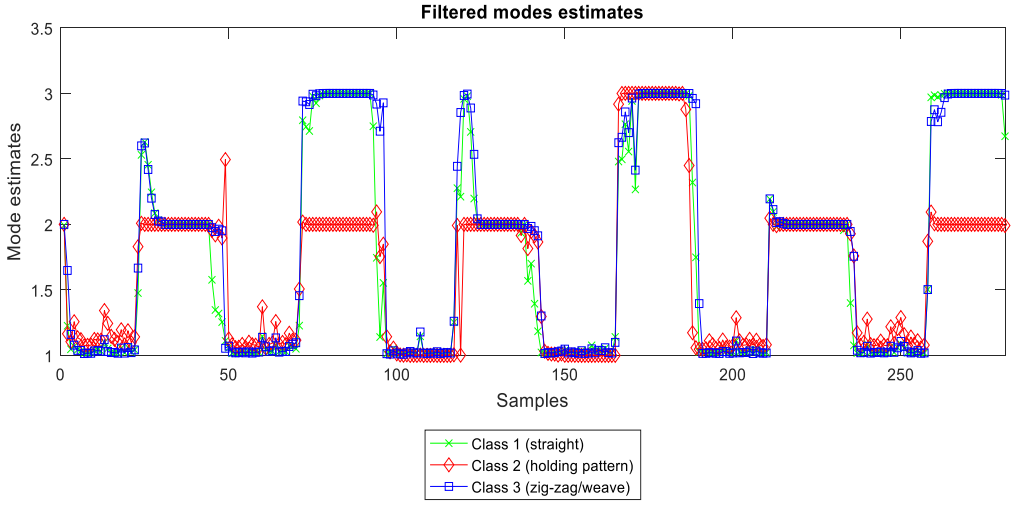


Fig. 5.7. Zig-zag pattern trajectory

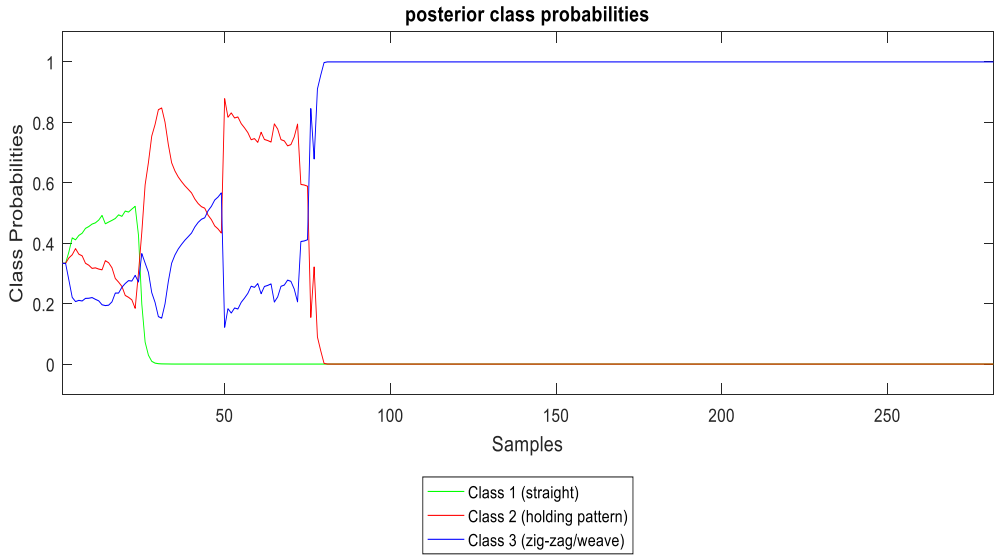
In this situation, we can now face a similar problem as before where one of the filters can diverge. In this situation since we have both left and right turns, the filter for the model corresponding to class 2 can be problematic. Since a holding pattern (as we have defined it in this work) only turns in one direction (left), the estimated probabilities for turning in the opposite direction (right), are either zero or extremely low. If the probability of switching into a turn mode is .005 for example, then with 15,000 particles that leaves on average only 75 particles to cover the mode switching. Again, we increase these probabilities slightly to avoid the filter from diverging. This is not an optimal solution, but for these simple "toy" examples, it does not have a drastic impact on performance. A better solution will be suggested in Chapter 6.

The posterior class probabilities and mode estimates are shown in Figure 5.8 below. This time we see more a decisive action from the classifier since the zig-zag trajectory is the only class that contains sustained right turns (mode 3). Initially starting in a straight heading, all three classes have a high probability of being the true class as we can see they all struggle against each other. As in the last example when going straight, class one has a higher probability since it has the highest weight of staying in mode 1 (π_{11}). As the target approaches the first turn, the holding pattern takes the lead as can be seen from the filtered mode estimates, class 1 and three overshoot the true mode (2), also causing a poor estimate of the true state (see trajectory above). One can easily see that on the second turn, class 2 fails to detect the turn, and likely had no particles in mode 3 (turn right), choosing instead mode two causing the posterior class probability to decline drastically and allowing the true class to take the lead. After two turns, we see that the zig-zag is correctly classified permanently with a high degree

of accuracy. From this, we could infer that classification performance is directly affected by how distinct the classes are from each other.



(a)



(b)

Fig. 5.8. System outputs for a zig-zag pattern trajectory. (a) filtered mode estimates (b) posterior class probabilities

5.2.3 Example 4: Holding Pattern With Two Added Sensors

In this example, a third sensor (radar) is incorporated at a position about one kilometer away from the original two sensors for a total of three sensors gathering measurements. The new location is contained in the sensor state $\overset{3}{x}^{*} = (200m, 1000m)$. The noise characteristics for this new sensor are $\sigma_{2,r}^2 = 10\ m, \sigma_{2,b}^2 = 5\ mrad\ , \sigma_{2,d}^2 = 3\ m/s$, for range, bearing and Doppler respectively. Notice these parameters are different from those of the first radar

sensor. Suppose now that a target is being observed at a distance in a holding pattern as shown in Figure 5.9. The same initial conditions from the previous two examples hold here.

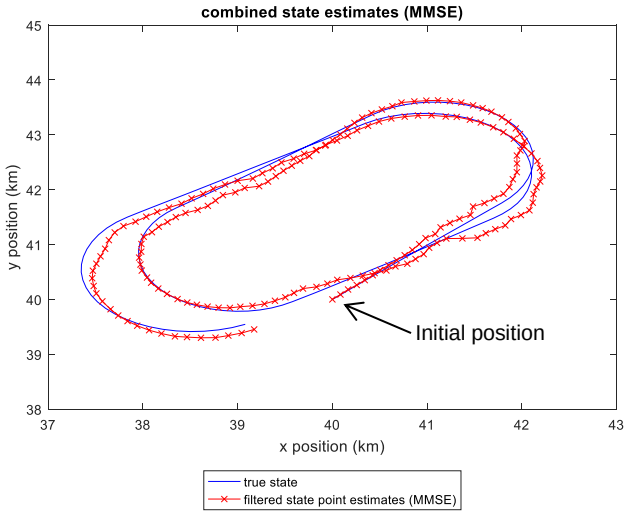
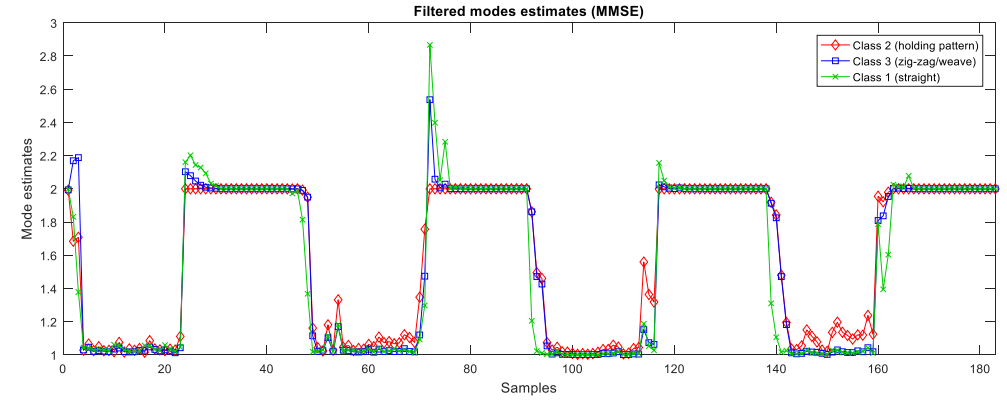
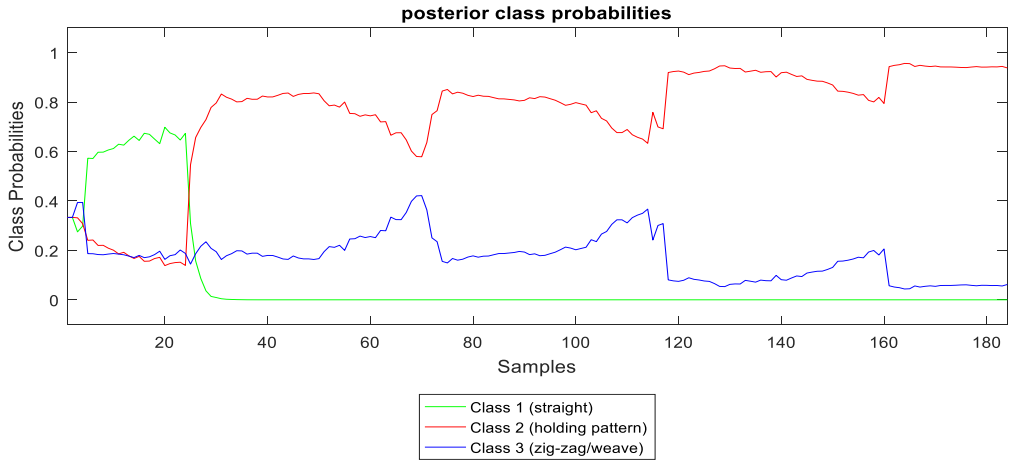


