A Tree Search Approach to Detection and Estimation with Application to
Communications and Tracking

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at George Mason University

By

Hossein Roufarshbaf
Master of Science
Amirkabri University of Technology, 2002
Bachelor of Science
Isfahan University of Technology, 1999

Director: Jill K. Nelson, Assistant Professor
Department of Electrical and Computer Engineering

Summer Semester 2011
George Mason University
Fairfax, VA

UMI Number: 3471058

![UMI Dissertation Publishing logo]

![ProQuest logo]

# Dedication

This dissertation is dedicated to my lovely wife Zeynab Masoudnia for her endless love, encouragement, patience, and support during these years.

# Acknowledgments

# Table of Contents

# List of Tables

# List of Figures

# Abstract

A TREE SEARCH APPROACH TO DETECTION AND ESTIMATION WITH APPLI-
CATION TO COMMUNICATIONS AND TRACKING

Hossein Roufarshbaf, PhD

George Mason University, 2011

Dissertation Director: Dr. Jill K. Nelson

We propose complexity efficient tree search approaches to detection and estimation problems, and we consider both communication and target tracking applications in this regard. We formulate the problems of interest using a dynamic state space model, where we seek to estimate the system state from sensor observations. The proposed tree search approach initializes a search tree from a given initially estimated system state. With each new observation, the algorithm expands the search tree to find the best possible system state. The paths through the search tree correspond to sequences of system states. They are evaluated by their associated path metrics, which are proportional to the posterior probability of the system state. The stack algorithm and the M-algorithm, two tree search approaches that were originally used for decoding convolutionally-encoded sequences, are applied to reduce the complexity of the tree search.

In wireless communication applications, we have considered blind channel equalization of time-varying channels and modulation classification of an unknown received signal. For both applications, tree search approaches are implemented to find the transmitted information sequence that maximizes the posterior probability distribution function of the system state.

For blind channel equalization of time-varying channels, an exponentially decaying window, matched to the variation rate of the channel, is implemented in the path metric calculation to successfully equalize an unknown time-varying channel. For modulation classification of challenging high-density QAM (Quadrature Amplitude Modulation) schemes, the statistical properties of the mean square error are derived analytically and compared with those of the estimated sequence to identify the modulation scheme of the received signal. The proposed tree-search approaches provide superior performance in comparison with other competitive approaches.

In target tracking applications, the proposed stack-based tree search approach sequentially tracks multiple targets in high clutter density with low observable targets and can be applied to both linear and nonlinear target models. The proposed technique is able to retain previously strong track candidates and may refer to them to refine the estimated track if the current track estimate is not satisfactory. A likelihood ratio track management technique has been embedded in the search tree to identify false tracks, and the sensitivity of the proposed tracker to different system parameters has been evaluated analytically.

The results of empirical performance evaluation conducted for the detection and estimation problems described above reveal that the proposed tree search technique performs well in challenging detection and estimation environments with severely non-linear dynamic state space models.

# Chapter 1: Introduction

Applications of detection and estimation theory appear in many scientific problems such as statistical signal processing, time series analysis, econometrics, finance etc. [1, 2]. In statistical signal processing, detection theory deals with identifying, detecting or classifying signals while estimation theory deals with estimating the values of parameters based on observations or measurements. Many statistical problems are modeled using a dynamic state space model. In dynamic state space modeling, variations of the system state over time are modeled by a hidden Markov process, and a measurement model relates the noisy observations to the hidden system state. The Bayesian filter provides the optimum solution with respect to maximum a posteriori (MAP) criteria to the dynamic state space model, but a closed form estimate of the posterior distribution exists only for a small set of scenarios [3]. The most important closed form solution for the Bayesian filter, known as the Kalman filter (KF) [4], is applicable when the state transition and observation models are linear and the additive noise in each model is Gaussian. In nonlinear/non-Gaussian state space models, however, the posterior density function may not have a closed form mathematical expression; in such cases, implementation of the Bayesian filter requires that the posterior density function be approximated by discrete points, an approach that may be computationally impractical. The conventional approach to solving detection and estimation problems involving nonlinear models is to implement the Kalman filtering and linearize the models when the nonlinear models cannot be applied. While these approaches maintain low complexity and may perform satisfactorily for mildly nonlinear systems, their performance suffers dramatically for severely nonlinear systems.

In this work, we propose complexity feasible approaches to detection and estimation problems using partial tree search algorithm techniques that approximate the Bayesian filter by computing the posterior distribution only in regions in which it has significant

mass. In tree search algorithms, the solution is found by navigating a tree of all possible solution candidates and evaluating the likelihood of each candidate based on observations. In order to reduce the complexity of tree search, rather than expanding the tree with all solution candidates, only some of the branches of the tree (those which are more likely to be the solution), are expanded and evaluated. The stack algorithm and the M-algorithm, used in decoding convolutional codes, are two techniques for reducing the complexity of tree search decoders. We have adapted these algorithms to solve detection and estimation problems through partial tree-search techniques.

For demonstration, we have applied the partial tree search techniques to problems in communication systems and to sonar target tracking. In communication systems, we apply a tree search approach based on the stack algorithm for joint blind channel equalization and data detection. We also apply a tree search classifier based on the M-algorithm to perform modulation classification of unknown signals. In the target tracking application, we apply a tree search tracker, based on the stack algorithm, that navigates a tree of most likely target locations to track a moving target in clutter. Our simulation results show that partial tree search techniques can achieve significant performance gains over conventional techniques while maintaining feasible complexity.

This thesis is organized as follows: Dynamic state space models of detection and estimation problems and a review of existing solutions based on these models are discussed in Chapter 2. In Chapter 3 we review the tree search techniques that are used in coding theory, and we expand their applications to general detection and estimation problems. We apply the tree search solutions to communication applications including blind channel equalization and modulation classification in Chapter 4. The proposed tree search technique for target tracking applications is explained and evaluated in Chapter 5. The conclusions and future work are presented in Chapter 6.

# Chapter 2: Background and Related Work

In this chapter, we provide background relevant to the work presented in this thesis. We first describe the dynamic state space model that is used to model many detection and estimation problems. In Section 2.2, we discuss existing solutions to problems in this class and their practical limitations with an emphasis on nonlinear state space models.

## 2.1 Dynamic State Space Model

Many statistical problems can be modeled by sequentially estimating the system state vector, denoted by $\mathbf{x}$, from a vector of noisy observations, denoted by $\mathbf{y}$. For example, in estimating time-varying wireless channels, the system state is given by the time-varying channel vector, and the observation is the signal observed at the receiver. In sonar target tracking systems, the system state vector may contain information about the location and speed of the target; observations are the signal strength, bearing, and arrival time of the returned signal. The dynamic state space model formulates variations of the system state and observations (or measurements) using two processes: the system state process and the observation process.

### 2.1.1 System State Process

In the system state process, variations of the system state $\mathbf{x}$ at time index $k$ are modeled using a first order Markov process given by

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{v}_k), \tag{2.1}$$

where $f_k(.)$ is the known state transition function and $\mathbf{v}_k$ denotes the state transition noise, which is modeled by a random vector with known statistics.

### 2.1.2 Observation Process

The observation model or measurement model relates the observed or measured signal $\mathbf{y}$ to the system state vector $\mathbf{x}$ by

$$\mathbf{y}_k = h_k(\mathbf{x}_k, \mathbf{w}_k), \tag{2.2}$$

where $h_k(.)$ represents the observation function and $\mathbf{w}_k$ denotes the observation noise, assumed to be independent of the state transition noise $\mathbf{v}_k$. Our objective is the sequentially estimation of the system state vector $(\mathbf{x}_k)$ from the observation vector $(\mathbf{y}_k)$ using the given models in (2.1) and (2.2). In the following section, we review the existing approaches to sequential estimation under this model.

## 2.2 Existing Approaches to Detection and Estimation Problem Using Dynamic State Space Models

In the general case, the optimum solution to the state space model with respect to the maximum a posteriori criteria is provided by the Bayesian filter [5], [3]. The Bayesian filter approach provides a general solution using Bayes rule to combine the predefined prior distributions with the likelihood functions of the observation sequences on the system states [3]. Using the Bayesian filter approach, the posterior distribution of the system state $P(\mathbf{x}_k|\mathbf{y}_1 , \cdots , \mathbf{y}_k)$ is derived and propagated as the observations are obtained. The Bayesian filter results in closed-form solutions for only certain classes of systems, however; the most notable are linear models with additive Gaussian noise. In this special case, the Bayesian filter results in the well known Kalman filter [4] approach which provides the efficient and optimum solution in the senses of the minimum mean square error (MMSE), the maximum likelihood (ML), and the maximum a posteriori (MAP) criteria [4], [5]. Under nonlinear/ non-Gaussian state space models, a variety of techniques for nonlinear estimation have been proposed. We review some of the approaches that are frequently used in

digital communication and target tracking applications.

Because of the efficiency and simple implementation of the Kalman filter, many approaches have been developed that focus on applying the Kalman filtering technique and linearizing the state space model when the nonlinear model cannot be applied. The most popular approach in this group is the well known extended Kalman filter (EKF) [1] which linearizes the state space model using Taylor series expansions of the nonlinear functions. While the EKF performs well under mild nonlinearity, it suffers from performance loss under sever nonlinearities. Another approach to linearize the state space model, presented in [6], implements the polynomial approximation of a nonlinear function. This technique, known in the literatures as the central-difference Kalman filter (CDKF), can replace the EKF in some practical applications in which linearization using Taylor series is not accurate. Note that for both EKF and CDKF techniques, because Kalman filtering is applied, there is an implicit Gaussian prior assumption on the posterior density function.

Another group of proposed techniques assume an explicit prior form (normally Gaussian) for the posterior density function. Perhaps the most famous approach in this group is the unscented Kalman filter (UKF) [7], [8] which uses a novel nonlinear transformation of the mean and covariance matrices. In this approach, the mean and covariance matrices of the assumed Gaussian posterior density are parametrized by a set of samples, and these samples are updated through processing by a nonlinear filter. In the Quadrature Kalman filter (QKF) [9], [10] the assumed Gaussian posterior density is parametrized through a set of Gauss-Hermite quadrature points, and the nonlinear process and observation functions are linearized using statistical linear regression (SLR). The Cubature Kalman Filter (CKF) [11] uses a spherical-radial cubature rule for numerical computation of the multivariate moment integrals that appear in the Bayesian filter formulation. The key assumption in this numerical computation is the Gaussianity of the posterior density function.

Some existing approaches do not have explicit assumptions on the posterior distribution. Instead, they estimate a discrete set of parameters and used them to approximate the posterior density function. The Gaussian mixture filter [12], for example, estimates a set

of discrete weight parameters and implements these weights to approximate the posterior density function using a weighted sum of Gaussian density functions. It is shown in [13] that any practical density function can be represented by a weighted sum of Gaussian density functions and as the number of Gaussian densities increases, the approximation will converge uniformly to the desired density function. Recently, particle filtering (PF) [14] has become popular in many applications of statistical signal processing. In particle filtering, a desired posterior density function is estimated through a set of particles and associated weights. At each time update, new particles are generated via sampling from an importance function, and the weight associated with each particle is updated. In some applications (sequence detection in communications, for example), the parameters to be estimated are drawn from a known and finite set of possible values. In these cases, it is possible to select particles in a deterministic fashion. This approach is known as deterministic particle filtering (DPF) [15] and has been shown to achieve better performance than traditional (stochastic) PF schemes when applied to blind equalization [16]. With fast development of powerful computational systems, the numerical approach to solving the Bayesian inference equation that was first introduced in [17] has renewed its attraction through the point mass approach [18]. In the point mass approach, the system state is discretized by a regular grid, and the Bayesian filter is evaluated numerically on the discrete points.

Although the general approaches that do not assume an explicit prior form on the posterior distribution perform well under non-linear and non-Gaussian conditions, they often suffer from extremely high computational complexity. In Chapter 3, we address this challenge by implementing the stack-based [19] and the M-algorithm [20] tree search techniques to evaluate the posterior density only in the regions that appear most likely.

# Chapter 3: Tree Search Algorithms

In the previous chapter, we discussed general approaches for solving detection and estimation problems that do not assume any prior form on the posterior distribution. Tree search-based techniques can be viewed as general approaches that navigates a tree to approximate the maximum likelihood or maximum a posteriori solution. In this chapter, we first describe Bayesian tree search techniques that were originally developed for applications in coding theory. We discuss a full tree search, as well as partial tree search techniques such as the stack algorithm and the M-algorithm that have been developed to decrease the complexity of the tree search technique. We then address the implementation of these techniques to solve general detection and estimation problems.

## 3.1    Bayesian Tree Search Decoder

Orthogonal tree codes (convolutional codes) in digital communication systems have been widely implemented for decades to allow for robust transmission of information sequence over dispersive noisy channels [21]. A variety of algorithms have been developed for decoding convolutional codes. An exhaustive tree search decoder provides the optimum decoding solution with respect to maximum likelihood (ML) or maximum a posteriori (MAP) criteria. In convolutional codes, because of the memory imposed on the coded sequence by the constraint length of the encoder, the current state depends only on limited number of the previous states. Using this property, the tree search technique can be implemented in the trellis format using Viterbi algorithm.

The exhaustive tree search decoder generates all possible transmitted sequences through a tree and evaluates the likelihood (stored in a metric) of each sequence using Bayes rule. At each time step, all new possible sequences are generated, and the metric is updated for each
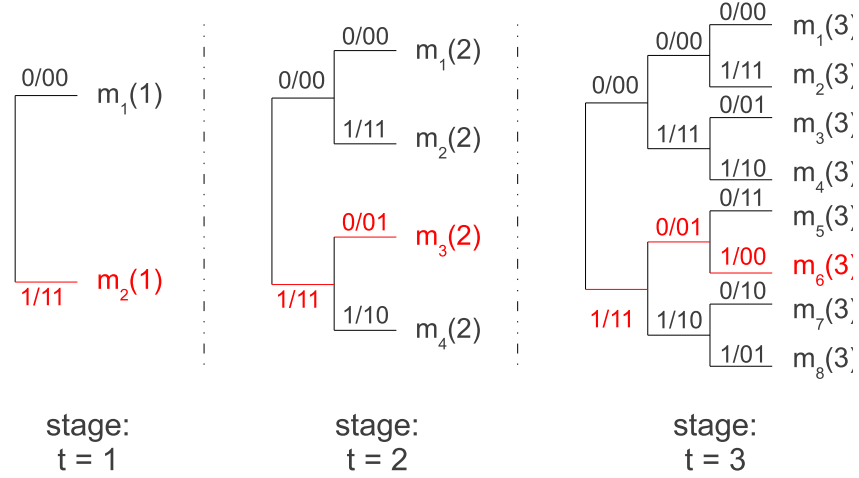
Figure 3.1: An example of the first three iterations of the exhaustive tree decoder algorithm for a rate-$\frac{1}{2}$ code.. We assume that the encoder polynomials are $x^2+1$ and $x^2+x+1$ and the sequence 101 is transmitted. At each iteration, the decoder makes a tree of all possible transmitted sequences and makes a decision based on the highest metric of each possible transmitted sequence. For this example, the highest metric is shown in red, and the decoder output is correct.

sequence using the new observation. Figure 3.1 shows an example of the tree search decoder for a convolutionally encoded sequence. In this example, we assume that the encoder uses the coding polynomials of $x^2 + 1$ and $x^2 + x + 1$ on the information sequence 101. The encoded sequence that is transmitted through the channel is 110100. As we can see in this figure, each branch of the tree represents one possible transmitted sequence; its associated metric is updated each time a new observation is received. In our example, we see that $m_6(3)$ is the highest metric among all the metrics at time stage 3 and 101 is declared as the transmitted sequence. ,

Since the number of possible states (tree branches) increases exponentially with time, implementation of the exhaustive tree search will be computationally complex even when the number of states is small. Due to the structure of the encoder, convolutional decoders

inherently require a finite number of prior states to make a decision. This allow us to implement a trellis decoder (Viterbi algorithm) instead of the tree search. Since the number of states grows exponentially with the constraint length of the encoder, for encoding systems with large constraint length, the trellis decoder may still be prohibitively computationally complex. The stack algorithm and the M-algorithm have been suggested as decoding methods that reduce the complexity of tree search decoding by only partially expanding the search tree.