Fig. 5.9. Holding Pattern

As can be seen from Figure 5.9, the object in motion initially undergoes straight motion then performs a left turn, and this process repeats a few times so that it stays circling the same general area. Therefore we expect the system to classify the trajectory as a holding pattern which is indeed the case as shown in Figure 5.10b. The filtered mode estimates are also shown for convenience. If we look at the filtered mode sequences, it is easy to see that the output sequence from the filter for class two (shown in red) most closely follows the true trajectory, which never enters mode three (turn right). A closer look reveals that the mode estimates for class two are most accurate when the system is in mode two. This is be explained by the fact that during training, the holding patterns received a slightly lower probability of staying in straight motion of about .92, while class three had a higher estimated probability of about .98. Therefore, when in straight motion the two class probabilities moved towards each other, while when turning, they moved away from each other as $P(\varphi_{2|t}|y_{0:t})$ tended to one. Since the target started in straight motion, initially as expected $P(\varphi_{1|t}|y_{0:t})$ was the highest, but shortly tended to zero as the first turn took place, never coming back. An important note is that the class probabilities approach zero and are very small values but do not equal zero. Due to the recursive nature of (3.26), once the class probabilities are very low, it is difficult for them to rise again and care should be taken to address this issue in practice.



(a)



(b)

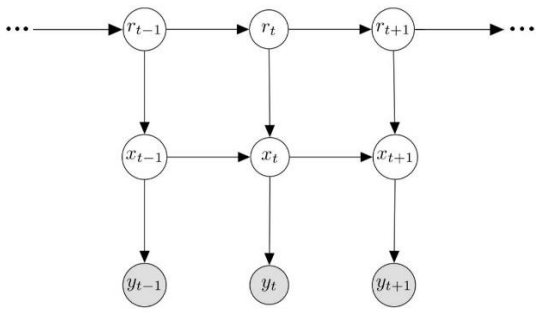
Fig. 5.10. System outputs for a holding pattern trajectory. (a) filtered mode estimates (b) class posterior probabilities.

6 Conclusions

6.1 Summary and Contributions

A novel method for classifying target behavior based on measurements from multiple sensors was proposed here. Put concisely, the purpose of this work was to investigate whether JMS were suitable for distinguishing trajectories, which in turn allows us to classify behavior. Training of hidden Markov models where the state space is discrete (DHMM) is well explored in the literature. As mentioned in Chapter 1, classification based on training of DHMMs in not new, and is widely used in many applications dating back to the 1970's. The first main contribution of this work can be illustrated through the use of dynamic Bayesian networks. The reader unfamiliar with these probabilistic graphical models is referred to [103] as an in-depth reference. The methods proposed in this thesis extend the classical methods used to train DHMMs shown in figure 6.1(a) to the Bayesian network shown in figure 6.1(b). These graphs give a high-level probabilistic representation of the relationship between the latent (hidden) variables and the observed data in HMMs. One can immediately see the difference by comparing this network to that of figure 6.1(a). The added continuous latent variable node in the graphical model called for the simultaneous sequential estimation of both the state variables x_t and the discrete mode r_t in the hybrid state space. This is in contrast to a DHMM where the unobserved process is solely composed of a discrete random variable in a finite state space. This problem was addressed via multiple model particle filters and smoothers.

(a) DHMM



(b) Hybrid system (JMS)

Fig. 6.1. Bayesian Networks

The second contribution of this thesis is that a novel method for jointly tracking and classifying dynamic object behavior using multi-sensor data was developed. It was shown in

the previous chapter that trajectories can indeed be recognized using a bank of particle filters, each corresponding to a specific class characterized by its unique transition matrix.

The third contribution of this thesis is the proposed solution addresses the requirement that the sensor system must be modular, in the sense that adding new sensors to the network should not require retraining or changes that deviate greatly from the original design. In the simulations, a scenario where two sensors were added, required simply making a slight adjustment to the measurement model and processing and did not require another training phase. This clearly met the goal of system modularity regarding sensor configuration.

Finally, the last contribution of this work is that simulations were carried out to validate these theoretical formulations. The simulations in section 5.1 showed that the techniques for system identification proved to be a suitable approach for estimating the parameters of dynamic stochastic systems with jumps. The simulations of the JTC scheme in section 5.2 showed promising results for classifying targets' special-temporal behavior based trajectory analysis from measurements from multiple sensors. In these examples, the proposed JTC scheme, using a Bayes classifier, quickly reacted to changes when the target changed its behavior. In the last example, simulations showed that the addition of an extra sensor required only the processing of additional measurements using a similar measurement model that had different noise characteristics. All of this culminates to one crucial point: JMNLS is a suitable class of models for trajectory recognition using multi-sensor data.

6.2 Future work and Improvements

A discussion of potential improvements will be given here. The following observations and suggestions can be made naturally by examining the assumptions and results from the previous chapter:

- In the simulations, only 3-4 models were used, and many simplifying assumptions were made. For starters, the turn rate omega is typically not known in practice and should be added to the state vector.
- The multiple-model bootstrap particle filter implemented is inefficient due to the suboptimal proposal distribution. This required a higher number of particles to avoid divergence, which drastically increases the number of floating-point operations. One approach to combat this issue would be to implement a more efficient particle filter which fixes the number of particles in each mode (see, e.g., [102]). Alternatively, one can take advantage of the structure of JMNLS proposed in [78], where the authors present an online EM algorithm Rao-Blackwellized particle filter to marginalize the mode out analytically and reduce estimation error variance.
- The measurement models used in simulations assumed a constant noise variance. In practice, the measurement noise will vary under different conditions. The range measurement, for example, will have a higher variance the farther the target is away from the radar, and this should be accounted for. The reason for this is that the farther away the target is from the radar, the longer the pulsed wave has to travel and the lower the received power at the receiver. This in turn decreases the SNR at the receiver, leading to less accurate measurements. In these low SNR environments, a track-before-detect scheme could be an alternative solution.

- Regarding the random error in the parameter estimates in section 5.1, due to the stochastic nature of SMC methods and the approximation of the complete data log-likelihood $Q(\theta, \theta_k)$, it is desirable to investigate a suitable stopping criterion for the EM algorithm. Although the stopping criterion chosen in this work was based on the work in [55] which showed reasonable performance, it is generally only been shown to be effective when $Q(\theta, \theta_k)$ can be calculated exactly. This could lead to poor performance in practice. Therefore, it is suggested for future work to investigate a more optimal stopping criterion (see, e.g., [42] for one such proposal).

Future work to follow could be aimed in many directions. The first one to discuss is computational efficiency. Recall one of the primary requirements of this work was to provide a solution that required relatively short training times. Here we showed that training JMS did not require large amounts of data which in itself can reduce training times, but we cannot overlook the fact that smoothing is a computationally expensive process. Therefore, in the training phase, the computation of the smoothed marginal densities, being the main culprit, would be the most impactful place to begin looking for improvements. Recall, that the FFBSm smoothing algorithm, which eradicates the path degeneracy problem entirely, has a convergence rate of $\mathcal{O}(NN_P^2)$, can be quite costly as the number of particles increases. One might naturally think to replace this smoother with a conventional fixed-lag smoother, but this will not perform as well regarding variance reduction. The FFBSi and PaRIS algorithms mentioned in 2.7 are two potential candidates. FFBSi having a complexity of $\mathcal{O}(N_P^2)$ would only have a marginal increase in performance. Following [64], this rate can be reduced to $\mathcal{O}(N_P)$, by employing an accept-reject approach, under the weak assumption that the transition kernel is uniformly bounded. The FFBSi although computationally cheaper than the FFBSm, comes at the cost of reduced accuracy (higher variance) due to the simulation of backward trajectories. The PaRIS algorithm [66] [104] on the other hand was shown to have the same complexity $\mathcal{O}(N_P)$ and performed as well as the FFBSm, with the advantage of being an online fixed-lag algorithm which reduces memory requirements. At the time of this writing, the author of this thesis proposed that PaRIS algorithm could be used to estimate the required smoothed marginal densities for the calculations of the expectations in the EM algorithm. Shortly before the completion of this thesis, it was discovered that authors in [79] did just that—they proposed an EM-based recursive ML estimation scheme for NLJMS using the PaRIS. The authors published their work during the timeframe of this master's thesis, which is why it was not found earlier. Their results showed near identical performance compared to the batch FFBSm smoothers with the exception of the transition probabilities which had a slight dependence on the number of backward samples needed.

Another area for potential expansion of this work would be to explore the adoption of semi-Markov models and stochastic context-free grammars (SCFG). Beginning with the former, recall from Chapter 1, a hidden semi-Markov models allow the unobservable process to be governed by a Markov renewal process, allowing for an arbitrary sojourn time. By contrast in a hidden Markov model, the sojourn time is exponentially distributed by construction. As mentioned, this has previously been exploited in [37] in a JTC scheme to classify targets based on their maneuverability. This behavior was characterized by the corresponding semi-Markov model the target was most likely to be following. Using this more general class of models for trajectory analysis could bring for some promising results since semi-Markov models have a more descriptive representation of the real underlying process.

Lastly, SCFGs were considered by the author as a potential direction for future work. One main advantage of SCFGs is that they can model more complex trajectories and they have a higher predictive capacity than HMMs. The latter is measured by a reduction in entropy (see Appendix C). The authors in [104] form a generalization where a multiple model particle filter can be used with SCFGs. Their aim in their paper is also the classify anomalous trajectories to aid a human operator, and they present two examples in radar tracking. They also improve their classification performance by combining SCFGs with a reciprocal Markov process (RP) model. The latter of which performed rather poorly on its own. There was even a mentioned of possibly using these methods on a network of sensors. The author of this thesis was not aware of this publication until nearing the end of this writing; hence this work was not mentioned in the related work section in Chapter 1.

References

[1] E. Blasch, C. Yang, I. Kadar, “Summary of Tracking and Identification Methods,” *Proc. SPIE*, Vol. 9119, 2014.

[2] R. Soleti, L. Cantini, F. Berizzi, A. Capria and D. Calugi, "Neural Network for polarimetric radar target classification," *2006 14th European Signal Processing Conference*, Florence, 2006, pp. 1-5.

[3] P. Zhang, L. Yang, G. Chen and G. Li, "Classification of drones based on micro-Doppler signatures with dual-band radar sensors," *2017 Progress in Electromagnetics Research Symposium - Fall (PIERS - FALL)*, Singapore, 2017, pp. 638-643.

[4] D. Habermann, E. Dranka, Y. Caceres and J. B. R. do Val, "Drones and helicopters classification using point clouds features from radar," *2018 IEEE Radar Conference (RadarConf18)*, Oklahoma City, OK, 2018, pp. 0246-0251.

[5] R. Gabler, W. Koch, “Detection and Tracking of Non-Cooperative Vessels,” *NATO Symposium on Port and Regional Maritime Security Proceedings*, Lerici, Italy, May, 2012 http://publica.fraunhofer.de/eprints/urn_nbn_de_0011-n-2171262.pdf, accessed in June 2018.

[6] ’Warning over drones use by terrorists.’ http://www.bbc.co.uk/news/uk-england-35280402, accessed in June 2018.

[7] S. Blackman and R. Popoli, *Design and analysis of Modern Tracking Systems*, Norwood, MA: Artech House, 1999.

[8] R. S. Mamon, R. J. Elliott, *Hidden Markov Models in Finance*, New York: Springer, 2007.

[9] E. Barucci, C. Fontana, *Financial Markets Theory*, London: Springer-Verlag, 2017.

[10] R.J. Elliott, T.K.Siu, “Option pricing and filtering with hidden Markov-modulated pure-jump processes,” *Applied Mathematical Finance*, vol. 20, no. 1, pp.1-25, 2013.

[11] A. Stuart. “Target Classification, Recognition and Identification with HF Radar,” https://www.sto.nato.int/publications/STO%20Meeting%20Proceedings/RTO-MP-SET-080/MP-SET-080-25.pdf , 2004, accessed in June 2018.

[12] Mahendra Mallick; Vikram Krishnamurthy; Ba-Ngu Vo, "Track-Before-Detect Techniques," in *Integrated Tracking, Classification, and Sensor Management:Theory and Applications* , 1, Wiley-IEEE Press, 2012, pp.768-

[13] Mahendra Mallick; Vikram Krishnamurthy; Ba-Ngu Vo, "Track-Before-Detect Techniques," in *Integrated Tracking, Classification, and Sensor Management:Theory and Applications* , 1, Wiley-IEEE Press, 2012, pp.768-

[14] D. Wang, F. He, S. Maslov, M. Gerstein, “DREISS: Using State-Space Models to Infer the Dynamics of Gene Expression Driven by External and Internal Regulatory Networks,” *PLoS Computational Biology*, vol. 12, no. 10, 2016.

[15] M. W. Pedersen, C. W. Berg, U. H. Thygesen, A. Nielsen, H. Madsen, “Estimation methods for nonlinear state-space models in ecology,” *Ecological Modeling*, vol. 222, no. 8, pp. 1394-1400, Apr. 2011.

[16] Y. Zeng, S. Wu, *State-Space Models: Applications in Econometrics and Finance*, New York: Springer Science & Business Media, 2013.

[17] G. Zhang and S. Godsill, "Fundamental Frequency Estimation in Speech Signals With Variable Rate Particle Filters," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 890-900, May 2016.

[18] Weiming Hu, Xuejuan Xiao, Zhouyu Fu, D. Xie, Tieniu Tan and S. Maybank, "A system for learning statistical motion patterns," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, Sept. 2006, pp. 1450-1464.

[19] J. C. Nascimento, M. A. T. Figueiredo and J. S. Marques, "Trajectory Classification Using Switched Dynamical Hidden Markov Models," in *IEEE Transactions on Image Processing*, vol. 19, no. 5, May 2010, pp. 1338-1348.

[20] J. Wen, C. Li, Z. Xiong, “Behavior pattern extraction by trajectory analysis.” in *Frontiers of Computer Science in China.* Vol. 5, no. 1, pp. 37-44, 2011.

[21] R. Fraile and S. J. Maybank, “Vehicle trajectory approximation and classification.” in *British Machine Vision Conference,* vol. 698, pp.702,1998.

[22] Corina Sas, Gregory O’Hare, and Ronan Reilly, *Online Trajectory Classification*, Berlin Heidelberg: Spriger, 2003.

[23] Y. Endo, H. Toda, K. Nishida, J. Ikedo, “Classifying spatial trajectories using representation learning.” in *International Journal of Data Science & Analytics.* Vol. 2, no.3-4, 2016, pp.107–117.

[24] A. Milan, H. Rezatofighi, A. Dick, I. Reid, K. Schindler, “Online Multi-Target Tracking Using Recurrent Neural Networks.” In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17).* 2017.

[25] H. Ghadaki and R. Dizaji, "Target track classification for airport surveillance radar (ASR)," *2006 IEEE Conference on Radar*, 2006, pp. 4 pp.-.

[26] L. E. Baum, T. Petrie, “Statistical inference for probabilistic functions of finite state Markov chains,” in *Annals of Mathematical Statistics*, vol. 37, no. 6, 1996, pp. 1554-1563.

[27] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains,” Annals of Mathematical Statistics, vol. 41, no. 1, 1970, pp. 164-171.

[28] L. E. Baum, “An inequalityand associated maximization technique in statistical estimation for probabilistic functions of Markov processes,” *Inequalities*, vol. 3, 1972, pp. 1-8.

[29] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, Feb 1989.

[30] T. Plötz, G. A. Fink, “Markov models for offline handwriting recognition: a survey.” in *International Journal on Document Analysis and Recognition (IJDAR).* 2009. pp. 269-298.

[31] M.J. Bishop, E.A Thompson. "Maximum likelihood alignment of DNA sequences". in Journal of Molecular Biology. vol. 190, no. 2, pp 159–65, Jul. 1986.

[32] R. Durbin, S. R. Eddy, S. Eddy, A. Krogh, G. *Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[33] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," in *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174-188, Feb 2002.

[34] J.M. Morales,; D.T Haydon, J. Frair, K. E Holsinger, J.M Fryxell, "Extracting more out of relocation data: building movement models as mixtures of random walks" *EEB Articles*. 4. 2004.

[35] H. L. Beyer, MJ Fortin, J. M. Morales. D. Murray, “The Effectiveness of Bayesian state-space models for estimating behavioural states from movement paths.” In *Methods in Ecology and Evolution.* Vol. 4, no. 5, 2013, pp. 413-441.

[36] S. Kim, R. Zbikowski, A. Tsourdos, B.A. White, “Behaviour recognition of ground vehicle for airborne monitoring of unmanned aerial vehicles” in *International Journal of Systems Science*. Vol. 45, no. 12, 2014, pp. 2499 -2514.

[37] S. Maskell, “Joint tracking of manoeuvring targets and classification of their manoeuvrability.” in *EURASIP Journal on Advances in Signal Processing*, 2004, pp. 2339–2350.

[38] H. A. P. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," in *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780-783, Aug 1988.

[39] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part V. Multiple-model methods," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1255-1321, Oct. 2005.

[40] R.R. Pitre, V. P. Jilkov, X. R. Li, “A comparative study of multiple-model algorithms for maneuvering target tracking” Proceedings of the SPIE, vol. 5809, p. 549-560, 2005.
[41] W. Shu, Z. Zheng, “Performance Analysis of Kalman-Based Filters and Particle Filters for non-linear/non-gaussian Bayesian Tracking,” in *IFAC Proceedings Volumes,* vol. 38, no.1, pp. 1131-1136, 2005.

[42] T. T. Ashley and S. B. Andersson, "A Sequential Monte Carlo framework for the system identification of jump Markov state space models," *2014 American Control Conference*, Portland, OR, 2014, pp. 1144-1149.

[43] C. Fritsche, E. Özkan and F. Gustafsson, "Online EM algorithm for jump Markov systems," *2012 15th International Conference on Information Fusion*, Singapore, 2012, pp. 1941-1946.

[44] N. J. Gordon, D. J. Salmond and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107-113, April 1993.

[45] R.H Shumway, and D.S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. 3rd ed. New York: Springer, 2011.

[46] S. N. Durlauf, L. E. Blume, *Macroeconometrics and Time Series Analysis*, New York: Macmillan Publishers, 2010.

[47] S. M. Kay, *Fundamentals of Statistical Signal Processing vol. 1: Estimation Theory*, Upper Saddle River, NJ: Prentice-Hall, Inc., 1993.

[48] J. V. Candy. *Bayesian Signal Processing: Classical Modern and Particle Filtering Methods*. Hoboken NJ USA: Wiley/IEEE Press 2009.

[49] G. A. Young, *Essentials of Statistical Inference*, Cambridge University Press, 2005.