## 3.2   Stack-Based Tree Search

The stack algorithm navigates a tree in search of the path, or equivalently the data sequence, with the largest likelihood, or metric. A set of possible paths and their associated metrics are stored in a list (or stack), and at each iteration, the algorithm extends the path with the largest metric.

As a simple case, consider a data sequence drawn from a binary alphabet. Starting from a known initial state, the stack algorithm considers both possible path extensions from the initial node, i.e., the single-element path segments $b_1 = 1$ and $b_1 = 0$. The metrics of these two paths are computed, and the paths are stored in the stack along with their metrics. The algorithm then selects from the stack the path with the largest metric and extends it to both children. The two extended paths are placed in the stack, and the parent path is removed. This process continues; in each iteration, the algorithm finds the most likely path in the stack, extends it to all possible children, and places the extended paths in the stack. The stack algorithm terminates when the most likely path in the stack reaches a leaf of the tree, i.e., when a full-length path has the largest metric of any path in the stack. In order to impose a fixed limit on complexity and on memory requirements, a maximum stack size $L$ is typically specified. If $L$ paths have been explored and are stored in the stack, the paths with the lowest metrics are purged to allow for new path extensions.

In Figure 3.2, the stack search algorithm is shown as a partial tree search approach. The same example as Figure 3.1 is used for comparison between the exhaustive tree search and
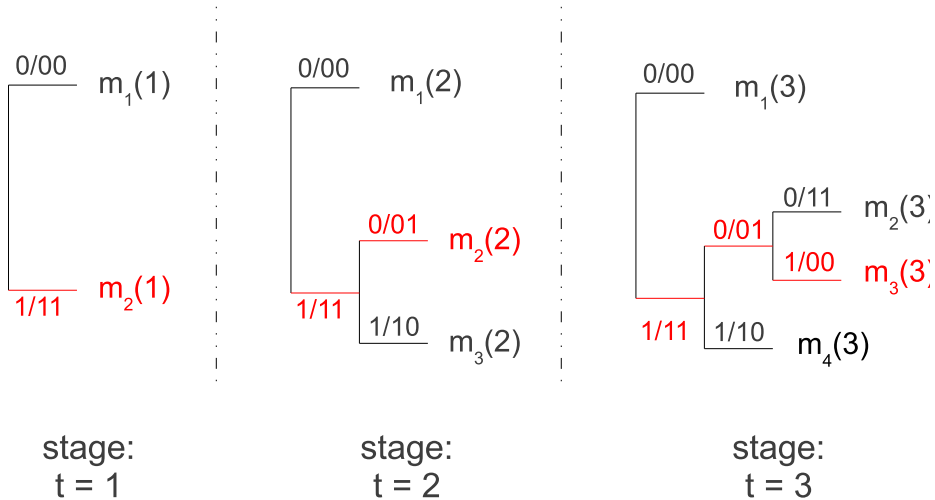
Figure 3.2: An example of the first three iterations of the stack algorithm for a rate-$\frac{1}{2}$ code. Iteration 1: The first two paths are placed in the stack along with their associated metrics. Iteration 2: Path [1] is extended to [1 0] and [1 1], which are placed in the stack. Iteration 3: Path [1 0] is extended to [1 0 0] and [1 0 1], which are placed in the stack. The sequence [1 0 1] has the largest metric at the end of the third iteration and will be extended in iteration 4.

the stack algorithm. At each time stage, rather than extending all possible sequences, only the most likely sequence is extended, and its associated metric is updated. In stage 1, the lower path has the largest metric, and hence it is extended to its two children in stage 2. At stage 3, the path associated with the largest metric $m_2(2)$ is extended and since the path with the largest metric $m_3(3)$ has the full sequence length, it is declared as the decoded path sequence.

Note that the various paths contained in the stack at a particular time are not all of the same length. The algorithm may extend a path of length $l$ in one iteration and in the next iteration find the most likely path to be one of length $j$ for some $j \ll l$. For blocks with low SNR, the tendency of the stack algorithm to jump around the tree during the detection process causes the number of required computations to be quite large [22]. Because paths of

unequal length may be compared, the path metric must include a bias term to compensate for differences in path length.

## 3.3   M-Algorithm Tree Search

The M-algorithm decoding technique differs from the stack algorithm in that, at each time update, rather than extending only the most likely path, all stored paths are extended to all possible children. After extension, only the $M$ most likely paths are retained. For example, suppose that there are $M$ stored path, and that each node has $L$ children. Therefore, at each time update $ML$ sequences will be generated. The metrics are updated for all $ML$ new paths, and only the $M$ most likely paths are retained for the next time update.

Recently, deterministic particle filtering has been proposed [15] as an alternative approach to the conventional particle filtering [14] for the systems in which the solution to the system state estimation lies in a deterministic set of possible solutions. In this algorithm, each particle represents one possible solution candidate and the weight associated to each particle shows its likelihood. At each time update, all the retained candidates are extended and the weight is updated for each candidate. Only M particles with highest weights are retained. Deterministic particle filtering is the same as the M-algorithm tree search when the number of particles at each time update are limited to $M$. In Section 4.2, we apply the deterministic particle filtering algorithm to modulation classification of an unknown signal.

## 3.4   Tree Search Approaches for Detection and Estimation Problems

In the previous section, we described tree search approaches for decoding convolutionally encoded sequences. Now we extend the applications of these techniques to a general detection and estimation problem using the dynamic state space modeling. The idea behind the tree search technique is to approximate the Bayesian inference solution by using a tree
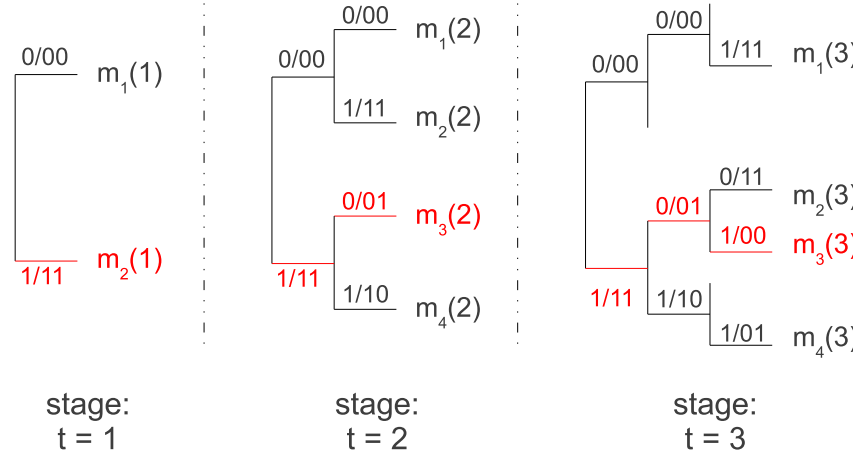
Figure 3.3: An example of the first three iterations of the M-algorithm with M equal to 4 and a rate-$\frac{1}{2}$ code. At the first and second iteration, the algorithm performs similar to full tree search since the number of paths does not exceed M. At the third iteration, all previously stored paths are expanded but the tree is pruned and only the $M$ most likely paths are retained, thereby reducing complexity for iterations that follow.

search algorithm to explore and evaluate only likely regions (e.g. regions with higher amplitude) of the posterior distribution on the system state. Modeling using dynamic state space model represented in (2.1) and (2.2), the tree search technique must evaluate the posterior density function and estimate the solution using MAP criteria:

$$[\hat{\mathbf{x}}_1 \; \cdots \; \hat{\mathbf{x}}_k] = \arg \max_l \mathbf{P}(\mathbf{x}_1^{(l)}, \cdots, \mathbf{x}_k^{(l)} | \mathbf{y}_1, \cdots, \mathbf{y}_k), \tag{3.1}$$

where $l$ denotes the $l$th solution candidate within the search tree.

Applying tree search techniques to a general detection and estimation problem requires three steps:

1. Defining a search tree structure.

2. Discretization of the system state process.

3. Metric derivation.

In the following, we discuss each of these steps in more detail.

### 3.4.1 Search Tree Structure

To apply a tree search technique for a general problem modeled by (2.1), and (2.2), the search tree must be able to generate all possible solution candidates. When the appropriate tree is generated, the solution $\hat{\mathbf{x}}$ defined in (3.1) can be found by navigating the tree and comparing the metric associated to each solution candidate. To make a search tree, each node of the tree at time stage $k$ must represent a possible value of the system state at that time, e.g. a possible value of $\mathbf{x}_k$. At each time update, the tree is extended from a present state or tree node $\mathbf{x}_k$, to all possible system states that can be produced according to (2.1).

In order to initiate the search tree, additional information about the initial point is required. In communication systems, when the transmitted information sequences are considered as the systems state, the initial state is usually assumed to be zero or the tree is started from the initial rest condition. In tracking applications, the initial state of the tree may denote the initial target location that is estimated by an external track initiation algorithm and is assumed to be available for the tracker.

Each path of the search tree starts from the initial point of the tree and ends to a unique leaf of the tree, so the number of full-length paths in the search tree is equal to the number of leaves. Each path represents a unique sequence of the system states $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k]$ and the tree search algorithm navigates the tree by exploring the sequences of states that appear most likely based on noisy observations.

### 3.4.2 Discretization of the System State Process

Application of the tree search algorithm always requires a discrete (or discretized) system state process, since the number of possible states must be finite in order to map each possible system state value to a branch of the search tree. By discretizing the system state process,

the continuous posterior distribution on the state is replaced by a probability mass function (PMF).

In some applications like digital communications, the system state process denotes the transmitted information sequence and is inherently discrete. For expansion of each tree node, the number of new generated branches from each node of the tree is equal to the number of symbols in the constellation scheme of the transmitted symbols.

In tracking applications, however, the system state contains the target location information in the space and must be discretized. Figure 3.4 shows an application of the tree search technique (stack algorithm) for target tracking. In this example, each element of the system state vector is represented only by 3 discrete values, and hence $3^2$ new branches are generated from each node of the tree. We can see that by increasing the number of discretized values in each dimension, the number of branches emanating from each node increases exponentially. On the other hand, decreasing the number of discrete values increases the estimation error due to the discretization error. In Chapter 5, we discuss this challenge in more detail, and we propose a dynamic discretization approach to decrease the discretization error without increasing the tree size.

### 3.4.3   Metric Evaluation

As stated before, the goal of the tree search technique is to find (or approximate) the MAP solution through navigating a tree of possible solutions. Since the state space must be discrete or discretized, the possibly continuous posterior distribution on the state is evaluated as a probability mass function (PMF). Each path of the tree represents a point in which the PMF is evaluated, and the metric associated to each path is proportional to
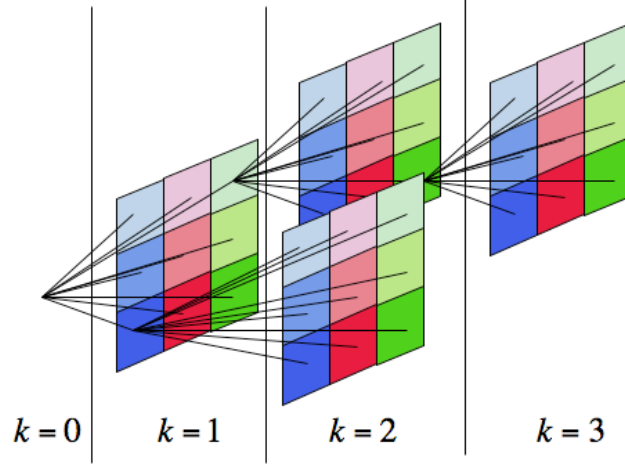
Figure 3.4: An example of the stack algorithm for target tracking application. The state vector that denotes the location of the target in a surface is discretized in the 2-dimensional Cartesian coordinates. Each dimension of a new generated state can take 3 discrete values that totally generates 9 new states for each extension.

the PMF value. The general approach to derive a path metric is through Bayes Theorem:

$$
\begin{aligned}
\mathbf{P}(\mathbf{x}_0^{(l)}, \cdots, \mathbf{x}_k^{(l)} | \mathbf{y}_0, \cdots, \mathbf{y}_k) &= \frac{\mathbf{P}(\mathbf{y}_0, \cdots, \mathbf{y}_k | \mathbf{x}_1^{(l)}, \cdots, \mathbf{x}_k^{(l)}) \, \mathbf{P}(\mathbf{x}_0^{(l)}, \cdots, \mathbf{x}_k^{(l)})}{\mathbf{P}(\mathbf{y}_1, \cdots, \mathbf{y}_k)} \\[2mm]
&= \frac{\left( \prod_{i=0}^{k} \mathbf{P}(\mathbf{y}_i | \mathbf{x}_i^{(l)}) \right) \left( \prod_{i=1}^{k} \mathbf{P}(\mathbf{x}_i^{(l)} | \mathbf{x}_{i-1}^{(l)}) \right) \mathbf{P}(\mathbf{x}_0^{(l)})}{\mathbf{P}(\mathbf{y}_1, \cdots, \mathbf{y}_k)}, \quad (3.2)
\end{aligned}
$$

where the second line is derived using the fact that the observation vector at each time $(\mathbf{y}_l)$, only depends on the system state vector of the current time $(\mathbf{x}_l)$ (2.2) and the system state process follows the first order Markov process (2.1).

The metric evaluated in (3.2) can be applied to the full tree search algorithm or the M-algorithm in which all the retained paths in the tree are extended to the same time $k$. In the stack-algorithm, however, the paths within the search tree are extended to different time indices; the metric of each path must contain a compensation, or bias, term to allow fair comparison between paths of different lengths.

The tree search technique with the metric defined by (3.2) approximates the MAP solution to the dynamic state space model. Some applications require the maximum likelihood (ML) solution that can be derived through the tree search technique if only the likelihood term of the metric presented in (3.2) is considered for the path metric:

$$\mathbf{P}(\mathbf{y}_0, \cdots, \mathbf{y}_k | \mathbf{x}_1^{(l)}, \cdots, \mathbf{x}_k^{(l)}) = \prod_{i=0}^{k} \mathbf{P}(\mathbf{y}_i | \mathbf{x}_i^{(l)}). \tag{3.3}$$

In Chapter 4 and Chapter 5, we apply tree search approaches to communication and tracking applications. In communication systems, we apply the stack algorithm for sequential detection of the transmitted sequence in the presence of a time-varying multipath channel. We also apply the M-algorithm for modulation classification of unknown signals. In tracking applications, we apply the stack algorithm for tracking a moving target in the presence of clutter; the stack algorithm estimates the target location through partially navigating of a tree of all possible target locations.

# Chapter 4: Implementation of Bayesian Tree Search Approaches in Wireless Communication Systems

In modern telecommunication systems, the demand for fast services in which the end user is able to remain mobile is increasing. An important and challenging problem in these systems is transmission of information over time-varying multipath fading channels. Multipath fading may generate intersymbol interference (ISI) in the time domain, and equivalently frequency selectivity in the frequency domain; time variation generates fast fading in the time domain and doppler shift in the frequency domain of the received signal [23]. While ISI provides time diversity for communication systems, it may degrade the performance of the system if not completely compensated at the receiver. The traditional way to combat ISI is through channel equalization techniques. To set up an appropriate equalizer, the communication channel should be estimated using a training sequence [24]. The training sequence is a sequence of bits that is transmitted by the transmitter and is known at the receiver. The receiver estimates the channel parameters based on the observations from the training period. Transmitting a training sequence wastes useful bandwidth, however, and in time varying channels the training sequence must be transmitted repeatedly. Thus it wastes bandwidth and decreases the efficiency of the communication system. Additionally, in new technologies such as ad hoc mobile networks and packet radios, data sequences are transmitted with differing packet lengths, and the channel may vary during the transmission of a packet and also from packet to packet. Channel estimation based on transmitting a training sequence is not efficient in these scenarios. In applications like modulation classification of an unknown signal, the receiver may not have access to the training sequence of the signal and implementation of blind equalization techniques, in which the receiver combats the ISI without using the training sequence, is inevitable.

In this chapter, we consider two applications of Bayesian tree search in wireless communications systems, in which for both scenarios the receiver implements blind equalization techniques. In Section 4.1, we expand the Bayesian tree search approach for joint blind equalization and sequence detection to time-varying multipath channels. In Section 4.2, we propose an M-algorithm for modulation classification of an unknown received signal transmitted over a multipath channels.

## 4.1 Stack Tree Search for Maximum Likelihood Sequence Detection in Multipath Rayleigh Fading Channels

### 4.1.1 Introduction

The problem of channel equalization and decoding appears in many communication systems. The traditional approach to channel equalization is through either estimating the channel vector or adjusting the equalizer coefficients using a transmitted training sequence. As discussed, however, transmitting a training sequence is not possible in all systems

The first idea of channel equalization without using a training sequence was introduced by Sato [25]. Since then, many blind channel equalization techniques have been considered in communication systems, but most of them use estimation of higher order statistics (HOS) of the received signal [26, 27]. The drawback of HOS approaches is the estimation of higher order statistics which require a large sequence of the received signal and increases the delay at the receiver. Additionally, in the case of fast time varying channels, due to non-stationarity of the received signal, these statistics can not be estimated precisely.

Among all the algorithms that have been proposed so far, those that are able to detect the transmitted data based on few received symbols are more promising for many new wireless applications, particularly those allowing packet transmission. Tong et. al. [28] proved that in the case of either multiple receivers or oversampling of the received signal, it is possible to estimate the channel coefficients from second order statistics. Since then, many approaches based on a single input multiple output (SIMO) model have been introduced,

the most recent of which can be found in [29]. For systems that can not be modeled as SIMO, these second-order based equalization techniques do not apply.

Joint channel equalization and sequence detection is introduced by Forney [30] in which he uses the Viterbi algorithm (VA) for maximum likelihood sequence detection in ISI conditions. However, the proposed algorithm requires channel estimation through transmitting the training sequence. Since then, many Viterbi-based approaches for blind channel equalization and sequence detection have been proposed. Seshadri [31] applies a minimum mean square error based approach for blind channel equalization and proposes a Viterbi-based blind algorithm that allows an increase in the number of surviving sequences retained in the Viterbi trellis. When the channel is known in the Viterbi algorithm, only the sequence with the highest likelihood is retained at each state. When the channel is unknown, Seshadri suggests keeping multiple possible sequences for each state and estimating the channel based on each possible sequence. At the next time update, all the retained sequences for each state are updated; only the M sequences with highest path metrics are retained. Kubo et al. [32] proposed an adaptive maximum likelihood sequence detection (MLSD) approach for combined channel equalization and decoding in fading environments. In this approach that is a Viterbi based algorithm, the channel is estimated separately for each respective state of the VA using the analytically derived least mean square (LMS) algorithm. Xiaohua [33] introduced a channel independent Viterbi algorithm (CIVA) in which the path metrics are evaluated from a bank of test vectors that are designed off-line. Yet another Viterbi based approach proposed in [34] uses the Viterbi algorithm and Kalman filtering for channel estimation on the surviving paths.

The optimum solution for both blind channel equalization and sequence detection is to generate all possible data sequences and evaluate the posterior density of each generated sequence given the observation signal. Tree search algorithms can be implemented to generate all possible transmitted sequences, but the tree expands exponentially and it is not practical to evaluate all possible transmitted sequences. Recently, a particle filtering approach has been suggested [35] that estimates the posterior density of the transmitted

symbols using a set of particles and their associated weights. Deterministic particle filtering (DPF), proposed in [16], is similar to the tree search approach when only the $M$ most probable paths are retained. DPF shows good performance in the presence of time-varying channels, but it suffers from complexity issues since at each time step, all possible new generated paths from the $M$ previously retained paths, must be extended and the metric must be updated for all new paths. For example, if a transmitted sequence comes from a QPSK constellation, there are 4 new paths from each retained path, and in total $4M$ paths must be generated and evaluated at each time update.

The stack like tree search algorithm presented in [36], which forms the basis for this work, allows unequal path lengths within the tree and extends only the path with the highest metric. Therefore, it can find the best path more quickly by avoiding unnecessary path extensions. Additionally, in comparison with [31], this algorithm has the ability to retain more possible sequences for situations in which signal to noise ratio is low. In this section, we have extended the stack-like tree search algorithm to be used for joint sequence detection and blind equalization in time-varying Rayleigh fading channels.

### 4.1.2   System Model

We consider the baseband equivalent of the telecommunication system as depicted in Figure 4.1. In order to model the communication system as its baseband equivalent, we assume that the receiver is synchronized to the transmitter and knows the pulse shaping filters. The information bits are assumed to be transmitted in blocks of length $N$ that are denoted by $\mathbf{b}_1^N$. The information bits are encoded through a rate $R = \frac{1}{r}$ convolutional encoder prior to transmission over the channel. The encoder parameters such as constraint length, encoding polynomial, and initial state are assumed to be known at the receiver. The transmitted signal is a block of symbols with length of $rN$ denoted by $\mathbf{x}_1^{rN}$. These encoded sequences are passed through a linear time-varying (LTV) complex channel denoted by vector $\mathbf{h}_t = [h_{t,0} \cdots h_{t,L-1}]$ where $t$ is the time index and $L$ is the cardinality of the channel vector (channel length). We have assumed that the channel variation is slow during transmission

of one symbol of the transmitted sequence (slow fading channel). The complex channel vector $\mathbf{h}_t$ can be written as $\mathbf{h}_t = \mathbf{h}_t^c + j\mathbf{h}_t^s$, where $\mathbf{h}_t^c$ and $\mathbf{h}_t^s$ are the in-phase and quadrature components, respectively. The noise $w_t$ is assumed to be additive white Gaussian noise (AWGN) with in-phase and quadrature components $w_t^c$ and $w_t^s$, where $w_t^s$ and $w_t^s$ are uncorrelated white Gaussian noise processes with variance $\sigma^2 = N_0/2$. We assume that the receiver knows the variance of the additive white Gaussian noise (AWGN).

For simplicity, the modulation of the transmitted sequence is assumed to be binary phase shift keying (BPSK) . The received signal can be written as

$$
\begin{aligned}
z_n &= z_n^c + j z_n^s = \left[ \sum_{k=0}^{L-1} h_{t,k} x_{t-kTs} + w_t \right]_{t=nT_s} \tag{4.1} \\
&= \left[ \sum_{k=0}^{L-1} h_{t,k}^c x_{t-kTs} + w_t^c \right]_{t=nT_s} + j \left[ \sum_{k=0}^{L-1} h_{t,k}^s x_{t-kTs} + w_t^s \right]_{t=nT_s},
\end{aligned}
$$

where $z_n$ denotes the $n$th sample of the channel output and $T_s$ is the sampling time period at the receiver. To find the statistical properties of the Rayleigh fading channel, we use the famous Jakes' channel reference model [23] where each channel tap is comprised of Q propagation paths as

$$
\begin{aligned}
h_{t,l} &= h_{t,l}^c + j h_{t,l}^s \tag{4.2} \\
&= E_0 \sum_{q=1}^{Q} C_{q,l} \cos(\omega_d t \cos \alpha_q + \phi_q) + \; j E_0 \sum_{q=1}^{Q} C_{q,l} \sin(\omega_d t \cos \alpha_q + \phi_q).
\end{aligned}
$$

$E_0$ is a scaling constant, $C_{q,l}$, $\alpha_q$ and $\phi_q$ are the random path gain, angle of incoming wave, and initial phase associated with the $q$th propagation path, respectively, and $\omega_d$ is the maximum radian Doppler frequency [37]. Based on these assumptions, the channel
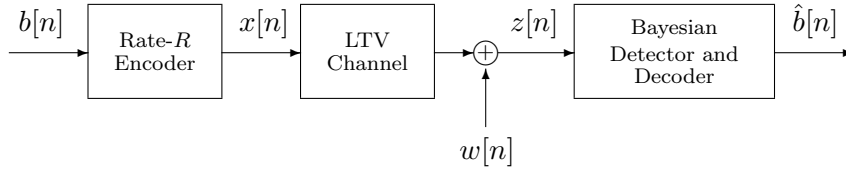
Figure 4.1: Block diagram of communication system. The information bits are encoded through a rate $R$ convolutional encoder. The encoded sequence $x[n]$ is transmitted over a multipath linear time-varying (LTV) channel. The additive noise is assumed to be white and Gaussian. The role of the Bayesian detector and decoder is to estimate the transmitted information bits from the observed signal.

autocorrelation functions, consistent with Jakes' reference model (4.2), are equal to

$$
\begin{aligned}
E[h_{t,l}^c h_{t+\tau,l}^c] &= E[h_{t,l}^s h_{t+\tau,l}^s] = J_0(\omega_d \tau) \qquad (4.3) \\
E[h_{t,l}^c h_{t+\tau,l}^s] &= E[h_{t,l}^s h_{t+\tau,l}^c] = 0,
\end{aligned}
$$

where $J_0(.)$ denotes the zero-order Bessel function of the first kind.

The block of the received samples $z[1], \cdots, z[rN]$ is denoted by $\mathbf{z}_1^{rN}$ and serves as input to the detection and decoding block. The detection and decoding block uses a stack-based algorithm to estimate the transmitted data sequence $\mathbf{b}_1^N$ by navigating a tree generated by the transmitted data state. The channel is implicitly estimated through the metric of each path. In what follows, we explain the Bayesian stack algorithm and its implementation issues for the LTV channel.

### 4.1.3 Implementation of Bayesian Stack Tree Search

When channel information is present at the receiver, MLSD can be performed via the Viterbi algorithm [23]. The key observation in the Viterbi algorithm is that the memory in the encoder can be represented by a finite state machine. When the channel information is not available at the receiver, the sequence likelihood no longer factors onto a trellis, and hence the Viterbi algorithm cannot be directly applied. As an alternative, the Bayesian stack

detector that is discussed in Chapter 2 traverses a tree to find the most likely sequence detector. The stack algorithm simplifies the tree search so that, at each time update, rather than expanding all branches of the tree, only the most probable branch is extended. The associated metric calculated at each time update is compared with the metric of the other paths in the tree, and only the most likely path is selected for extension the next time update. When the number of stored paths in the stack exceeds the size of the stack, the paths with lower metrics are dropped from the stack. The algorithm finishes when the path with highest metric is of full length, e.g. reaches a leaf of the tree.

To calculate the metric for the proposed stack algorithm, we have to evaluate the probability of each possible transmitted information bit sequence in the stack based on the received signal

$$m \propto p(\mathbf{b}_1^n | \mathbf{z}_1^{rN}, C) \tag{4.4}$$

where $C$ denotes the explored tree at the current iteration. Assuming a prior over the channel taps, the conditional likelihood of a bit sequence $\mathbf{b}_1^n$ is averaged over the unknown quantities,

$$p(\mathbf{b}_1^n | \mathbf{z}_1^{rN}, C) = \int_{\mathbf{h}_t} p(\mathbf{b}_1^n, \mathbf{h}_t | \mathbf{z}_1^{rN}, C) d\mathbf{h}_t. \tag{4.5}$$

Using Bayes Theorem, (4.5) can be written as

$$
\begin{aligned}
\int_{\mathbf{h}} p(\mathbf{b}_1^n, \mathbf{h} | \mathbf{z}_1^{rN}, C) d\mathbf{h} &= \int_{\mathbf{h}} p(\mathbf{b}_1^n | \mathbf{h}, \mathbf{z}_1^{rN}, C) p(\mathbf{h}) d\mathbf{h} \\
&= \int_{\mathbf{h}} \frac{p(\mathbf{z}_1^{rN} | \mathbf{h}, \mathbf{b}_1^n, C) p(\mathbf{h}) p(\mathbf{b}_1^n) d\mathbf{h}}{p(\mathbf{z}_1^{rN} | \mathbf{h}, C)} \\
&\propto \int_{\mathbf{h}} p(\mathbf{z}_1^{rN} | \mathbf{h}, \mathbf{b}_1^n, C) p(\mathbf{h}) d\mathbf{h}.
\end{aligned} \tag{4.6}
$$

In order to evaluate the integral in (4.6), we first assume that the channel is time-invariant ($\mathbf{h}_t = \mathbf{h}$) and the number of channel taps (channel length) is known at the receiver. After that we extend our results to incorporate time-varying channels and unknown channel length. The key assumption in deriving a closed-form metric is that the likelihood function $p(\mathbf{z}_1^{rN}|\mathbf{h}, \mathbf{b}_1^n, C)$ can be written as the product of the likelihood function of each observed symbol as

$$p(\mathbf{z}_1^{rN}|\mathbf{h}, \mathbf{b}_1^n, C) = \prod_{k=1}^{rN} p(\mathbf{z}_k|\mathbf{h}, \mathbf{b}_1^n, C). \tag{4.7}$$

In other words, the received symbols, when conditioned on the channel and the transmitted data, are independent. Although, this is not a valid assumption in general since the transmitted symbols are correlated due to the convolutional channel encoder and the multipath channel, it simplifies the calculation of the stack metric. In practice, a scrambling technique employed in turbo-style minimized correlation between successive transmitted symbols. This technique reduces the correlation between successive symbols by reordering the transmitted symbols using the specific scramble generation polynomial. Given the channel state vector and the transmitted sequence, $p(\mathbf{z}_k|\mathbf{h}, \mathbf{b}_1^n, C)$ follows the probability distribution function of the additive noise

$$p(\mathbf{z}_k|\mathbf{h}, \mathbf{b}_1^n, C) = \mathcal{N}(0, \sigma^2). \tag{4.8}$$

Substituting (4.7) and (4.8) in (4.6) and assuming a Gaussian prior over the channel vector,

the integral in (4.6) has a closed form and can be calculated as [36]

$$p(\mathbf{b}_1^n | \mathbf{z}_1^{rN}, C) \tag{4.9}$$

$$\propto \quad \frac{\sigma_h^L}{2^n} \left( \frac{(\sigma^2 + 1)^{\frac{r(n-N)}{2}}}{\sigma^{rn}} \right) \left| \frac{\hat{R}_{xx}^{rn}}{\sigma^2} + \frac{I}{\sigma_h^2} \right|^{-1/2} \times$$

$$\exp\left\{ -\frac{\hat{R}_{zz}^{rn}[0]}{2\sigma^2} + \frac{1}{2\sigma^4} \hat{\mathbf{r}}_{zx}^{rnT} \left( \frac{\hat{R}_{xx}^{rn}}{\sigma^2} + \frac{I}{\sigma_h^2} \right)^{-1} \hat{\mathbf{r}}_{zx}^{rn} \right\} \times$$

$$\exp\left\{ -\frac{1}{2(\sigma^2 + 1)} \sum_{i=rn+1}^{rN} z_i^{\,2} \right\},$$

where

$$R_{zz}^k[0] \quad = \quad \sum_{i=1}^{k} z_i^2 \tag{4.10}$$

$$\mathbf{r}_{zx}^k \quad = \quad \sum_{i=1}^{k} z_i \mathbf{x}_{i-L+1}^i \tag{4.11}$$

$$R_{xx}^k \quad = \quad \sum_{i=1}^{k} (\mathbf{x}_{i-L+1}^i)(\mathbf{x}_{i-L+1}^i)^{\mathrm{H}}. \tag{4.12}$$

When the channel response varies with time, equations (4.10)-(4.12) estimating the auto-correlation and cross-correlation functions are no longer valid, since the ensemble estimate of $R_{zz}$, $r_{zx}$ and $R_{xx}$ is not equal to the time average estimate. To allow for a time-varying channel, we incorporate a forgetting factor $\lambda$ $(0 \leq \lambda \leq 1)$ to control the excessive lag error of the estimates of $R_{zz}$, $r_{zx}$ and $R_{xx}$ with channel variation rate. With this adjustment, the

auto-correlation and cross-correlation estimates in (4.10)-(4.12) can be rewritten as

$$\hat{R}^k_{z^c z^c}[0] = \frac{1-\lambda}{1-\lambda^{k+1}} \sum_{i=1}^{k} \lambda^{k-i} z_i^{c2} \tag{4.13}$$

$$\hat{\mathbf{r}}^k_{z^c x} = \frac{1-\lambda}{1-\lambda^{k+1}} \sum_{i=1}^{k} \lambda^{k-i} z_i^c \mathbf{x}_{i-L+1}^i \tag{4.14}$$

$$\hat{R}^k_{xx} = \frac{1-\lambda}{1-\lambda^{k+1}} \sum_{i=1}^{k} \lambda^{k-i} \left(\mathbf{x}_{i-L+1}^i\right) \left(\mathbf{x}_{i-L+1}^i\right)^{\mathrm{H}}. \tag{4.15}$$

The scaling factor $(1-\lambda)/(1-\lambda^{k+1})$ has been incorporated to compensate for the case in which only a short sequence is available and the memory imposed by the forgetting factor is greater than the number of available symbols used for estimating the correlation matrices. Note that

$$\frac{1-\lambda}{1-\lambda^{k+1}} \sum_{i=1}^{k} \lambda^{k-i} = 1.$$

For the stationary (time-invariant) channel, the expected value of (4.13)-(4.15) is the same as the expected value of (4.10)-(4.12).