[50] D. Panchenko, “Lecture 3: Properties of Maximum Likelihood Estimators,” in *Statistics for Applications (MIT course number: 18.650),* 2006,

[51] The Analytic Sciences Corporation , Arthur Gelb, *Applied Optimal Estimation*, The MIT Press, 1974

[52] E. L. Lehmann, G. Casella, *Theory of Point Estimation (2nd ed.).* New York: Springer, 1998.

[53] Z. Chen, "Bayesian filtering: From Kalman filters to particle filters and beyond" *Statistics* vol. 182 no. 1 pp. 1-69 2003.

[54] D. Panchenko, Lecture notes, Topic: “Statistics for Applications” 18.650, Faculty of Mathematics, MIT, 2006, https://ocw.mit.edu/courses/mathematics/18-443-statistics-for-applications-fall-2006/lecture-notes/lecture3.pdf, accessed May 2018.

[55] R. P. S. Mahler, *Statistical Multisource-Multitarget Information Fusion*, Norwood MA: Artech House, 2007.

[56] C. M. *Bishop, Pattern Recognition and Machine Learning, New York: Springer, 2006.*

[57] A. Doucet, S. J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. In *Statistics and Computing*, vol. 10, no. 3 pp.197–208, 2000.

[58] A. Doucet, J. F. G. de Freitas, N. J. Gordon, A. Doucet, J. F. G. de Freitas, N. J. Gordon, "An introduction to sequential Monte Carlo methods" in *Sequential Monte Carlo Methods in Practice*, New York: Springer-Verlag, 2001.

[59] F. Lindsten; T.B. Schön, "Backward Simulation Methods for Monte Carlo Statistical Inference," in *Backward Simulation Methods for Monte Carlo Statistical Inference* , Now Foundations and Trends, vol. 6, no. 1, pp. 1-143, 2013.

[60] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[61] M. Isard and A. Blake,"Condensation–conditional density propagation for visual tracking," *International. Journal of Computer Vision*, vol. 29, no.1, pp. 5–28, 1998.

[62] A. Doucet, "On sequential Monte Carlo methods for Bayesian filtering," Dept. Eng., Univ. Cambridge, UK, Tech. Rep., 1998.

[63] M. Hürzeler, H. R. Künsch, "Monte Carlo approximations for general state-space models," Journal of Computational and Graphical Statistics vol. 7, no. 2, pp. 175–193. 1998.

[64] R. Douc, A. Garivier, E. Moulines "Sequential Monte Carlo Smoothing for General State Space hidden Markov Models" in *The Annals of Applied Probability*, Vol. 21 No. 6, Dec. 2011., pp. 2019-2145.

[65] M. Briers, A. Doucet, and S. R. Maskell. "Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics* ," in *Annals of the institue of Mathematical Statistics*, vol 62. pp. 61-89, 2010.

[66] J. Westerborn and J. Olsson, "Efficient particle-based online smoothing in general hidden Markov models," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, 2014, pp. 8003-8007.

[67] R.H. Shumway, D. Stoffer. "An Approach to Time Series Smoothing and Forecasting Using the EM Algorithm." in *Journal of Time Series Analysis.* Vol. 3, no. 4, Jul. 1982, pp. 253-264.

[68] V. P. Jilkov, X. R. Li, "Adaptation of Transition Probability Matrix for Multiple Model Estimators," *Proceedings of 4th Annual Conference on Information Fusion*, Montreal, Aug. 2006.

[69] A. Doucet and B. Ristic, "Recursive state estimation for multiple switching models with unknown transition probabilities," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 1098-1104, Jul 2002.

[70] U. Orguner and M. Demirekler, "Maximum Likelihood Estimation of Transition Probabilities of Jump Markov Linear Systems," in *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5093-5108, Oct. 2008.

[71] A. Doucet, N. J. Gordon and V. Krishnamurthy, "Particle filters for state estimation of jump Markov linear systems," in *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 613-624, Mar 2001.

[72] Y. Boers and J. N. Driessen, "Interacting multiple model particle filter," in *IEE Proceedings - Radar, Sonar and Navigation*, vol. 150, no. 5, pp. 344-349, 2 Oct. 2003.

[73] C. Andrieu and A. Doucet, "Online expectation-maximization type algorithms for parameter estimation in general state space models," *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 2003, pp. VI-69-72 vol.6.

[74] T. B. Schön, A. Wills, B. Ninness, "Parameter Estimation for Discrete-Time Nonlinear Systems Using EM Algorithm," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 4012-4017, 2008.

[75] T. B. Schön, A. Wills, B. Ninness, "Maximum Likelihood Nonlinear System Estimation," *IFAC Proceedings Volumes*, vol. 39, no. 1, pp. 1003-1008, 2006.

[76] O. Cappe, "Online sequential Monte Carlo EM algorithm," *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, Cardiff, 2009, pp. 37-40.

[77] A. Doucet, V. B. Tadić, "Parameter Estimation in General State-Space Models using particle methods," *Annals of the institute of Statistical Mathematics*, vol. 55, no. 2, pp. 409-422, Jun. 2003.

[78] E. Özkan, F. Lindsten, C. Fritsche and F. Gustafsson, "Recursive Maximum Likelihood Identification of Jump Markov Nonlinear Systems," in *IEEE Transactions on Signal Processing*, vol. 63, no. 3, pp. 754-765, Feb.1, 2015.

[79] A. R. Braga, C. Fritsche, F. Gustafsson and M. G. S. Bruno, "Rapid system identification for jump Markov non-linear systems," *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Curacao, 2017, pp. 1-5.

[80] R. Braga, C. Fritsche, F. Gustafsson and M. G. S. Bruno, "Gradient-based recursive maximum likelihood identification of Jump Markov Non-Linear Systems," *2017 20th International Conference on Information Fusion (Fusion)*, Xi'an, 2017, pp. 1-7.

[81] C. M. Carvalho, H. F. Lopes, "Simulation-based sequential analysis of Markov switching stochastic volatility models," *Computational Statistics and Data Analysis*, vol. 51, no. 9, May 2007, pp- 4526-4542.

[82] R. S. Mamon, R. J. Elliott, Hidden Markov *Models in Finance, Further Developments and Applications, Volume II*, New York: Springer, 2014.

[83] P. Dreesen, K. Tiels, M. Ishteva and J. Schoukens, "Nonlinear system identification: Finding structure in nonlinear black-box models," *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Curacao, 2017, pp. 1-4.

[84] O. Capp´e, E. Moulines, and T. Ryden, Inference in Hidden Markov Models (Springer Series in Statistics). Secaucus, NJ, USA: Springer Verlag New York, Inc., 2005.

[85] Ng S.K., Krishnan T., McLachlan G.J. "The EM Algorithm." In *Handbook of Computational Statistics, Springer Handbooks of Computational Statistics.* Gentle J., Härdle W., Mori Y. (eds) Berlin, Heidelberg: Springer, 2012.

[86] A. P. Dempster, N. M. Laird, D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm." in *Journal of the Roayal Statistical Society. Series B (Methodological).* vol. 39, no. 1, 1977, pp. 1-38.

[87] S. McGinnity and G. Irwin, "Multiple model estimation using the bootstrap filter," *IEE Colloquium on Target Tracking and Data Fusion (Digest No. 1998/282)*, Birmingham, 1998, pp. 3/1-3/3.

[88] S. McGinnity and G. W. Irwin, "Multiple model bootstrap filter for maneuvering target tracking," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, Jul 2000, pp. 1006-1012.

[89] C. Kreucher A. O. Hero K. Kastella "Multiple model particle filtering for multi-target tracking" in *Proceedings of the Twelfth Annual Workshop on Adaptive Sensor Array Processing*, Mar. 2004.

[90] M. Ekman and E. Sviestins, "Multiple model algorithm based on particle filters for ground target tracking," *2007 10th International Conference on Information Fusion*, Quebec, Que., 2007, pp. 1-8.

[91] De-Ping Yuan and Juan-Yi Zheng, "Interacting multiple model target tracking algorithm based on particle filtering," *Proceedings of 2011 IEEE CIE International Conference on Radar*, Chengdu, 2011, pp. 1907-1910.

[92] A. Doucet, A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. http://www.cs.ubc.ca/_arnaud/doucet johansen tutorialPF.pdf, 2008.

[93] N. Kantas, A. Doucet, S. Singh, J. Maciejowski, N. Chopin, "Particle methods for parameter estimation in state-space models" in *Statistical Science,* vol. 30, no. 3. Pp- 328-351, 2015.

[94] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge UK: Cambridge University Press, 2009.

[95] N.Gordon, S.Maskell and T.Kirubarajan, *Efficient particle filters for joint tracking and classification, Proceedings of SPIE: Signal and Data Processing of Small Targets*, vol. 4728, pp 439-449, 2002

[96] D. Angelova, L. Mihaylova, “Sequential Monte Carlo algorithms for joint target tracking and classification using kinematic radar information”, in: *Proceedings of the Seventh International Conference on Information Fusion*, Stockholm, Sweden, 2004, pp. 709–716.

[97] Y. Boers, H. Driessen and A. Bagchi, "Point estimation for jump Markov systems: Various MAP estimators," *2009 12th International Conference on Information Fusion*, Seattle, WA, 2009, pp. 33-40.

[98] S. Saha, Y. Boers, H. Driessen, P. K. Mandal and A. Bagchi, "Particle based MAP state estimation: A comparison," *2009 12th International Conference on Information Fusion*, Seattle, WA, 2009, pp. 278-283.

[99] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333-1364, Oct. 2003.

[100] M. A. Richards, J. A. Scheer, W. A. Holm, *Principles of Modern Radar vol 1: Basic Principles*, Raleigh, NC: Scitech Publishing, 2010.

[101] X. Lin, T. Kirubarajan, Y. Bar-Shalom, S. Maskell, “Comparison of EKF, pseudomeasurement, and particle filters for a bearing-only tracking problem,” *SPIE Proceedings*, vol. 4728, 2002.

[102] H. Driessen and Y. Boers, "An efficient particle filter for jump Markov nonlinear systems," *IEE Target Tracking 2004: Algorithms and Applications*, 2004, pp. 19-22.

[103] D. Koller, N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, Cambridge, Massachusetts, London, England: The MIT Press, 2009.

[104] J. Olsson, J. Westerborn, “Efficient Parameter Inference in General Hidden Markov Models Using the Filter Derivatives,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 3984–3988.

[105] M. Fanaswala and V. Krishnamurthy, "Detection of Anomalous Trajectory Patterns in Target Tracking via Stochastic Context-Free Grammars and Reciprocal Process Models," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 76-90, Feb. 2013.

Appendices

Appendix A
Analysis of Sequential Monte Carlo Approximations

The Kalman filter is one of the most widely used algorithms for state estimation in target tracking, data fusion, control theory, econometrics, aerospace, and telecommunications to name a few. Examples include tracking moving targets in aerospace applications, to the dynamic estimation of hedging ratios between two equities which is common in many statistical arbitrage strategies in finance. In this section, we will show the reader that the particle filter, can provide close to optimal results if enough particles are used. These examples will show the correctness of the implementation of the SIR particle filter used in all the simulations of this thesis.

A closed form solution for a state space model only exists when either the system is linear and Gaussian, or when the state space of the hidden Markov chain is finite. Consider a Gaussian DLM where the future state for time $t \in \{1, \dots, n\}$ is described by

$$x_{t+1} = \Phi x_t + v_t \tag{A. 1a}$$
$$y_t = A x_t + e_t. \tag{A. 1b}$$

The Kalman filter aims at producing a sequence of state estimates and an associated error covariance estimates. In the case of a Gaussian DLM where the dynamic and measurement models accurately describe the system, the Kalman filter is the optimal Bayesian estimator in the mean square sense [51]. Furthermore, it has other optimal properties such as the fact that it is an MVU estimator. As an alternative view, it can also be seen and derived as a recursive least squares estimator. All of these properties make it a great candidate to use as a reference to measure the accuracy of the particle filter performance. The following notation will be used in the rest of this section:

$$x_t^s = \mathbb{E}[x_t | y_{0:s}] \tag{A. 2}$$

$$P_t^s = \mathbb{E}[(x_t - x_t^s)(x_t - x_t^s)^T] \tag{A. 3}$$

Assume that $v_t \sim \mathcal{N}(0, Q)$, $e_t \sim \mathcal{N}(0, R)$, and initial conditions $x_0^0 = \mu_0$ and $P_0^0 = \Sigma_0$. Then the Kalman equations for $t = \{1, \dots, n\}$ are

$$x_t^{t-1} = \Phi x_{t-1}^{t-1} \tag{A. 4}$$

$$P_t^{t-1} = \Phi P_{t-1}^{t-1}\Phi' + Q \tag{A.5}$$

$$x_t^t = x_t^{t-1} + K_t(y_t - A_t x_t^{t-1}) \tag{A.6}$$

$$P_t^t = [I - K_t A_t]P_t^{t-1} \tag{A.7}$$

$$K_t = P_t^{t-1}A_t'[A_t P_t^{t-1}A_t' + R]^{-1} \tag{A.8}$$

Together, (A.4) and (A.5) are the prediction stage, since they forecast the future state and associated covariance matrix before the measurements are received. The update stage is comprised of the state update (A.6) and covariance update (A.7) once the new measurements are received, where K_t is called the *Kalman gain matrix.* We also include the *innovations (prediction error)* and the corresponding *innovation-covariance* matrices

$$\epsilon_t = y_t - \mathbb{E}[y_t|Y_{0:t-1}] = y_t - A_t x_t^{t-1} \tag{A.9}$$

$$\Sigma_t \triangleq var(\epsilon_t) = var[A_t(x_t - x_t^{t-1}) + v_t] = A_t P_t^{t-1}A_t' + R \tag{A.10}$$

for $t = 1, \dots, n$. It is important to note that the equations in this section will still hold for the time-varying case where the system matrices and covariance matrices are all time dependent, provided the appropriate substitutions are made, see [45] for more details.

The Kalman filter can also be extended for smoothing [51] by the following equations for $t = \{n, n-1, \dots, 1\}$:

$$x_{t-1}^n = x_{t-1}^{t-1} + J_{t-1}(x_t^n - x_t^{t-1}) \tag{A.11}$$

$$P_{t-1}^n = P_{t-1}^{t-1} + J_{t-1}(P_t^n - P_t^{t-1})\, J_{t-1}^T \tag{A.12}$$

where,

$$\begin{aligned} J_{t-1} &= cov(x_{t-1}, x_t - x_{t-1}^n)[P_t^{t-1}]^{-1} \\ &= P_{t-1}^{t-1}\Phi^{\mathrm{T}}[P_t^{t-1}]^{-1} \end{aligned} \tag{A.13}$$

Example A.1

In this example, a comparison of the performance of the Kalman optimal state estimator against the boothstrap particle filter on a coordinated turn model for 100 and 500 particles N_p will be examined. Resampling will be done every time step. A sampling period of $T = .5\ s$ will be used. Suppose that in this example we are trying to track a speedboat under the presence of sea clutter moving in circles at about 140 km/h with a turn-rate of $\omega = .20\ rad/s$. The process noise variance for the coordinated turn model is $\sigma_\omega^2 = 7$. The measurement noise for the linear model was set to $\Xi = diag(3\,m/s, 3\ m/s^2, 3\ m/s, 3\,m/s^2)$. The object is first detected at a distance of approximately 2.8 km away heading northeast. The sensor tracking the object is assumed to

be on a vessel at the origin. Below in Figure A.1, we can see the trajectory and particle clouds.

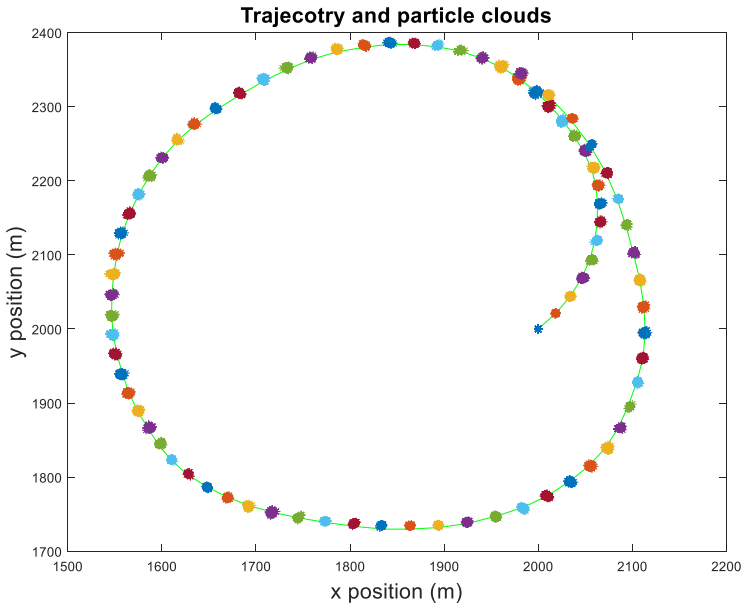


Fig. A.1. Trajectory of speed boat going in circular motion with particle clouds.

Now if zooming in to a small area and plot the variance of both the Kalman and Particle state estimators, it will give a good picture of how they compare. For the following simulations, an MMSE estimator was used to approximate the state. In Figure A.2 we see the variance of both the Kalman filter and smoother. As expected the smoother (seen in green) has a much smaller variance, and the smoothed state estimate is much closer to the true value.

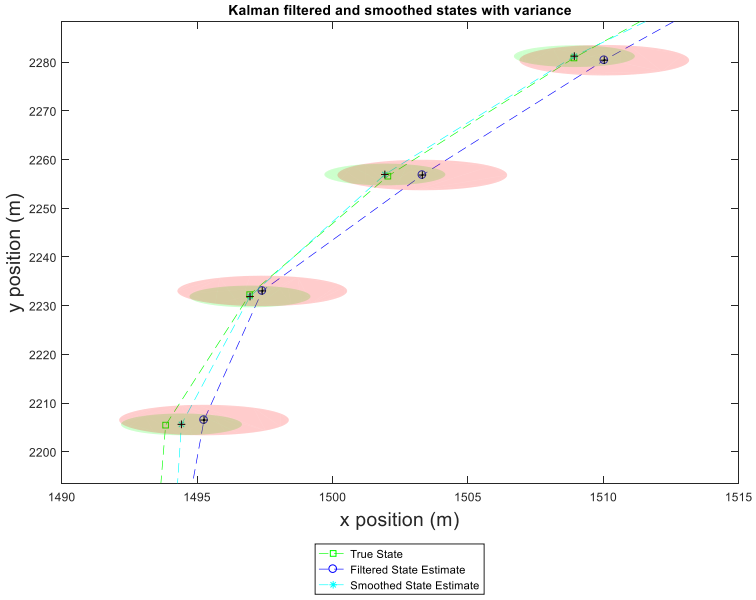


Fig. A.2. Kalman filter variance plots.