The following two subsections address (1) the relationship between the forgetting factor and the channel coherence time, and (2) the challenge of unknown channel order at the receiver.

**Setting the Forgetting Factor**

Reliable estimation of the correlation matrices in (4.13)-(4.15) requires appropriate selection of the forgetting factor value. To achieve this, we first model the channel variations in time. We assume that the time-varying channel follows a first-order Markov model given by

$$\mathbf{h}_{nT_s} = \mathbf{h}_{(n-1)T_s} + \mathbf{v}_n, \tag{4.16}$$

where $\mathbf{v}_n$ is a zero-mean Gaussian random vector with diagonal covariance matrix $\sigma^2 I$. The optimum channel estimate based on the minimum mean square error (MMSE) criteria is the one that minimizes the mean square error (MSE) of the estimated observation data with the actual observation data, which gives [1]

$$\hat{\mathbf{h}}^T = E\{zx^*\}\left(E\{xx^*\}\right)^{-1}.$$

Note that $\hat{R}_{xx}^{rn}$ and $\hat{\mathbf{r}}_{zx}^{rn}$ are the scaled estimate of the autocorrelation and cross correlation matrices up to time $rn$. The MSE of the system using the estimated channel is given by

$$\epsilon \;=\; E\{zz^*\} - E\{zx^*\}E\{xx^*\}^{-1}E\{xz^*\} \qquad (4.17)$$

$$\propto \;\; \hat{R}_{zz}^{rn}[0] - \frac{1}{\sigma^2}\hat{\mathbf{r}}_{zx}^{rnT}\left(\frac{\hat{R}_{xx}^{rn}}{\sigma^2} + \frac{I}{\sigma_h^2}\right)^{-1}\hat{\mathbf{r}}_{zx}^{rn}. \qquad (4.18)$$

The first exponential term of the metric derived in (4.9) contains the MSE of the channel estimator. When the channel is perfectly known, the MSE is equal to the noise variance, $\sigma^2$. When the channel is both unknown and time-varying, additional MSE is introduced via estimation and lag error through the auto- and cross- correlation expressions. The estimation error depends on how much data is employed for approximating the auto-correlation and cross-correlation matrices. The minimum value of the estimation error is equal to the variance of the additive noise, and the estimation error decreases by increasing the forgetting factor. The lag error reflects the excess MSE component that comes from the non-stationarity of the channel. Lag error decreases by selecting a smaller forgetting factor. Therefore, the forgetting factor must be chosen to achieve a balance between estimation error and lag error. A similar result for excessive estimation error and lag error is derived for the recursive least squares algorithm (RLS) tracking a time varying system in [38]. Using the result of this work, the excessive MSE component due to estimation error and lag can

be approximated by

$$\epsilon_{est} \quad \approx \quad \frac{1-\lambda}{1+\lambda}L\sigma^2 \tag{4.19}$$

$$\epsilon_{lag} \quad \approx \quad \frac{1}{2(1-\lambda)}L\sigma_x^2\sigma_v^2, \tag{4.20}$$

where $\sigma_x^2$ is the variance of the transmitted signal, $L$ denotes the cardinality of the channel vector, and $\sigma_v^2$ is the variance of the channel transition noise vector. The total excessive MSE is then defined by

$$\epsilon_{tot} \approx \frac{1-\lambda}{1+\lambda}L\sigma^2 + \frac{1}{2(1-\lambda)}L\sigma_x^2\sigma_v^2. \tag{4.21}$$

The optimum $\lambda$ can be derived by differentiating (4.21) with respect to $\lambda$, and the result is given by

$$\lambda_{opt} = \frac{1 - \left(\frac{\sigma_x^2\sigma_v^2}{4\sigma^2}\right)^{\frac{1}{2}}}{1 + \left(\frac{\sigma_x^2\sigma_v^2}{4\sigma^2}\right)^{\frac{1}{2}}}. \tag{4.22}$$

The variance of the channel transition noise, $\sigma_v^2$, is the parameter that relates to the channel coherence time of the time-variant Rayleigh fading channel. To find this relationship, from (4.16) we have

$$v \quad = \quad (h_{nT_s} - h_{(n-1)T_s})$$

and

$$\sigma_v^2 \quad = \quad E\{(h_{nT_s} - h_{(n-1)T_s})(h_{nT_s} - h_{(n-1)T_s})^H\}.$$

Using the result of (4.3) and the fact that $h = h^c + jh^s$, we have

$$
\begin{aligned}
\sigma_v^2 &= J_0(0) - J_0(\omega_d T_s) + J_0(0) - J_0(\omega_d T_s) + J_0(0) - J_0(\omega_d T_s) + J_0(0) - J_0(\omega_d T_s) \\
&= 4J_0(0) - 4J_0(\omega_d T_s),
\end{aligned}
\tag{4.23}
$$

where $\omega_d$ is the maximum radian doppler frequency and the coherence time of the channel is approximated by [23]

$$
T_c \approx \frac{2\pi}{\omega_d}.
\tag{4.24}
$$

Equations (4.22), (4.23), and (4.24) relate the coherence time of the channel to the optimum forgetting factor.

**Addressing Unknown Channel Length**

In the derivation of the path metric for the stack algorithm, we have assumed that the receiver knows the channel vector length. This is not true in general, so the receiver should be capable of estimating the order of the channel. Estimating the channel length plays an important role in blind equalization techniques since many blind equalization techniques are sensitive to overestimating the channel length. Common approaches for estimating channel order are through the well known Akaike information criterion (AIC) [39] and minimum description length (MDL) [40] approaches. From these two approaches, MDL is preferable since it avoids over-estimation of the channel length. Implementing the AIC or MDL methods requires a good estimate of the covariance matrix of the observation sequence. The stack algorithm, particularly in its early stages, does not have a good estimate of this covariance matrix. The other limitation of implementing AIC or MDL is that their performance degrades with the symbol correction introduced by the encoder.

To overcome uncertainty of the channel order, we calculate the path metric given in (4.9) under different channel length assumptions. In communication systems, some statistical channel information such as the power delay profile is often available. From the power

delay profile, we can estimate the maximum channel duration ($L_{max}$) for a communication system. Given the range of possible channel lengths, we can average over all the possible values of channel length to evaluate the metric. Through simulation experiments, we have observed that the stack algorithm is sensitive to over-estimating the channel length, and hence averaging over all possible values of $L$ does not perform well. From this experiment, we decided to use only the maximum metric that is generated from each possible value of $L$ as follows:

$$P(b_1^n|z_1^{rn}, C^k) \tag{4.25}$$

$$= \max\{P(b_1^n|Z_1^{rN}, C^{(k)}, l)\} \quad l = 1, .., L_{max},$$

In the next section, we evaluate the performance of the proposed algorithm and compare it with some competitive approaches for joint sequence detection and blind channel equalization.

## 4.1.4  Simulation Results

To evaluate the performance of the proposed algorithm, we have simulated the communication system model of Figure 4.1. The information bits are randomly generated and are encoded using convolutional encoder shows in Figure 4.1 using a rate $R = 1/2$ convolutional code with generator matrix $G(x) = [x+1 \quad x^2+x+1]$. We have simulated a time-varying channel using the corrected statistics simulator of a Rayleigh fading channel as published in [41]. Each tap of the channel vector is generated as follows:

$$h_l(t) = h_l^c(t) + jh_l^s(t) \tag{4.26}$$

$$h_l^c(t) = \frac{2}{\sqrt{M}} \sum_{n=1}^{M} \cos(\psi_n).\cos(\omega_d t \cos\alpha_n + \phi) \tag{4.27}$$

$$h_l^s(t) = \frac{2}{\sqrt{M}} \sum_{n=1}^{M} \sin(\psi_n).\cos(\omega_d t \cos\alpha_n + \phi), \tag{4.28}$$
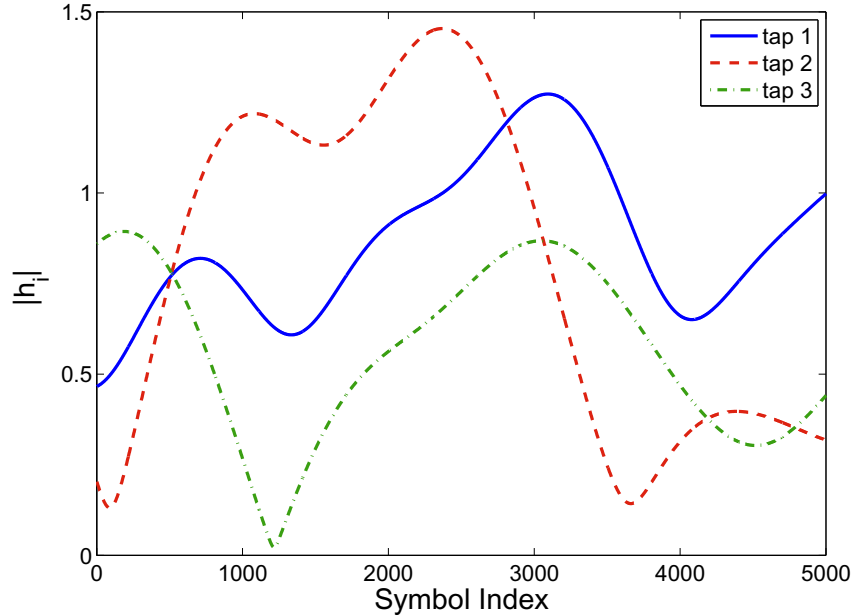
Figure 4.2: 5000 samples of a 3 taps time-varying channel with maximum Doppler shift of $f_D = 40$Hz, sampling frequency of $T_s = 10\mu$sec and carrier frequency of 900MHz.

with $\alpha = \frac{2\pi n - \pi + \theta}{4M}$, where $\phi$, $\theta$ and $\psi_n$s are statistically independent random variables following uniform distributions over $[-\pi, \pi]$ [41]. The generated channel vector contains three independent taps with maximum Doppler shift of $f_D = 40$ Hz, sampling frequency of $T_s = 10$ $\mu$sec and carrier frequency of 900 MHz. In the channel simulator model given by (4.27), and (4.28), $M$ is selected as 100. For each realization of the information block an independent time varying channel is generated. The randomly generated channel taps are weighted by $[0.407\ 0.815\ 0.407]$ respectively, and the average channel energy is normalized to unity. Figure 4.2 shows one sample of the generated 3-tap channel over the duration of 5000 transmitted symbols.

Figure 4.3 shows the performance of the proposed algorithm as a function of the forgetting factor calculated in (4.22); average SNR is held constant at 7 dB. The performance of the Bayesian stack-detector is compared with that of the optimum MLSD with full channel knowledge. The optimum forgetting factor based on the above channel parameters is calculated from (4.22) and is equal to $\lambda = 0.99$. The simulation shows that when the time-varying
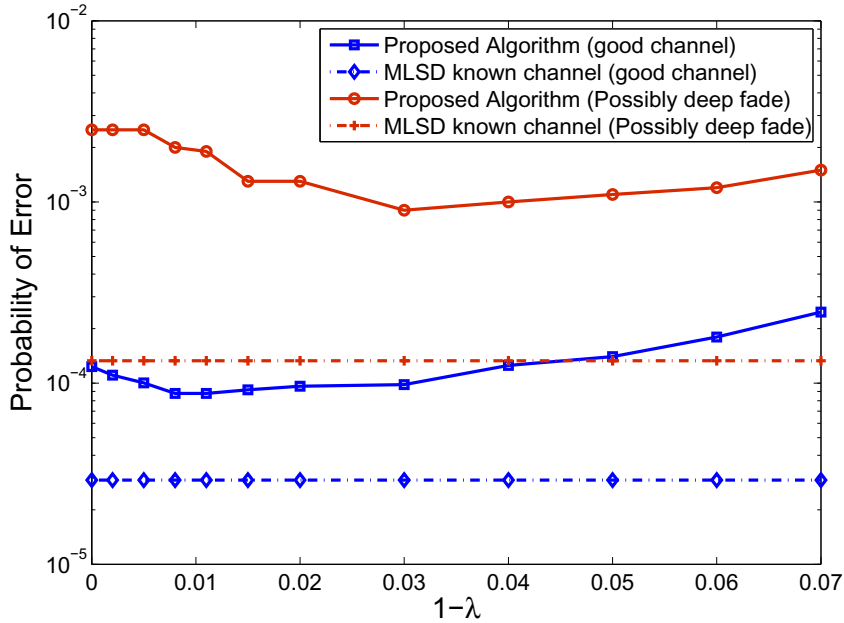
Figure 4.3: Algorithm performance v.s. forgetting factor. SNR is 7dB and the size of information bit block (N) is 50.

channel taps are in good condition (e.g. not in a deep fade), the optimum forgetting factor matches with its theoretical value. An example of the deep fade condition of the channel in Figure 4.2 is the third channel tap between the symbol indices of 1200 to 1300. Figure 4.3 also shows the effect of the forgetting factor on the algorithm performance when the channel taps may possibly be in deep fade condition. This plot shows that choosing a larger forgetting factor improves the performance in deep fade conditions. This is because we have selected the forgetting factor based on the average variance of the channel taps, $\sigma_v^2$, in (4.22). In deep fade conditions, (4.22) suggests choosing a larger forgetting factor due to rapid variations of the channel taps, or larger $\sigma_v^2$. Choosing $\lambda = 0.97$ results in strong performance for both good and deep fade channel conditions. We have fixed the forgetting factor ($\lambda$) to 0.97 for all simulations presented here.

Figure 4.4 shows the simulated performance of the proposed algorithm (BMLSD) across varying block lengths for channels not in a deep fade. The performance is compared to
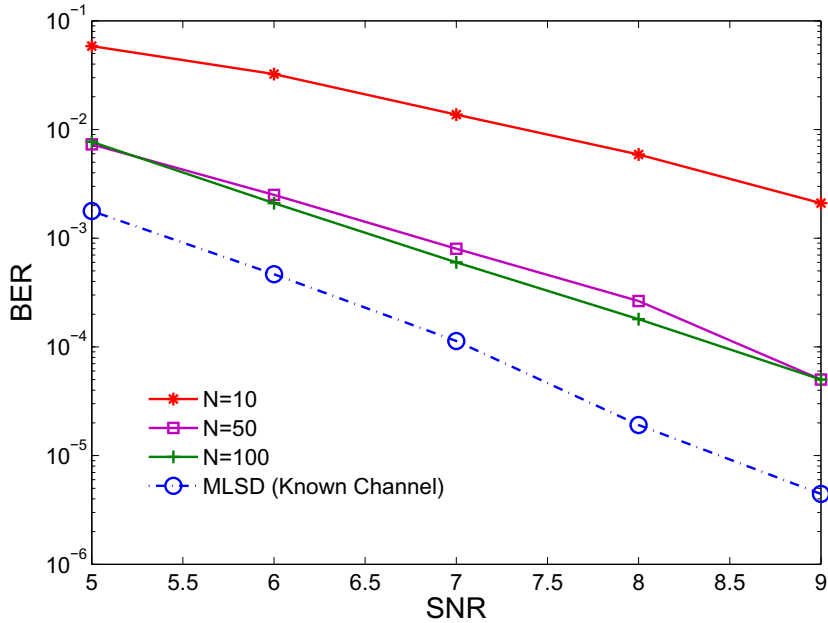
Figure 4.4: Simulated performance of the stack-based Bayesian MLSD algorithm with different data block sizes N transmitted over "good" channels.

that of the VA algorithm with full channel knowledge. The simulation shows that the algorithm performs well with information block sizes as small as 50 bits. The performance is not good for blocks of 10 bits as sufficient observations are not available to reliably estimate the covariance and cross-covariance matrices in (4.13) and (4.14). As the block size increases beyond the 50 bits, performance remains relatively constant due to the finite memory imposed by the forgetting factor.

Figure 4.5 shows the simulated performance across information block length when the channel may experience a deep fade. The performance is also shown for the case in which the channel length is unknown. In this case the algorithm calculates the metric using (4.25) where $L_{max}$ is assumed to be 5. We can see that our proposed algorithm performs approximately 4 dB better relative to the LMS Viterbi algorithm [31]. It can also work on block sizes of 20 bits, which makes it suitable for applications like packet transmission in which the data block size may be very small.
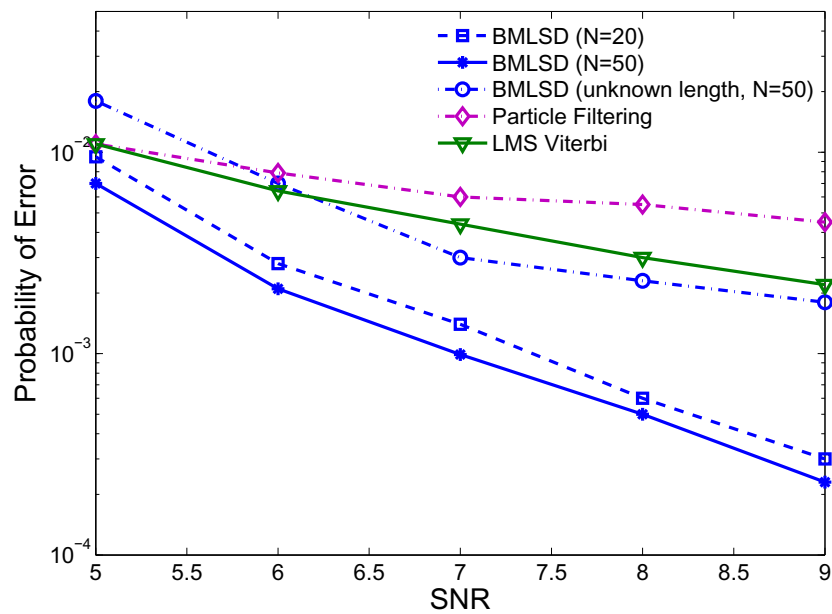
Figure 4.5: Simulated performance of the stack-based Bayesian MLSD (BMLSD) algorithm with different block sizes where the channel may possibly be in deep fade condition. The performance is evaluated for both known and unknown channel length information and the results are compared with LMS-Viterbi algorithm.

## 4.2 Modulation Classification of QAM Signals via Tree Search

In this section, we address the challenge of modulation classification in an unknown dispersive environment. A bank of M-algorithm tree search blocks, or deterministic particle filters (DPF), is used to jointly estimate the communication channel and data sequence for each possible modulation scheme, and the best path metrics from each DPF form a feature vector. Maximum likelihood modulation classification is performed and makes use of the statistics of the feature vector under different modulation schemes. Simulation results show that the algorithm can successfully classify the observed modulation schemes with as few as 50 symbol observations, making the algorithm practical under time-varying conditions, as well. Additionally, since the communication channel and the transmitted sequence are estimated in the modulation classification process, the proposed algorithm is a natural choice for joint channel estimation, data detection, and modulation classification.

### 4.2.1 Introduction

AMC has been studied for decades considering both analog and digital modulation, as well as single and multi-carrier signals. AMC approaches are categorized in to two main classes [42, 43] : likelihood-based (LB) approaches and feature-based (FB) approaches. In likelihood-based approaches, likelihood functions of the received signal are evaluated based on different possible modulation scheme assumptions, and the likelihood values are used for classification. In feature-based approaches, one or several features of the unknown signal are used in neural network (NN) or hidden Markov model (HMM) classifiers. Many of the proposed approaches for modulation classification cannot be implemented for classification of different QAM constellations. As an example, cyclostationary based approaches that implement cyclic autocorrelation function (CAF), spectral correlation density (SCD), and spectral correlation function (SCF) as features for classification, generate the same feature values for different type of QAM signals [43].

Early work on QAM modulation classification can be found in [44] and [45]. Polydoros

[44] considers modulation classification of binary phase shift keying (BPSK) and quadrature phase shift keying (QPSK) signals in the presence of additive white Gaussian noise using a square-law classifier and a phase-based classifier. Benvenuto [45] considers modulation classification of voice band data signals in a time-dispersive environment using the constant modulus algorithm (CMA) for channel equalization. The amplitude change at the output of the blind equalizer is used for classification. A likelihood ratio test technique for classification of $M$ symbol phase shift keying (MPSK) signals in additive white Gaussian noise is proposed in [46]. Wei et al. [47, 48] propose a maximum likelihood approach for QAM modulation classification in which unknown QAM signals are classified in additive white Gaussian noise conditions. Higher order statistics for QAM modulation classification have been implemented in [49,50] where in [49], fourth order cumulants are used for classification of QAM signals in the presence of dispersive channels and in [50], eighth order cumulants have been used for classification of high density QAM modulation schemes in AWGN conditions. While HOS approaches are robust with respect to carrier phase, frequency offset, and impulsive non-Gaussian noise, they require a large sequence of observations for higher order statistics estimation. Statistical moments for QAM modulation classification have been introduced in [51] and [52] where [51] uses power moments for modulation classification in AWGN conditions. Cyclostationary based blind equalization techniques that implement second order statistics of the signal for channel estimation are used in [52] to combat the channel effects, and Zernike moments of the equalized signal are used for classification. The algorithm proposed in [52] works well with a small number of signals, but it uses a SIMO model that may not be applicable to all problems. Recently, statistical sampling and Gibbs sampling have been implemented for QAM constellation classification under ISI conditions [53], [54]. Because of the similarities between Bayesian tree search algorithms and Gibbs sampling, we have compared our proposed approaches with a Gibbs sampling classifier [54].

Among all the approaches proposed for AMC, only a few are designed under ISI conditions. Since wireless signals are often observed over frequency selective channels, we consider

modulation classification in a dispersive environment. The most common approach to mitigating inter-symbol interference is through channel equalization. In this scenario, however, the channel cannot be estimated using conventional channel estimation approaches since there is no training sequence between the transmitter and the receiver. Blind equalization techniques provide solutions for these scenarios, but most of the developed blind equalization approaches require knowledge of the signal modulation scheme, which is not available. This makes the AMC problem especially challenging, particularly for dense modulation schemes such as 8QAM and 16QAM. We have considered the problem of joint classification, equalization and sequence detection of a QAM signal with an unknown constellation scheme. We use a tree search algorithm that searches through possible transmitted sequences from a bank of possible modulation schemes. The feature required for classification is extracted from the best path metric, and based on statistical properties of the feature under different conditions, we classify the constellation scheme.

In what follows, we first explain the system model and our assumptions. We then explain the proposed DPF algorithm that is used for joint channel equalization, data detection, and modulation classification of the received signal. The feature extraction from the most probable path is then discussed, and the analytical statistics of the features are evaluated. Finally, classification of the extracted features using a likelihood ratio test is explained.

### 4.2.2   System Model

Figure 4.6 shows the block diagram of a typical communication system with a modulation classifier in the presence of an unknown dispersive channel. The transmitted sequence is observed at the output of an unknown dispersive channel with additive noise. The receiver does not have information about the modulation type of the transmitted signal. The first step is to remove the channel effects from the received signal. This must be accomplished through blind equalization, since there is no training sequence for setting up the equalizer parameters. The blind equalizer should be able to function without knowledge of the signal modulation scheme. When the inter-symbol interference is compensated, data is transferred
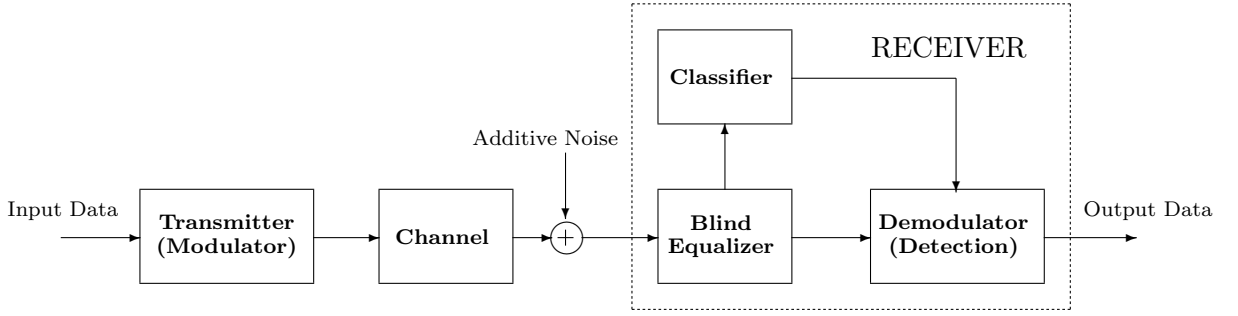
Figure 4.6: Block diagram of the overall system that shows the role of a modulation classifier in a typical receiver. The transmitted symbols are observed at the output of an unknown dispersive channel with additive noise. The blind equalizer compensates channel inter-symbol interference and the classifier identifies the modulation scheme. The demodulator demodulates and extracts the transmitted information bits using the information from the classifier units.

to the modulation classifier unit. The modulation classifier identifies the modulation scheme of the received signal, which is necessary for detection of transmitted information bits.

We have implemented DPF as a technique for blind equalization. Since DPF needs information about the modulation scheme of the transmitted sequence, a bank of DPF blind equalizers is used, and each is tuned to a different possible modulation scheme. The likelihoods of the data sequences detected by each DPF module form the feature vector used for modulation classification.

Assuming that the channel is static or varying slowly with time and that the receiver has perfectly estimated the carrier frequency and the symbol rate of the received signal, we can employ a discrete-time baseband-equivalent model. Figure 4.7 shows the baseband model of the channel, as well as the blind equalizer and classifier in the baseband-equivalent model. The received signal for a block of $N$ symbols $\mathbf{z} = [z_1 \ z_2 \ \ldots \ z_N]^T$ is given by

$$\mathbf{z} = \mathbf{X}^H \mathbf{h} + \mathbf{w}, \tag{4.29}$$

where $\mathbf{h} = [h_0 \ h_1 \cdots h_{L-1}]^T$ is the length-$L$ channel impulse response, $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ldots \mathbf{x}_N]$, $\mathbf{x}_n = [x_n \ x_{n-1} \cdots x_{n-L+1}]^T$ is formed from the transmitted symbols in the time interval

$[n - L + 1, \; n]$, the superscript $H$ denotes the Hermitian operator, and $\mathbf{w} = [w_1 \; w_2 \ldots w_N]^T$ denotes additive noise. We make the following assumptions in our model and throughout this section:

- The transmitted symbols have unit energy and are independent and identically distributed (i.i.d.), e.g.

$$E\{x_n x_m^*\} = \delta_{nm}$$

  for all $n$ and $m$, where $*$ denotes the complex conjugate operator, and $\delta_{nm}$ is the Kronecker delta.

- The additive noise samples $w_n$ are i.i.d. and follow a Gaussian distribution with mean zero and known variance $\sigma_w^2$.

- The receiver has a reliable estimate of the channel length $L$.

- The constellation scheme of the transmitted signal, denoted by $\mathcal{A}$ and composed of $M$ symbols, belongs to a known space $\Omega$ of possible constellation schemes, e.g.

$$\mathcal{A} \in \Omega, \quad \text{where} \quad \Omega = \{\mathcal{A}_1, \cdots, \mathcal{A}_K\}.$$

- The vector of channel coefficients $\mathbf{h}$ follows a multivariate complex Gaussian distribution with unknown expected value $\overline{\mathbf{h}}$ and covariance matrix $C$ and unit energy $E\{\mathbf{h}^H \mathbf{h}\} = 1$. We use $\mathcal{N}_c(\;\mathbf{h}\;;\;\overline{\mathbf{h}}\;,\;C)$ to denote the probability density function (PDF) of $\mathbf{h}$.

In our model, the receiver classifies the modulation scheme of the transmitted signal based on observation of a block of $N$ transmitted symbols. Given the observed signal vector $\mathbf{z}$, the maximum likelihood estimate of the modulation scheme is given by

$$\hat{\mathcal{A}} = \max_{i=1,..K} P(\mathcal{A}_i | \mathbf{z}), \tag{4.30}$$
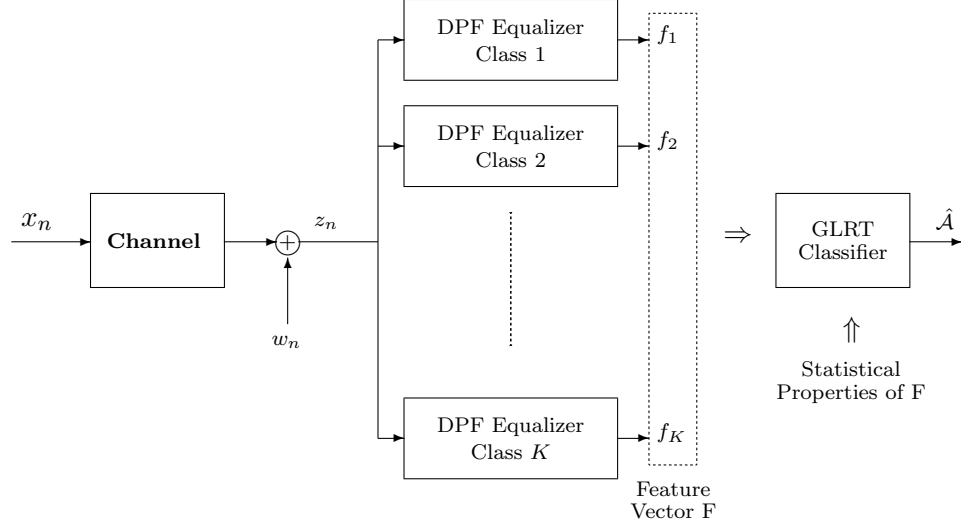
Figure 4.7: The transmitted symbols at time $n$, denoted by $x_n$, are observed at the output of an unknown dispersive channel with additive noise. $\mathbf{h}$ denotes the unknown ISI channel, $w_n$ denotes additive white Gaussian noise (AWGN) with variance $\sigma_w^2$, and $z_n$ denotes the received signal. A bank of DPF-based blind equalizers, each assuming a different modulation scheme, is used to generate the elements $f_i$, $i = 1, \ldots, K$, of the feature vector $F$. The modulation scheme of the received signal is classified as $\hat{\mathcal{A}}$ based on the values in $F$.

where $\mathcal{A}_i$ denotes the $i$th of $K$ possible modulation schemes. Using Bayes' Theorem, (4.30) can be written as

$$
\begin{aligned}
P(\mathcal{A}_i|\mathbf{z}) &= \frac{p(\mathbf{z}|\mathcal{A}_i)\,P(\mathcal{A}_i)}{p(\mathbf{z})} \\[2mm]
&= \frac{P(\mathcal{A}_i)}{p(\mathbf{z})} \sum_{j=1}^{M_i^N} p(\mathbf{z}|\mathbf{x}_{1:N}^{(i,j)}, \mathcal{A}_i)\,P(\mathbf{x}_{1:N}^{(i,j)}|\mathcal{A}_i),
\end{aligned}
$$ (4.31)

where $\mathbf{x}_{1:N}^{(i,j)} = [x_1^{(i,j)}\ x_2^{(i,j)}\ \ldots\ x_N^{(i,j)}]^T$, $j = 1, \ldots, M_i^N$, denotes all possible length-$N$ symbol sequences drawn from modulation scheme $\mathcal{A}_i$. Assuming that all modulation schemes are equally likely to be observed a priori and that all symbols within each modulation scheme are equally likely to be transmitted, we have $P(\mathcal{A}_i) = 1/K$, and $P(\mathbf{x}_{1:N}^{(i,j)}|\mathcal{A}_i) = 1/M_i^N$. Since $p(\mathbf{z})$ is constant across all modulation schemes, it can be eliminated without loss of

information, and (4.31) can be written as

$$P(\mathcal{A}_i|\mathbf{z}) \propto \sum_{j=1}^{M_i^N} \frac{1}{M_i^N} \, p(\mathbf{z}|\mathbf{x}_{1:N}^{(i,j)}, \mathcal{A}_i). \tag{4.32}$$

Exact computation of (4.32) as a metric for classification requires computation of the received sequence likelihood conditioned on each of the $M_i^N$ possible transmitted sequences, which results in an impractically high level of computational complexity. In order to maintain a feasible complexity level in the proposed algorithm, we replace the sum over all possible transmitted sequences with the maximum element in the summation:

$$m_i = \max_{(j=1..M_i^N)} \, p(\mathbf{z}|\mathbf{x}_{1:N}^{(i,j)}, \mathcal{A}_i). \tag{4.33}$$

While the DPF-based blind equalizer is not guaranteed to find the maximum in (4.33), the value of $m_i$ is well approximated by the metric of the most likely path identified by the DPF algorithm.