Now in Figure A.3, we can see a similar plot showing the results for a particle filter running with only 100 particles on the same data. Here it is evident that the covariance ellipses are skewed or tilted, especially the ones for the smoothed data. Also on some of the state estimates, neither the filtered nor the smoothed variances cover the true state. The smoothed estimate is still more accurate than the filtered estimate but it's variance does not cover the true state. These observations are due to particle depletion issues. There are just not enough particles to cover the true posterior leading to a poor estimate.

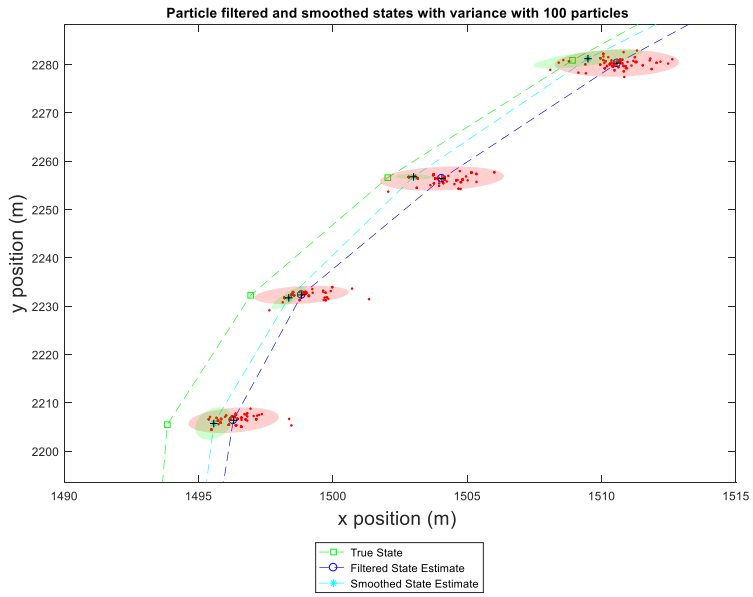


Fig. A.3. Particle filter and smoother variance plots using 100 particles.

If we increase the number of particles to 500, we can see below in Figure A.4 that we get a much better result. Now both the filtered and smoothed variances cover the actual state. The results look quite similar to those of the Kalman filter.

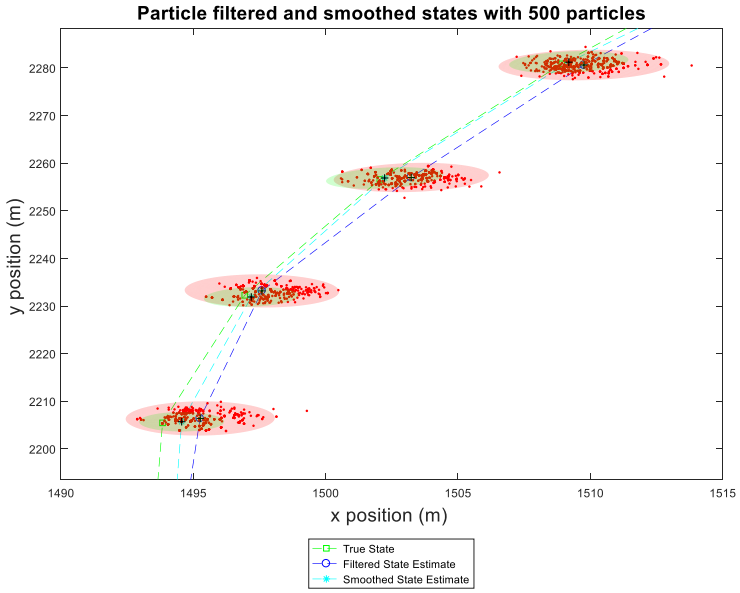


Fig. A.4. Particle filter and smoother variance plots using 500 particles.

Now we turn to the root mean square error (RMSE) to quantify performance and obtain a clear metric as opposed to visually inspecting. Beginning with the RMSE for each time step t:

$$RMSE(t) = \sqrt{\frac{1}{N_{mc}} \sum_{t=1}^{T} \left(\hat{x}_t^j - x_t^j\right)^2} \tag{A.14}$$

where N_{MC} is the number of Monte Carlo runs, and $\hat{x}_t^j$ and x_t^j are the estimated and true states respectively. We can plot the error at ever time step as seen in Figure A.5. For this simulation, 500 particles were used. A close look will reveal that performance is as expected, with the Kalman filter performing the best, and the particle filter giving the highest RMSE on most samples.

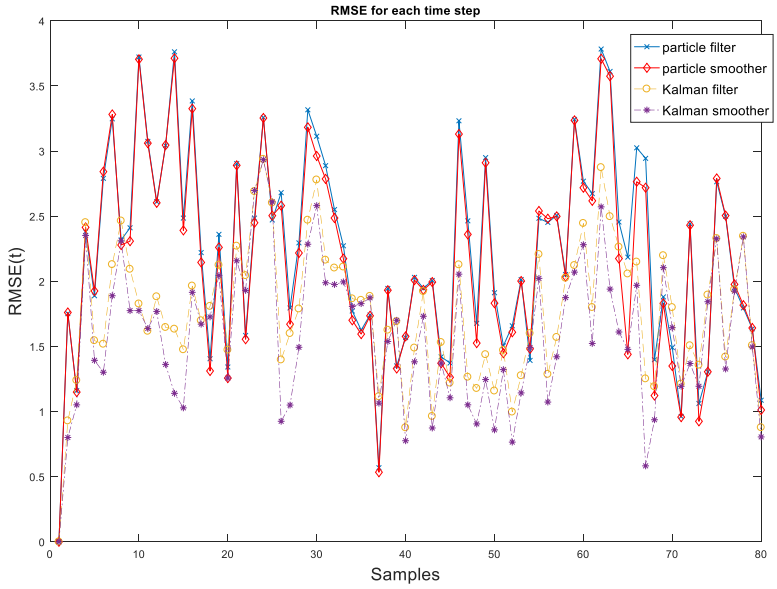


Fig. A.5. RMSE for all filters at each times step.

As a last test, we define the *total* RMSE [41] as:

$$RMSE_{total} = \sqrt{\frac{1}{T} \sum_{t=1}^{N} \frac{1}{N_{mc}} \sum_{j=1}^{N_{mc}} \left(\hat{x}_t^j - x_t^j\right)^2} \tag{A.15}$$

This is a measure of the overall RMSE for the entire data set. Blow in Figure A.6 we can see the results for a varying number of particles starting from 50 all the way to 500 in increments of 50 particles. The Kalman filter and smoother results are plotted for reference. Here 25 MC runs were executed. It is evident that as the number of particles increased, the SMC approximations converge towards the optimal (Kalman) estimates for both the filter and smoother.

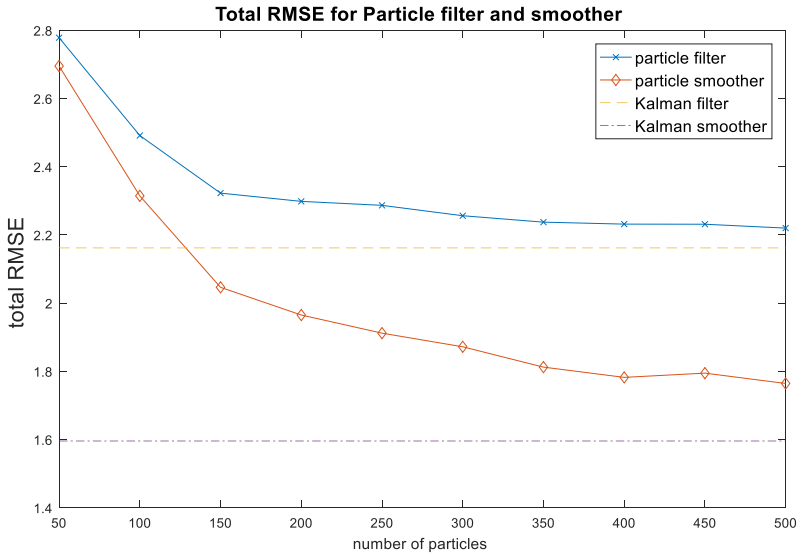


Fig. A.6. Total RMSE using 25 Monte Carlos runs for the particle filter and smoother. The Kalman filter and smoother are shown for comparison.

Appendix B
Additional Simulations and Results

In this section, we present the results of a parameter learning example using a nonlinear measurement model. We will repeat the example in section 5.2 for a zig-zag trajectory but instead, use and process data from three sensors. The nosie characteristics and the sensor states (positions) $\overset{j}{x}^* = (x^s, y^s)$ for each sensor are listed in Table B.1. For this simulation, 600 particles were used. We repeat the same example as in section 5.1 and assume the same dynamic models and process nose parameters are used.

Table 6.1. Sensor parameters

Sensor	Range (m)	Doppler (m/s)	Bearing (mrad)	x^* (m)
1 (Radar)	15	5	10	(0,0)
2 (Radar)	10	3	5	(200,1000)
3 (Optical)	N/A	N/A	1	(0,0)

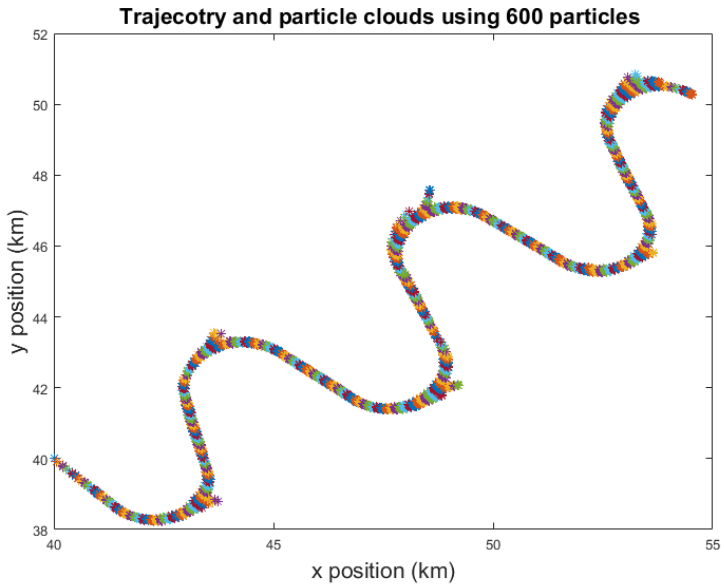


Fig. B.1. Zig-zag trajectory tracking by particle filter with particle clouds shown.