For blind equalization, the particles propagated by the DPF algorithm at time $n$ are given by the set $\{\mathbf{x}_{1:n}^{(i,j)}\}$ for $j = 1, \ldots, P$. The metric $m_i$ is approximated for each possible modulation scheme, e.g. for $i = 1, \ldots, K$, using a bank of DPF modules, each of which propagates a reduced number $P$ of possible transmitted sequences, or particles, through the detection process. The feature vector element $f_i$, which represents the $i$th possible modulation scheme for classification, is extracted from the particle that has the largest metric in the $i$th DPF block.

### 4.2.3 Feature Extraction

When (4.33) is approximated using stochastic PF, the new sample $x_n^{(i)}$ that is added to the particle $\mathbf{x}_{1:n-1}^{(i,j)}$ at each time update $n$ is drawn stochastically from an importance function

that is proportional to the conditional likelihood of $z_n$ [14]. When $x_n^{(i)}$ comes from a discrete sample space, however, it has been recommended that $\mathbf{x}_{1:n-1}^{(i,j)}$ be updated deterministically by appending all possible sample values and generating all possible updated particles [16]. In the proposed modulation classification scheme, we employ DPF to find the specific particle (transmitted sequence) for each possible modulation scheme $\mathcal{A}_i$, $i = 1, \ldots, K$, that maximizes the metric defined in (4.33). At each stage of the DPF algorithm, only a fixed number $P$ of particles, $\mathbf{x}_{1:n}^{(i,j)}$, $j = 1, \ldots, P$, are retained. Each time a new observation is obtained, the existing particles (sequences) are extended to all possible children, and the metrics of the extended paths are computed. The $P$ particles with the largest metrics are retained for further extension.

Under the assumption of a Gaussian channel, the update to the sequence likelihood (metric) with observation $z_n$ can be written as

$$
\begin{aligned}
& \mathrm{p}(z_n | x_n^{(i)}, \mathbf{x}_{1:n-1}^{(i,j)}, z_1, z_2, \ldots, z_{n-1}, \mathcal{A}_i) \\
&= \int \mathrm{p}(z_n | x_n^{(i)}, \mathbf{x}_{1:n-1}^{(i,j)}, z_1, z_2, \ldots, z_{n-1}, \mathcal{A}_i, \mathbf{h}) \\
&\quad \times \mathrm{p}(\mathbf{h} | z_n, x_n^{(i)}, \mathbf{x}_{1:n-1}^{(i,j)}, z_1, z_2, \ldots, z_{n-1}, \mathcal{A}_i) d\mathbf{h} \\
&= \int \mathcal{N}_c(z_n; \tilde{\mathbf{x}}_n^{(i,j)H} \mathbf{h}, \sigma_w^2) \times \mathcal{N}_c(\mathbf{h}; \hat{\mathbf{h}}_n^{(i,j)}, C_n^{(i,j)}) d\mathbf{h} \\
&= \mathcal{N}_c(z_n; \tilde{\mathbf{x}}_n^{(i,j)H} \hat{\mathbf{h}}_n^{(i,j)}, \sigma_w^2 + \tilde{\mathbf{x}}_n^{(i,j)H} C_n^{(i,j)} \tilde{\mathbf{x}}_n^{(i,j)}),
\end{aligned}
\tag{4.34}
$$

where $\tilde{\mathbf{x}}_n^{(i,j)} = [x_n^{(i)} \ x_{n-1}^{(i,j)} \ \cdots x_{n-L+1}^{(i,j)}]^T$ denotes the new children generated by appending $x_n^{(i)}$ to the $j$th previously existing particle, and $N_c(\mu, \sigma^2)$ denotes the probability density function of a complex Gaussian random variable with mean $\mu$ and variance $\sigma^2$. The estimated mean and covariance matrix of the channel, denoted by $\hat{\mathbf{h}}_n^{(i,j)}$ and $C_n^{(i,j)}$ respectively, are computed

using a one-step prediction Kalman filter [1]:

$$\hat{\mathbf{h}}_n^{(i,j)} = \hat{\mathbf{h}}_{n-1}^{(i,j)} + \frac{C_{n-1}^{(i,j)}\tilde{\mathbf{x}}_n^{(i,j)}(z_n - \tilde{\mathbf{x}}_n^{(i,j)H}\hat{\mathbf{h}}_{n-1}^{(i,j)})}{\tilde{\mathbf{x}}_n^{(i,j)H}C_{n-1}^{(i,j)}\tilde{\mathbf{x}}_n^{(i,j)} + \sigma_w^2} \tag{4.35}$$

$$C_n^{(i,j)} = C_{n-1}^{(i,j)} - \frac{C_{n-1}^{(i,j)}\tilde{\mathbf{x}}_n^{(i,j)}\tilde{\mathbf{x}}_n^{(i,j)H}C_{n-1}^{(i,j)}}{\tilde{\mathbf{x}}_n^{(i,j)H}C_{n-1}^{(i,j)}\tilde{\mathbf{x}}_n^{(i,j)} + \sigma_w^2}. \tag{4.36}$$

Using (4.34), the updated metric at time $n$, denoted by $m_n^{(i,j)}$ for a new generated particle $\mathbf{x}_{1:n}^{(i,j)}$, is given by

$$m_n^{(i,j)} = \mathrm{p}(z_n|\mathbf{x}_{1:n}^{(i,j)}, z_1, \ldots, z_n, \mathcal{A}_i) \times m_{n-1}^{(i,j)}. \tag{4.37}$$

Once the DPF modules have reached $n = N$ and identified the most likely full-length sequence for each possible modulation scheme, we must extract from the results of each DPF module a feature that allows us to distinguish among the possible modulation schemes. We have selected as our feature the squared Euclidean distance between the true received sequence vector and the expected noise-free received sequence vector associated with the most likely full-length particle:

$$f_i = \|\mathbf{z} - \mathbf{X}^{(i)H}\hat{\mathbf{h}}_N^{(i,j)}\|^2 \quad i = 1, \ldots, K, \tag{4.38}$$

where $\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)}\ \mathbf{x}_2^{(i)}\ \ldots\ \mathbf{x}_N^{(i)}]$ is the estimated signal matrix, and $\mathbf{x}_n^{(i)} = [x_n^{(i)}\ x_{n-1}^{(i)}\ \ldots\ x_{n-L+1}^{(i)}]^T$ is the estimated signal vector The feature vector $F = [f_1\ f_2\ \cdots\ f_K]$ is given as input to the maximum likelihood classifier. Table 4.1 summarizes the proposed algorithm used in each of the $K$ DPF modules.

Table 4.1: Feature Generation as Performed in the $i$th DPF Module

---

- for $n = 1, \ldots, N$,
    - for $j = 1, \ldots, P$,
       - for $l = 1, \ldots, M_i$,
          - extend $\mathbf{x}_{1:n-1}^{(i,j)}$ with the $l$th symbol of
            constellation $\mathcal{A}_i$
         - update the channel estimate using (4.35) and (4.36)
         - update the particle (sequence) metric using (4.37)
       - retain the $P$ particles with the largest metrics
  - extract the feature element $f_i$ from the metric of the most
   probable particle using (4.38).

---

## 4.2.4 Classification

Following generation of the feature vector, modulation classification is performed based on the maximum likelihood estimate of the feature vector:

$$\hat{\mathcal{A}} = \arg \max_{\mathcal{A}_k} p(F|\mathcal{A}_k) \quad k = 1, \ldots, K. \tag{4.39}$$

Using (4.39) requires approximation of the conditional pdf $p(F|\mathcal{A}_k)$. One approximation method is to train the classifier and empirically estimate the conditional pdf based on the training results. This approach has some disadvantages in practice, however, since training is not available in many scenarios (surveillance and cognitive radio, for example). Additionally, a change in any parameter, such as the set of possible modulation schemes, requires that the classifier be retrained. To avoid these disadvantages, we develop an analytical approximation to the conditional pdf in (4.39). Each element of the vector $F$ is generated by a separate DPF block, as shown in Figure 4.7. Assuming independence among the elements of $F$, we have

$$p(F|\mathcal{A}_k) = \prod_{i=1}^{K} p(f_i|\mathcal{A}_k). \tag{4.40}$$

The problem is thus simplified to approximating $p(f_i|\mathcal{A}_k)$ for $i = 1, \ldots, K$ and $k = 1, \ldots, K$.

The model used to derive the statistical properties of the feature vector elements $f_i$ is shown in Figure 4.8. The transmitted symbols are from modulation scheme $\mathcal{A}_k$, and the DPF module generates data sequences (particles) from modulation scheme $\mathcal{A}_i$. Note that $k$ is not necessarily equal to $i$. The DPF identifies the most likely sequence and the associated MMSE channel estimate. For our analysis, we assume that the DPF has found the best achievable MMSE channel estimate, which is true when infinite observations are available and all possible particles are propagated. This assumption is reasonable in practical scenarios when the observation window and number of propagated particles are sufficiently large. Under these conditions, the MMSE channel estimate $\hat{\mathbf{h}}^{(i)}$ and the associated mean square error (MSE) generated by the most likely particle (transmitted sequence) under assumed constellation $\mathcal{A}_i$ are given by [1]

$$\hat{\mathbf{h}}^{(i)} = \Sigma_{\mathbf{X}^{(i)}}^{-1} \Sigma_{\mathbf{X}^{(i)}\mathbf{X}} \mathbf{h} \approx \hat{\mathbf{h}}_N^{(i)} \tag{4.41}$$

and

$$\text{MMSE} = \mathbf{h}^H (\Sigma_{\mathbf{X}} - \Sigma_{\mathbf{X}^{(i)}\mathbf{X}} \Sigma_{\mathbf{X}^{(i)}}^{-1} \Sigma_{\mathbf{X}\mathbf{X}^{(i)}}) \mathbf{h} + N\sigma_w^2, \tag{4.42}$$

where $\Sigma_{\mathbf{X}}$ denotes the auto correlation matrix of the matrix $\mathbf{X}$ and $\Sigma_{\mathbf{X}\mathbf{X}^{(i)}}$ denotes the cross correlation matrix of the matrices $\mathbf{X}$ and $\mathbf{X}^{(i)}$, defined as

$$\Sigma_{\mathbf{X}} = E\{\mathbf{X}\mathbf{X}^H\} \tag{4.43}$$

$$\Sigma_{\mathbf{X}\mathbf{X}^{(i)}} = E\{\mathbf{X}\mathbf{X}^{(i)H}\}. \tag{4.44}$$

Assuming the average energy of each signal constellation is equal to 1 and that the
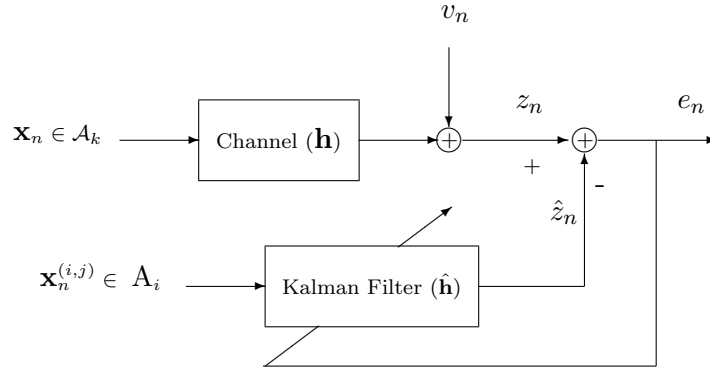
Figure 4.8: MMSE channel estimation model. The particles from $\mathcal{A}_i$ are filtered by the adaptive channel estimator, and the result is compared to the observed signal. Mean square error is used to update the channel estimate.

elements of the true transmitted sequence and the estimated sequence are i.i.d. such that

$$E\{x_n x_m^*\} \quad = \quad \delta[n - m] \tag{4.45}$$

$$E\{x_n^{(i)} x_m^{(i)*}\} \quad = \quad \delta[n - m] \tag{4.46}$$

$$E\{x_n^{(i)} x_m^*\} \quad = \quad \rho_k^{(i)} \delta[n - m], \tag{4.47}$$

we have

$$\Sigma_{\mathbf{X}} \quad = \quad N\mathbf{I}_L \ \text{ and} \tag{4.48}$$

$$\Sigma_{\mathbf{X}^{(i)}\mathbf{X}} \Sigma_{\mathbf{X}^{(i)}}^{-1} \Sigma_{\mathbf{X}\mathbf{X}^{(i)}} \quad = \quad \rho_k^{(i)} N\mathbf{I}_L, \tag{4.49}$$

where $\mathbf{I_L}$ denotes the identity matrix $(L \times L)$, and we define $\rho_k^{(i)}$ as the normalized cross correlation between the true symbols $x_n$ from $\mathcal{A}_k$ and those estimated from $\mathcal{A}_i$. With the additional assumption of unit channel energy, the MMSE calculated in (4.42) simplifies to

$$\text{MMSE} = N(1 - \rho_k^{(i)} + \sigma_w^2).$$

As shown in (4.34), $\mathrm{p}(z_n|x_{1:n}^{(i)}, z_1, z_2, \ldots, z_{n-1}, \mathcal{A}_i)$ has a Gaussian distribution. Since the channel estimate is generated based on the same conditioning, the error term $z_n - \mathbf{x}_n^{(i)T}\hat{\mathbf{h}}_N^{(i)}$ that appears in (4.38) also follows a Gaussian distribution. The error term at time $n$ has mean zero and variance equal to a scaled version of the MMSE, e.g.

$$\mathrm{p}(z_n - \mathbf{x}_n^{(i)H}\hat{\mathbf{h}}_N^{(i)}|x_{1:n}^{(i)}, z_1, z_2, \ldots, z_{n-1}, \mathcal{A}_i)$$

$$= \mathcal{N}_c(z_n - \mathbf{x}_n^{(i)H}\hat{\mathbf{h}}_N^{(i)};\ 0\ ,\ 1 - \rho_k^{(i)} + \sigma_w^2).$$

Given the form of the feature $f_i$ in (4.38), we can conclude that the normalized feature $f_i/(1 - \rho_k^{(i)} + \sigma_w^2)$ has a Chi-square distribution with $2N$ degrees of freedom. Hence, (4.40) is the product of $K$ Chi-square PDFs.

The only remaining unknown parameter is $\rho_k^{(i)}$, which depends upon the relative geometries of the assumed modulation scheme and the true modulation scheme. We consider three possible cases, which are detailed below.

**Case I:** When the assumed modulation scheme $\mathcal{A}_i$ is equal to the true modulation scheme $\mathcal{A}_k$, we can assume (under sufficiently large $P$ and with sufficient window size) that the best particle generated by the PDF is equal to the true transmitted sequence up to a phase ambiguity, e.g.

$$\mathbf{X}^{(i)} = e^{j\theta}\mathbf{X} \quad \text{where} \quad \mathbf{X}^{(i)} \in \mathcal{A}_i\ , \quad \mathbf{X} \in \mathcal{A}_k\ , \quad \text{and} \quad i = k. \tag{4.50}$$

For example, suppose that the constellation scheme ($\mathcal{A}$) of the transmitted sequence is 4QAM with symbols defined by $s_1 = (1 + j)/\sqrt{2}$, $s_2 = (1 - j)/\sqrt{2}$, $s_3 = (-1 - j)/\sqrt{2}$ and $s_4 = (-1 + j)/\sqrt{2}$. If the transmitted sequence is $\{s_3\ s_1\ s_2\ s_3\ s_2\ s_4\}$, then the following

particles may become the particles with the highest metric:

$$\{s_3\ s_1\ s_2\ s_3\ s_2\ s_4\} \qquad \text{for} \quad \theta = 0$$

$$\{s_1\ s_3\ s_4\ s_1\ s_4\ s_2\} \qquad \text{for} \quad \theta = \pi$$

$$\{s_4\ s_2\ s_3\ s_4\ s_3\ s_1\} \qquad \text{for} \quad \theta = \pi/2$$

$$\{s_2\ s_4\ s_1\ s_2\ s_1\ s_3\} \qquad \text{for} \quad \theta = -\pi/2.$$

We see that for all the above particles, we have

$$\Sigma_{\mathbf{X}^{(i)}\mathbf{X}} = N \times I_{L \times L} \qquad \text{and} \qquad \Sigma_{\mathbf{X}^{(i)}}^{-1} = \Sigma_{\mathbf{X}}^{-1} = \frac{1}{N} \times I_{L \times L}.$$

The channel estimated for these particles using (4.41) is then given by

$$\hat{\mathbf{h}}^{(i)} = (e^{j\theta})^{-1}\mathbf{h},$$

and feature $f_i$ in (4.38) has its minimum possible value as

$$
\begin{aligned}
\|\mathbf{z} - \mathbf{X}^{(i)H}\hat{\mathbf{h}}_N^{(i)}\|^2 &= \|\mathbf{z} - (e^{j\theta})^{-1}\mathbf{X}^H\mathbf{h}(e^{j\theta})\|^2 \\
&= \|\mathbf{w}\|^2.
\end{aligned}
$$

Additionally, by the definition of $\rho_k^{(i)}$ (4.47), we have $\rho_k^{(i)} = \rho_i^{(k)} = 1$ for $i = k$.

**Case II:** When the assumed modulation scheme is related to the true modulation scheme in a manner such that, for every sequence matrix $\mathbf{X} \in \mathcal{A}_k$, there exists a particle matrix $\mathbf{X}^{(i)} \in \mathcal{A}_i$ that satisfies $\mathbf{X}^{(i)} = \alpha e^{j\theta}\mathbf{X}$, then we can assume that the best particle is given

by $\mathbf{X}^{(i)} = \alpha e^{j\theta}\mathbf{X}$. In this case

$$\Sigma_{\mathbf{X}^{(i)}\mathbf{X}} = \alpha e^{j\theta} N \times I_{L \times L} \qquad \text{and} \qquad \Sigma_{\mathbf{X}^{(i)}}^{-1} = \frac{1}{\alpha^2 N} \times I_{L \times L}.$$

The associated channel estimate (4.41) is then given by

$$\hat{\mathbf{h}}^{(i)} = (\alpha e^{-j\theta})^{-1}\mathbf{h}$$

and the feature $f_i$ in (4.38) is given by

$$
\begin{aligned}
\|\mathbf{z} - \mathbf{X}^{(i)H}\hat{\mathbf{h}}_N^{(i)}\|^2 &= \|\mathbf{z} - (e^{j\theta})^{-1}\mathbf{X}^H\mathbf{h}(e^{j\theta})\|^2 \\
&= \|\mathbf{w}\|^2.
\end{aligned}
$$

In this case, the true symbol constellation is equal to a scaled and rotated version of a subset of the assumed symbol constellation. Hence, the size of the assumed signal constellation must be at least as large as the size of the true constellation. For example, BPSK as the true modulation scheme with symbols of $\{+1, -1\}$ and 4QAM as the assumed modulation scheme with symbols of $\{(1 + j)/\sqrt{2}, (1 - j)/\sqrt{2}, (-1 - j)/\sqrt{2}, (-1 + j)/\sqrt{2}\}$ would fall within this case. In this case, for $\alpha = \sqrt{2}$ and each value of $\theta$ in $\{0, \pi, \pi/2, \pi/2\}$ there are particles from the 4QAM modulation scheme that minimize the value of the feature to its global minimum $\|\mathbf{w}\|^2$ .

Note that, in the case that DPF finds the global minimum, we have $\rho_k^{(i)} = 1$. However, $\rho_k^{(i)}$ is not necessarily equal to $\rho_i^{(k)}$, since every symbol in the larger assumed constellation scheme cannot be written as a scaled and rotated version of a symbol in the smaller true constellation scheme.

**Case III:** When the symbols of the true modulation scheme cannot be written as a scaled and rotated version of symbols in the assumed modulation scheme $\mathcal{A}_i$, the DPF module searches for a sequence of symbols within the assumed modulation scheme that is highly correlated with the transmitted sequence. For any symbol in $\mathcal{A}_k$, the most highly correlated symbol in $\mathcal{A}_i$ is the one that is closest in Euclidean distance. For example, if the true symbol constellation is from 4QAM ($\mathcal{A}_2$) and the assumed constellation is BPSK ($\mathcal{A}_1$), then the BPSK symbol $\{+1\}$ will be most highly correlated with 4QAM symbols whose real part is positive. In this case we only have four possible cases for transmitted symbol and estimated symbol as follows

- transmitted symbol is $+1$ and the estimated symbol is $(1+j)/\sqrt{2}$,

- transmitted symbol is $+1$ and the estimated symbol is $(1-j)/\sqrt{2}$,

- transmitted symbol is $-1$ and the estimated symbol is $(-1+j)/\sqrt{2}$ and

- transmitted symbol is $-1$ and the estimated symbol is $(-1-j))/\sqrt{2}$.

Note that each of the above cases may happen with the same prior probability of $1/4$. The correlation coefficient $\rho_k^{(i)}$ is simply calculated as the correlation between the symbols in the true constellation $\mathcal{A}_k$ and the nearest symbols in $\mathcal{A}_i$. Based on this assumption, we can calculate $\rho_2^{(1)}$ as follows:

$$
\begin{aligned}
\rho_2^{(1)} &= 1/4 \left[ 1 \times (1+j)^*/\sqrt{2} + 1 \times (1-j)^*/\sqrt{2} - 1 \times (-1+j)^*/\sqrt{2} - 1 \times (-1-j)^*/\sqrt{2} \right] \\
&= 1/\sqrt{2}.
\end{aligned}
$$

When the appropriate case has been determined and $\rho_k^{(i)}$ calculated accordingly, the probability of any realization of the feature vector can be computed as given in (4.40), and the modulation scheme can be estimated using (4.39).
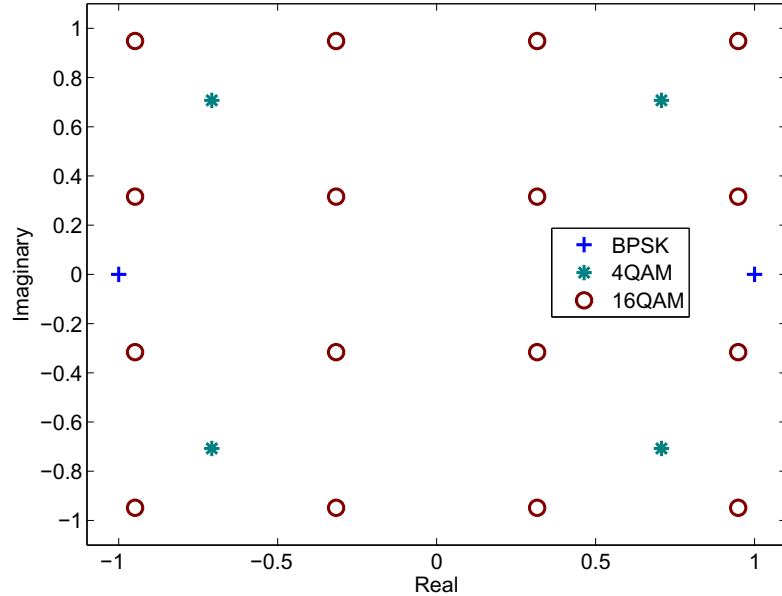
Figure 4.9: Modulation schemes used for classification: BPSK ('+'), 4QAM ('*') and 16QAM ('o').

### 4.2.5 Simulation Results and Discussion

The proposed algorithm has been simulated for three modulation schemes: BPSK, 4QAM and 16QAM, which are indexed by $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, respectively, and are depicted in Figure 4.9. The correlation coefficients $\rho_j^{(i)}$ for the chosen modulation schemes are given by $\rho_2^{(1)} = 1/\sqrt{2}$, $\rho_3^{(1)} = 2/\sqrt{10}$, and $\rho_3^{(2)} = 2/\sqrt{5}$; $\rho_j^{(i)} = 1$ for all other $i, j$ pairs. Five hundred Monte Carlo simulation runs were averaged to determine the probability of correct classification at each SNR value considered. For each run, independent transmitted symbols, channel taps, and additive noise are generated.

Figure 4.10 displays the probability of correct classification vs. signal to noise ratio (SNR) for blocks of 50, 100 and 150 symbol observations. For each run, an unknown 3-tap channel is drawn from a zero-mean multivariate Gaussian distribution, and the channel energy is normalized to 1. The number of particles retained in each DPF module is $P = 64$. For comparison, we have also simulated Gibbs algorithm for constellation classification
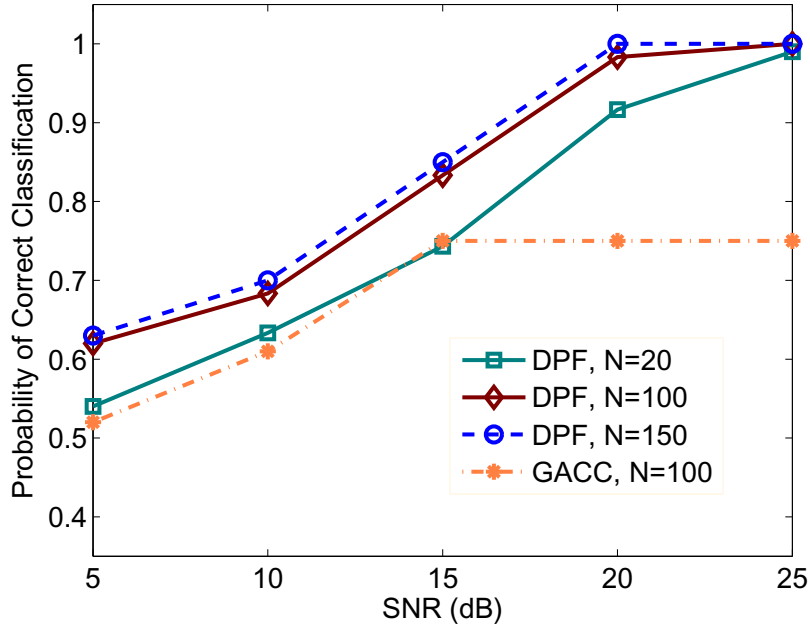
Figure 4.10: Probability of correct classification vs. SNR for the DPF and GACC modulation classification schemes.

(GACC) [54]. The burn-in period and total number of iterations for GACC are set to 100 and 300 respectively, and 100 observation symbols are used for classification; the noise variance is assumed known. Simulation results reveal that, as SNR increases, the DPF-based classifier significantly outperforms the GACC method and can achieve perfect classification with observation windows as small as 50 symbols.

Figure 4.11 shows the effect of the number of particles retained by each DPF module on classification capabilities. The observation window is fixed at 100 symbols, and $P$ varies from 16 to 64. The simulation results indicate that reducing the number of retained particles to as small as 16 has only a small effect on the performance of the classifier. Figure 4.12 shows the performance of the algorithm as a function of channel length. The observation window is fixed at 100 symbols, and $P$ is fixed at 64. Probability of correct classification is presented vs. SNR for 2, 3 and 5-tap channels. While performance is nearly equal for 2 and 3-tap channels, degradation is apparent when the channel length grows to 5. We hypothesize that performance can be improved through a suitable increase in $P$ and in the
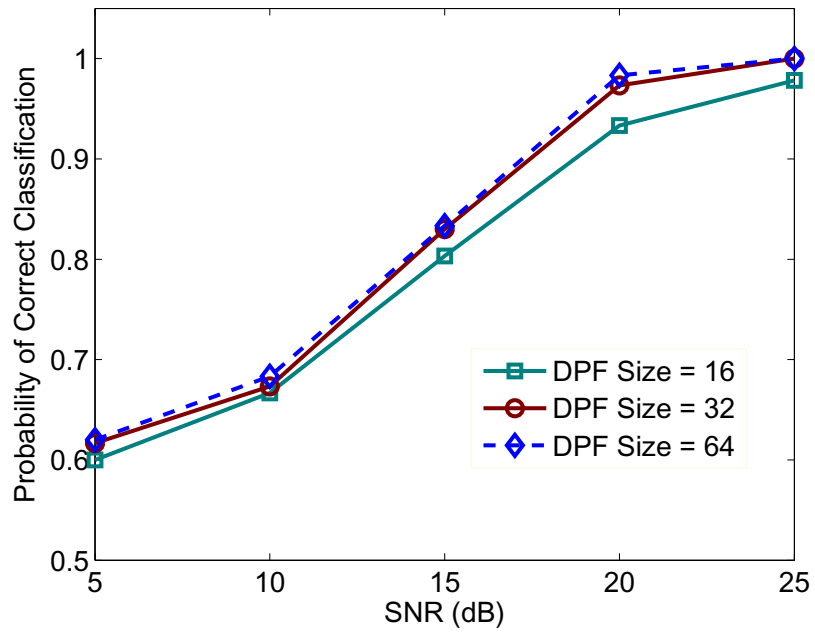
Figure 4.11: Probability of correct classification vs. SNR for varying $P$, number of retained particles.

length of the observation window.
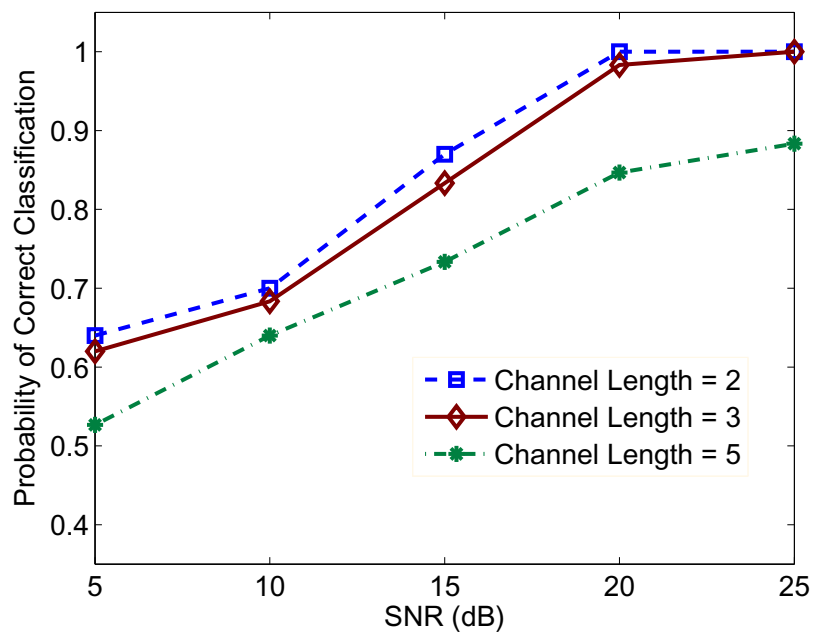
Figure 4.12: Probability of correct classification vs. SNR for varying channel length $L$.