In Figure B.1 we can see already that some of the particles deviate from the real trajectory, already giving a hint that the state estimates are not as accurate. Below in figure B.2, we can see the process noise parameters perform moderately well with slightly less accuracy on the CV model noise. It is essential to keep in mind that an additional 250 particles had to be used to achieve similar accuracy. This drastically increased the amount of memory needed for the smoothing recursions and the use of additional particles resulted in MALAB running out of storage on a computer with 8GB of RAM.

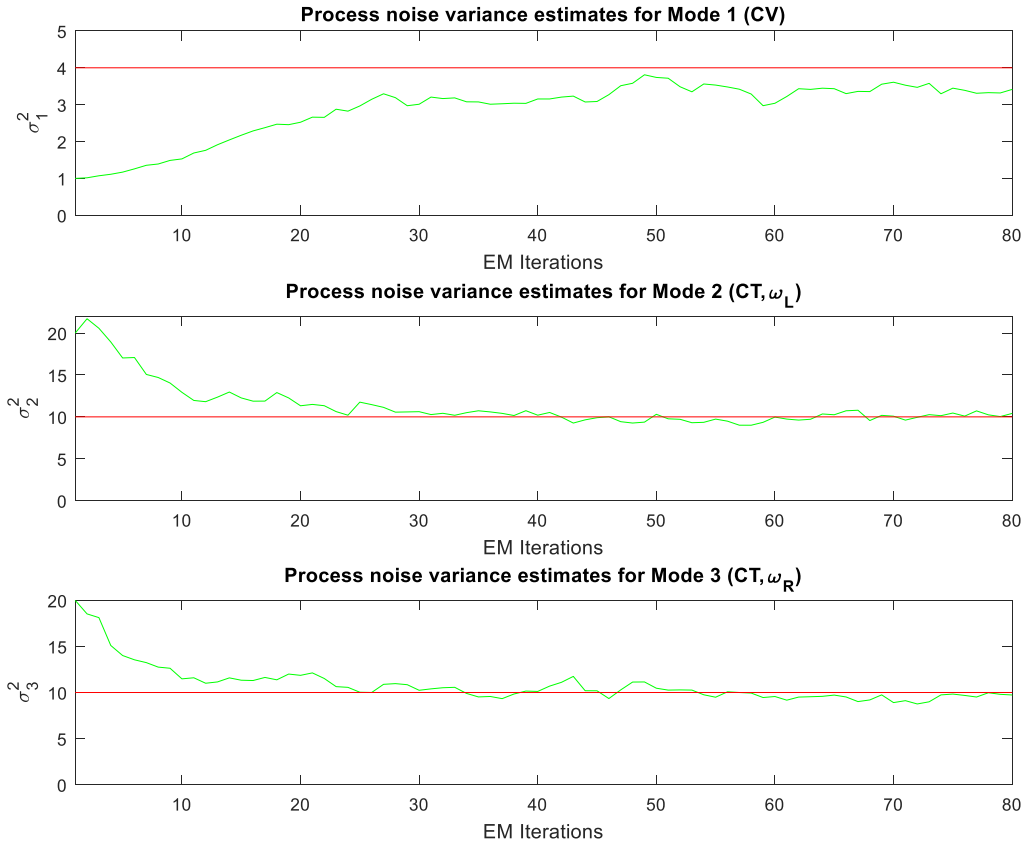


Fig. B.2. Process noise estimates for 80 EM iterations using nonlinear measurement model and three sensors.

The estimates of the transition probabilities seen in figure B.3 are also on par with the simulations from section 5.1 except with a higher variance. Is should now be clear that Algorithm 7 is capable of estimating systems parameters with nonlinear measurement models. The limitations here are due to the reasons discussed in Chapter 6 involving a suboptimal particle filter.

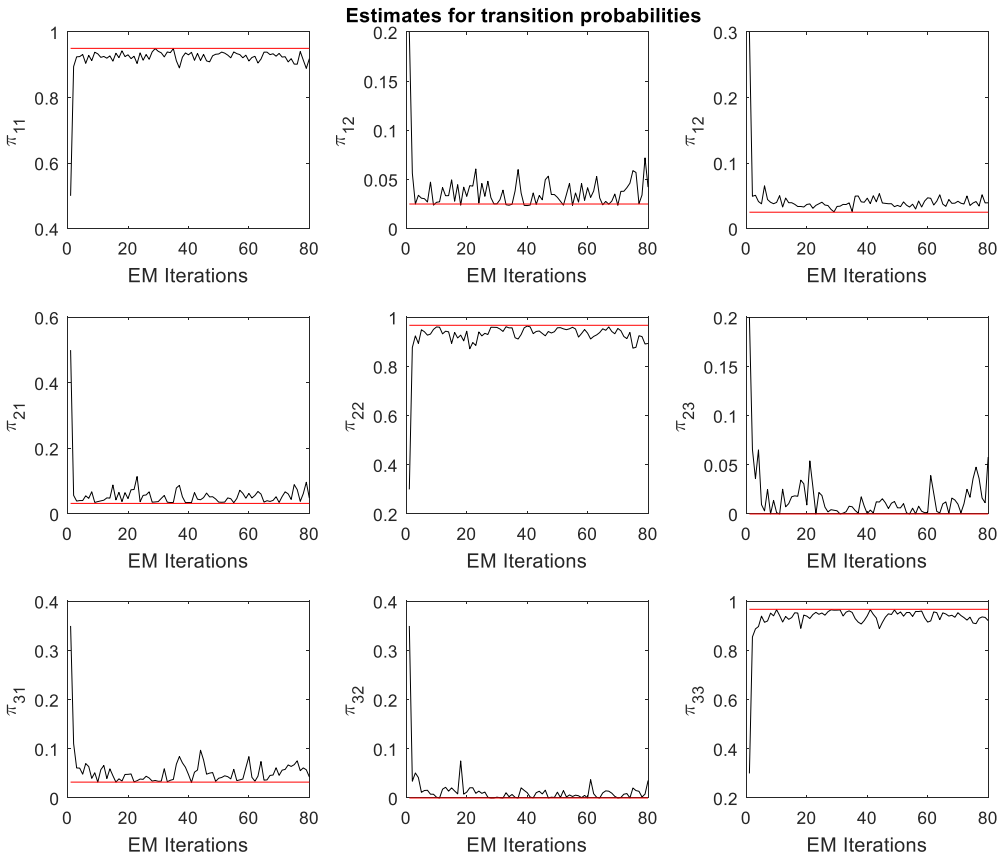


Fig. B.3. Estimates of transition probabilities for 80 EM iterations using nonlinear measurement model and three sensors.

Appendix C
Proofs

C.1 Multiple Model fixed-interval FFBSm

Recall that the goal of SMC smoothing using the FFBSm algorithm in the case of hybrid systems is to obtain an approximation of the marginal posterior density

$$p_\theta(x_t, r_t|y_{0:N}) \approx \hat{p}_\theta^{N_p}(x_t, r_t|y_{0:N}) \triangleq \sum_{i=1}^{N_p} w_{t|N}^i \, \delta\left(x_t - x_t^i\right)\mathbb{1}\left(r_t = r_t^i\right) \tag{C1.1}$$

where $w_{t|N}^i$ are the smoothed weights. Suppose N data samples are available. To derive an expression for these weights, we build upon the work done in [60] to include a discrete mode r_t.

Proof:

With the use the of the definition of conditional probability:

$$p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:N}) = p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:t}, y_{t+1:N})$$

$$= \frac{p_\theta(x_t, x_{t+1}, r_{t+1}, y_{0:t}, y_{t+1:N})}{p_\theta(x_{t+1}, r_{t+1}, y_{0:t}, y_{t+1:N})}$$

$$= \frac{p_\theta(y_{t+1:N}|x_t, x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_t, x_{t+1}, r_{t+1}, y_{0:t})}{p_\theta(x_{t+1}, r_{t+1}, y_{0:t}, y_{t+1:N})}$$

$$= \frac{p_\theta(y_{t+1:N}|x_t, x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_{t+1}, r_{t+1}, y_{0:t})}{p_\theta(x_{t+1}, r_{t+1}, y_{0:t}, y_{t+1:N})}$$

$$= \frac{p_\theta(y_{t+1:N}|x_t, x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_{t+1}, r_{t+1}, y_{0:t})}{p_\theta(y_{t+1:N}|x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_{t+1}, r_{t+1}, y_{0:t})}$$

$$= \frac{p_\theta(y_{t+1:N}|x_t, x_{t+1}, r_{t+1}, y_{0:t})p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:t})}{p_\theta(y_{t+1:N}|x_{t+1}, r_{t+1}, y_{0:t})} \tag{C1.2}$$

Given the Markov property of 3.1a-3.1b then:

$$p_\theta(y_{t+1:N}|x_t, x_{t+1}, r_{t+1}, y_{0:t}) = p_\theta(y_{t+1:N}|x_{t+1}, r_{t+1}, y_{0:t}) \tag{C1.3}$$

Applying this to C1.2, we arrive at:

$$p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:N}) = p_\theta(x_t|x_{t+1}, r_{t+1}, y_{0:t}). \tag{C1.4}$$

Next, we apply Bayes' theorem and the law of total probability to $p_\theta(x_t, r_t|y_{0:N})$:

$$\begin{aligned} p_\theta(x_t, r_t|y_{0:N}) &= \sum_{r_{t+1}\in\mathcal{S}} \int_{\mathcal{X}} p_\theta(x_t, r_t|x_{t+1}, r_{t+1}, y_{0:t}) p_\theta(x_{t+1}, r_{t+1}|y_{0:N}) dx_{t+1} \\ &= \sum_{r_{t+1}\in\mathcal{S}} \int_{\mathcal{X}} \frac{p_\theta(x_{t+1}, r_{t+1}|x_t, r_t) p_\theta(x_t, r_t|y_{0:t})}{p_\theta(x_{t+1}, r_{t+1}|y_{0:t})} p_\theta(x_{t+1}, r_{t+1}|y_{0:N}) dx_{t+1} \\ &= p_\theta(x_t, r_t|y_{0:t}) \sum_{r_{t+1}\in\mathcal{S}} \int_{\mathcal{X}} \frac{p_\theta(x_{t+1}, r_{t+1}|x_t, r_t) p_\theta(x_{t+1}, r_{t+1}|y_{0:N})}{p_\theta(x_{t+1}, r_{t+1}|y_{0:t})} dx_{t+1} \\ &= p_\theta(x_t, r_t|y_{0:t}) \sum_{r_{t+1}\in\mathcal{S}} \int_{\mathcal{X}} \frac{p_\theta(x_{t+1}|x_t, r_{t+1}) p_\theta(r_{t+1}|r_t) p_\theta(x_{t+1}, r_{t+1}|y_{0:N})}{p_\theta(x_{t+1}, r_{t+1}|y_{0:t})} dx_{t+1} \end{aligned} \tag{C1.5}$$

If one takes a close look at C1.5 one will notice that the first term is the filtered density (3.10). This term is multiplied times a summation and integral dependent on r_t and x_t. Again using the law to total probability the denominator can be written as:

$$\begin{aligned} p_\theta(x_{t+1}, r_{t+1}|y_{0:t}) &= \sum_{r_t\in\mathcal{S}} \int_{\mathcal{X}} p_\theta(x_{t+1}, r_{t+1}|x_t, r_t) p_\theta(x_t, r_t|y_{0:t}) dx_t \\ &= \sum_{r_t\in\mathcal{S}} \int_{\mathcal{X}} p_\theta(x_{t+1}|x_t, r_{t+1}) p_\theta(r_{t+1}|r_t) p_\theta(x_t, r_t|y_{0:t}) dx_t \end{aligned} \tag{C1.6}$$

This distribution can be approximated using importance sampling discussed in Chapter 2 to approximation using a particle filter:

$$p_\theta(x_{t+1}, r_{t+1}|y_{0:t}) \approx \sum_{i=1}^{N_p} w_t^i \, p_\theta\left(x_{t+1}|x_t^i, r_{t+1}\right) p_\theta\left(r_{t+1}|r_t^i\right) \tag{C1.7}$$

The smoothing recursion begins with the weights at time N being set equal to those of the filtered density, and same goes for their particles. Therefore at time N the smoothed weights can be initialized as $w_{t|N}^i = w_N^i$. We now have all the necessary components to estimate the integral in (C1.5) recursively. It is easy to see that at any time t the importance density $\hat{p}_\theta^{N_p}(x_{t+1}, r_{t+1}|y_{0:N})$ is available during the backward recursion. This together with (C1.7)

allows us to make yet another approximation using importance sampling to approximate the integral and sum in (C1.5) as:

$$\sum_{r_{t+1}\in\mathcal{S}}\int_{\mathcal{X}}\frac{p_\theta(x_{t+1}|x_t,r_{t+1})p_\theta(r_{t+1}|r_t)p_\theta(x_{t+1},r_{t+1}|y_{0:N})}{p_\theta(x_{t+1},r_{t+1}|y_{0:t})}dx_{t+1}=$$

$$\sum_{j=1}^{N_p}\frac{w_{t+1|N}^{k}\,p_\theta\left(x_{t+1}^{j}|x_t,r_{t+1}^{j}\right)p_\theta\left(r_{t+1}^{j}|r_t\right)}{\sum_{i=1}^{N_p}w_t^i p_\theta\left(x_{t+1}^{j}|x_t^i,r_{t+1}^{j}\right)p_\theta\left(r_{t+1}^{j}|r_t^i\right)} \tag{C1.8}$$

Finally, to calculate (C1.5) the only thing missing is the filtered density. Since the particle filter approximation this density is already available from the forward recursion, we can approximate the smoothed density at time t as:

$$p_\theta(x_t,r_t|y_{0:N})\;\approx\hat{p}_\theta^{N_p}(x_t,r_t|y_{0:N})\triangleq\sum_{i=1}^{N_p}w_{t|N}^{i}\,\delta\left(x_t-x_t^i\right)\mathbb{1}\left(r_t=r_t^i\right)$$

Where the smoothed weights can be updated recursively for as

$$w_{t|N}^{i}=w_t^i\sum_{j=1}^{N_p}w_{t+1|N}^{j}\frac{p_\theta\left(x_{t+1}^{j}|x_t^i,r_{t+1}^{j}\right)p_\theta\left(r_{t+1}^{j}|r_t^i\right)}{\sum_{l=1}^{N_p}w_t^l\,p_\theta\left(x_{t+1}^{j}|x_t^l,r_{t+1}^{j}\right)p_\theta\left(r_{t+1}^{j}|r_t^l\right)}$$

■

C.2 Closed-Form Maximizer for Transition Probabilities

The derivation for the closed-form maximizer (5.5) is placed here for completeness. The proof presented here is taken directly from [42].

Recall the solution to the constrained maximization problem (5.3-5.4) is proposed to be :

$$\begin{aligned}&\underset{p_{ij}}{minimize}\quad\sum_{i\in}\sum_{j\in}\Psi_{ij}\log\pi_{ij}\\&subject\ to\ \sum_{j\in}\pi_{ij}=1,\ \ \forall i\ \in\mathcal{S}\\&\qquad\pi_{ij}\geq 0,\forall(i,j)\in\mathcal{S}\times\mathcal{S}\end{aligned} \tag{C2.1}$$

with

$$\Psi_{ij}=\sum_{t=1}^{N-1}\sum_{k=1}^{N_p}\sum_{l=1}^{N_p}w_{t|N}^{kl}\,\mathbb{1}\left(r_t^k=j\right)\mathbb{1}\left(r_t^l=i\right) \tag{C2.2}$$

It can be shown that if $\Psi_{ij} \in \mathbb{R}_+ \; \forall (i,j) \in (\mathcal{S} \times \mathcal{S})$ then

$$\hat{\pi}_{ij} = \frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}}, \forall (i,j) \in (\mathcal{S} \times \mathcal{S}) \qquad \text{C2.3}$$

is a maximzer of B2.1.

Proof:

Due to the equality constraint, in B2.1 the system can be decoupled into S independent optimization problems. Thus for each $i \in S$, the following individual optimization problems are solved which is equivalent to the original problem:

$$\underset{p_{ij}}{minimize} \sum_{j \in \mathcal{S}} \Psi_{ij} \log \pi_{ij}$$
$$subject\ to \quad 0 \leq \pi_{ij} \leq 1, \forall (i,j) \in \mathcal{S} \times \mathcal{S} \qquad \text{C2.4}$$

Then,

$$\begin{aligned}
\hat{\pi}_{ij} &= \underset{\pi_{ij}}{argmax} \sum_{j \in \mathcal{S}} \Psi_{ij} \log \pi_{ij} \\
&= \underset{\pi_{ij}}{argmin} \left(-\frac{1}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \right) \sum_{j \in \mathcal{S}} \Psi_{ij} \log \pi_{ij} \\
&= \underset{\pi_{ij}}{argmin} \sum_{j \in \mathcal{S}} \left\{ \frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \log \left(\frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \right) - \frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \log \pi_{ij} \right\} \\
&= \underset{\pi_{ij}}{argmin} \sum_{j \in \mathcal{S}} \left\{ \frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \log \left[\frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \left(\frac{1}{\pi_{ij}} \right) \right] \right\} \\
&= \underset{\pi_{ij}}{argmin} \sum_{j \in \mathcal{S}} \left\{ \frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \log \left[\frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \left(\frac{1}{\pi_{ij}} \right) \right] \right\} \\
&= \underset{\pi_{ij}}{argmin}\, \mathcal{D}_{\mathrm{KL}} \left(\frac{\Psi_{ij}}{\sum_{k \in \mathcal{S}} \Psi_{ik}} \middle\| \pi_{ij} \right)
\end{aligned}$$

where $\mathcal{D}_{\mathrm{KL}}(p\|q) \geq 0$ operator denotes the Kullback-Liebler (KL) divergence or the *relative entropy*, which is a measure of dissimilarity between two distributions p and q [56]. Since the KL divergence satisfies $\mathcal{D}_{\mathrm{KL}}(p\|q) \geq 0$, with equality iff $p = q$, then it follows that C2.3 is a unique feasible maximize of each of the decoupled optimization problems in (C2.4) and is therefore a feasible maximize for the original optimization problem (C2.1) $\forall(i,j) \in \mathcal{S} \times \mathcal{S}$. ■