# Chapter 5: Target Tracking via Tree Search

In Chapter 2, we introduced tree search techniques as a general solution to detection and estimation problems modeled using a DSSM. In this chapter, we extend the application of tree search techniques to target tracking. In a target tracking problem, we seek to estimate target parameters of interest such as location and speed from noisy observations. Using DSSM, variations of the target parameters in time are modeled by the state process model and sensor observations are modeled by the observation model. A target track is defined as a sequence of successive target states, starting from the initially estimated target state and ending at the current observation time. By defining the nodes of a search tree as target states, a target track is modeled by a path through the search tree, starting from the initial node (root) and ending at a leaf. Therefore, tree search techniques can be applied to this kind of problem, as well. In a target tracking problem, the state vector often contains more elements than that of a decoding problem in communication systems and makes the complexity of the tree search intractable if partial navigation techniques are not applied. Additionally, real-time estimation of the target state is necessary in a wide range of tracking applications including radar and sonar. The stack-based tree search algorithm, discussed in Section 3.2, substantially reduces the tree search complexity and is a good candidate for target tracking applications. In this chapter, we develop a stack-based tree search technique for target tracking applications. To develop a tree search algorithm, we need to model the problem using DSSM, define a search tree structure, and develop a metric associated to each path.

A background on the target tracking problem and the existing techniques for tracking is provided in Section 5.1. In Section 5.2, we describe the stack-based tree search approach for tracking a single target in clutter. We propose and evaluate an embedded track validation technique in Section 5.3, and we evaluate the performance of the tree-search based tracking

algorithm using the SEABAR'07 dataset in Section 5.4. Extension of the tree-search tracker to follow multiple targets is presented in Section 5.5, and an analysis of the path metric behavior in the stack-based tracker is presented in Section 5.6.

## 5.1   Existing Approaches to Target Tracking

Tracking moving targets based on noisy sensor observations is studied as an estimation problem and modeled using a dynamic state space model. Target tracking problems appear in many scientific and engineering applications including sonar, radar, biomedical imaging, computer vision, etc. In an active sonar/radar system, a transmitter sends signals, or pings, in the surveillance region, and receivers look for the return signals. The measured signals, known as contacts, are reflected either from targets or from other undesired sources. In the latter case, the measured signal is known as a false alarm or clutter. The existence of clutter adds uncertainty in assigning the observations to the targets, and data association techniques have been developed to address this problem.

A review of the existing approaches to a general detection and estimation problem modeled using DSSM is discussed in Chapter 2. In this section, we review the existing techniques to this problem which also incorporate data association. We first review existing approaches to data association that can be combined with the general estimation approaches discussed in Chapter 2. Then, we discuss more advanced techniques in the context of a target tracking problem. A comprehensive review of multiple target tracking approaches can be found in [55, 56].

The early work on target tracking that addresses the challenge of data association in a single target tracking problem was introduced in [57]. The approach described in [57] splits a track whenever more than one measurement is present. The likelihood function for each track is evaluated, and the tracks with likelihoods less than a predefined threshold are deleted. The nearest neighbor Kalman filter (NNKF) approach was introduced by Singer, Sea and Stein [58–60] and augments the traditional Kalman filter to address data association uncertainty. In this approach, among all the measurements of each scan, the one that has

the smallest statistical distance to the current estimated target location is considered as a true target measurement, and the others are considered as clutter. The global nearest neighbor algorithm has been suggested in [61] for tracking multiple targets as a general form of the NNKF algorithm. In this approach, each measurement is associated to its nearest track, and each track is updated with at most one associated measurement (the nearest one) using Kalman filtering.

When a single target is present, the optimal Bayesian approach for data association considers all possible association hypotheses from the target initiation time index to the current time and updates the target track using sum of the all possible track updates weighted by evaluated association probabilities [62]. Since the optimal Bayesian method is computationally intractable, suboptimal approaches have been suggested. The suboptimal N-scan-back algorithm [62] considers all association hypotheses from $N$ previous scans up to the current scan. The probabilistic data association filter (PDAF) introduced in [63], considers only the measurements of the current scan and can be considered as an N-scan-back algorithm with $N = 0$. In this approach, the target track is updated using the weighted sum of all possible track updates from each data association hypotheses of the current scan. The joint probabilistic data association filter (JPDAF) [64] is an extension to the PDAF algorithm for multiple target tracking. In this approach, all possible association hypotheses at the current time are evaluated for all targets, and the tracks are updated by a weighted sum of all possible track updates from the data association hypotheses.

Other data association algorithms implement either maximum a posteriori (MAP) or maximum likelihood (ML) approaches to find the best sequence of measurements that can be associated to each target. The Viterbi algorithm [65] and the expectation maximization (EM) algorithm [66] have been widely used for this purpose. Multi-hypothesis tracking (MHT) [67] is a MAP estimator in which at each scan, instead of combining the hypotheses like JPDAF or selecting the best hypothesis like nearest neighbor (NN), the algorithm makes a tree of possible association hypotheses and decides based on the future observations through a MAP estimator.

All of the data association approaches discussed above are combined with the Kalman filtering family of approaches to estimate the target track. Kalman filtering is optimum for systems with a linear state space model and additive Gaussian noise. In the case of nonlinearity, the Extended Kalman filter (EKF) and similar approaches are implemented with one of the above data association techniques. In severely nonlinear state space models or in the presence of non-Gaussian noise distribution functions, however, the EKF approach does not perform well. The unscented Kalman filter, particle filtering, Cubature Kalman filter, etc., discussed in Chapter 2, are some recent alternative approaches in the family of Kalman filtering.

Sequential Monte Carlo approaches [68] have been introduced in multiple target tracking problems for both track estimation and data association. In the approach presented in [68], Gibbs sampling is implemented to solve the data association problem, and particle filtering is used for target track estimation. The algorithm is developed for the cases of single target, multiple target, single receiver, and multiple receiver systems for both fixed and variable number of targets [68, 69].

Recently, the probability hypothesis density (PHD) filter [70] has been developed for multiple target tracking. The PHD filter uses a Bayesian approach and finite-set statistics (FISST) for joint estimation of the time-varying number of targets and the target states from noisy measurements. The PHD filter is a sub-optimum approach based on optimal multi-target Bayesian filter that implements random finite set (RFS) theory. While the optimal Bayesian approach is computationally intractable, the PHD approach remains computationally feasible by propagating only a first order multi-target moment, defined as the probability hypothesis density, in time. Implementation of the PHD filter using a sequential Monte Carlo approach is also suggested in [71]. The GM-PHD filter, a closed form solution to the general PHD filter for multi-target linear state space models with additive Gaussian noise, has been introduced in [72] and estimates jointly the target states and the number of targets. Although GM-PHD dramatically reduces the implementation complexity of the general PHD filter, it is optimum only for linear Gaussian system state space models.

In the following section, we propose a modification of the stack-based tree search algorithm, discussed in Chapter 2, for tracking a single target in clutter. This approach can be classified as a sub-optimum Bayesian approach for both track estimation and data association and can be implemented for linear/non-linear state space models as well as Gaussian/non-Gaussian additive noise.

## 5.2 Tracking a Single Target in Clutter

In this section, we develop a stack based tree search algorithm for tracking a single target in the presence of clutter. In Section 5.2.1 we first talk about the system model under which we consider the single target tracking problem. The proposed algorithm for target tracking via stack-based tree search including metric calculation, discretization, and data association, is described in Sections 5.2.2 - 5.2.5. Simulation models and results are presented in Section 5.2.6.

### 5.2.1 System Model

The goal of target tracking is to adaptively estimate target parameters of interest based on relevant noisy measurements. We use a conventional state-space model in which the relevant target parameters form the state space vector denoted by $\mathbf{x}$. Common parameters to be tracked include location in space, target speed, target type, etc. We assume that the target motion follows a first order Markov model given by

$$\mathbf{x}_k = \mathcal{F}(\mathbf{x}_{k-1}) + G_k \mathbf{v}_k, \tag{5.1}$$

where $\mathbf{v}_k$ denotes the target state transition noise vector with known probability distribution function $p_{\mathbf{v}}(v)$, and $\mathcal{F}(.)$ is a function that governs the deterministic state space transition in time. The (possibly nonlinear) function $\mathcal{F}(.)$ and matrix $G_k$ are assumed to be known. The index $k$ denotes the arrival of the $k$th set of contact data, or the $k$th scan. Scans are assumed to be separated by $\Delta_t$ time units, and hence the time at which the $k$th scan is

received is given by $k\Delta_t$.

The tracker is modeled as a single sensor with known location in the region of interest. At each time index $k$, the tracker may observe the target state through the observation model given by

$$\mathbf{z}_k = \mathcal{H}(\mathbf{x}_k) + \mathbf{w}_k, \tag{5.2}$$

where $\mathbf{z}_k$ denotes the target observation vector at scan index $k$, and $\mathbf{w}_k$ denotes the measurement noise vector with known probability distribution function $p_{\mathbf{w}}(w)$, which is assumed to be independent of the state transition noise $\mathbf{v}_k$. The function $\mathcal{H}(.)$, defines the transformation from the target state to the observed quantity.

At each scan, the observation matrix $\mathbf{Z}_k = [\mathbf{z}_k^1 \quad \mathbf{z}_k^2 \quad \cdots \quad \mathbf{z}_k^{m_k}]$ is formed, where $\mathbf{z}_k^j$, $j = 1, \ldots, m_k$, denotes the $j$th measurement (or contact) of the scan at time index $k$. The number of contacts in the $k$th scan is denoted by $m_k$. We assume that at most one contact is generated by the target and that the target contact is present in the observation matrix with probability of $P_D$. Hence, each set of contacts may include only one noisy observation of the target position, and all other contacts in the scan are caused by clutter. Depending upon the context in which the tracker is operating, sources of clutter may include, for example, trees, buildings, ocean waves, fish, and the sea floor. Contacts generated by clutter are assumed to be uniformly distributed in the surveillance region with volume of $V$. The number of clutter that observed in the surveillance region in the $k$th scan follows a Poisson distribution with expected value of $\zeta V$, where $\zeta$ is the average number of clutter per unit volume (clutter density). It is assumed that the initial state space estimate of the target $(\mathbf{x}_0)$ is available for the tracker.

We frame target tracking in clutter as a tree-search problem in which we aim to find the most likely path through the tree. The search tree is constructed such that each path represents a possible sequence of target states, and the stack algorithm navigates the tree to identify likely paths, or equivalently likely target tracks.

### 5.2.2 Stack-Based Tree Search Approach to Tracking

The stack-based tree search approach to tracking approximates the Bayesian inference tracking approach by approximating the posterior probability density function of a sequence of the target states:

$$P(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k | \mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_k). \tag{5.3}$$

The stack-based tracker navigates the tree of possible target states by extending only the most likely path (target state sequence) at each iteration. In this manner, the tracker explores only highly likely state sequences (or equivalently computes the posterior distribution only at likely state values) and hence provides reduced complexity relative to full Bayesian inference. The stack algorithm maintains a list, or stack, of paths through the tree that have been explored thus far. At each iteration of the algorithm, the path (target state sequence) with the largest metric (likelihood) is extended to all possible children, each representing a possible state at the next time index. The extended paths, along with their metrics, are then added to the stack.

As discussed further in Section 5.2.5, application of the stack-based tracker always requires discretization of the state space to replace the continuous posterior distribution on the state by a probability mass function (PMF). Discretization plays a significant role in the complexity and performance of tree search tracking since the dimension of the search tree significantly changes with changing the number of the discretized states. In Section 5.2.5, we propose a sampling based discretization technique to reduce the number of discrete states for tree search tracking.

### 5.2.3 Metric Calculation

In a tree search target tracking technique, possible target tracks are modeled by the paths of the search tree. A metric is associated to each path and provides a measure for comparison among the paths. To reduce the complexity, the stack algorithm avoids to explore all the

existing paths and explores the path with largest metric until the largest metric path is extended up to current scan index. Therefore, the existing paths within the search tree are extended up to different scan indexes. We denote the $l$th path of the search tree, extended up to scan index $k_l$, by $\chi_{k_l}^{(l)} = \{\mathbf{x}_1^{(l)}, \mathbf{x}_2^{(l)}, \cdots, \mathbf{x}_{k_l}^{(l)}\}$. The posterior probability density function of each target state given its previous state and the measured contacts is given by

$$
\begin{aligned}
p(\mathbf{x}_{k_l}^{(l)}|\mathbf{x}_{k_l-1}^{(l)}, \mathbf{Z}_{k_l}) &= \frac{p(\mathbf{Z}_{k_l}|\mathbf{x}_{k_l}^{(l)}, \mathbf{x}_{k_l-1}^{(l)})p(\mathbf{x}_{k_l}^{(l)}|\mathbf{x}_{k_l-1}^{(l)})}{p(\mathbf{Z}_{k_l})} \\
&\propto p(\mathbf{Z}_{k_l}|\mathbf{x}_{k_l}^{(l)})p(\mathbf{x}_{k_l}^{(l)}|\mathbf{x}_{k_l-1}^{(l)}).
\end{aligned}
\tag{5.4}
$$

Evaluation of the term $p(\mathbf{Z}_{k_l})$ requires using the total probability theorem that expands over all possible target states $\mathbf{x}_{k_l}$. In a surveillance region with high number of clutter contacts and few targets, the clutter generated contacts have more contribution in $p(\mathbf{Z}_{k_l})$ than the target generated contacts. Assuming that the clutter density is not changing by time index $k_l$, $p(\mathbf{Z}_{k_l})$ is assumed to be constant over different time indexes and is ignored in the metric calculation. The state transition probability $p(\mathbf{x}_{k_l}^{(l)}|\mathbf{x}_{k_l-1}^{(l)})$ is calculated from target motion model (5.1)

$$
p(\mathbf{x}_{k_l}^{(l)}|\mathbf{x}_{k_l-1}^{(l)}) = p_\mathbf{v}(G^+(\mathbf{x}_{k_l} - \mathcal{F}(\mathbf{x}_{k_l-1}))),
\tag{5.5}
$$

where the superscript $+$ denotes the Moore Penrose pseudo-inverse [73] operation. Calculation of the likelihood function $p(\mathbf{Z}_{k_l}|\mathbf{x}_{k_l})$ in (5.4) requires incorporating a data association technique, since the target generated contact is not identified from the clutter generated contacts in the observation matrix $\mathbf{Z}_{k_l}$. We discuss the proposed data association technique in Section 5.2.4.

The stack algorithm generates paths that are explored to different scan indexes, and the path metric must provide a fair comparison among these paths. While the original

stack algorithm incorporates a bias term in the path metric to allow for fair comparison across paths of differing lengths, the structure of the tracking problem does not facilitate computation of such a bias term. Hence, instead of calculating the path metric over the full path, we define the path metric using a time window of the state transition posterior probabilities in (5.4). If we assume that the statistics of the tracking problem such as probability density function of the clutter density, measurement noise, and number of targets are not changing with time, then the suggested metric can be considered as a fair metric since the same window length is used for all paths.

In a target tracking application, the estimation accuracy of the current target state is often of more interest than that of states have been visited in the past. Additionally, calculating the metric in the logarithmic domain converts the products into the summations and avoids working with very small values. Therefore, we consider the posterior probability density function of each estimated target state (5.4) in the logarithmic domain, and we implement an exponentially decaying window with parameter $\lambda$ to discount the weight of the states that are visited in the past. The path metric of the $l$th path is given by

$$b^{(l)} = \frac{1 - \lambda}{1 - \lambda^{k_l - 1}} \times \sum_{i=2}^{k_l} \lambda^{k_l - i} \log \left[ p(\mathbf{Z}_i | \mathbf{x}_i^{(l)}) p(\mathbf{x}_i^{(l)} | \mathbf{x}_{i-1}^{(l)}) \right], \tag{5.6}$$

where a normalizing factor $\frac{1-\lambda}{1-\lambda^{k_l - 1}}$ is incorporated to magnify the path metrics whose length is smaller than the memory imposed by the exponential windowing. Note that in (5.6), all the paths are generated from the same initial state $\mathbf{x}_1$ in the tree; hence, they have equal values of $p(\mathbf{x}_1)$, which can be deleted from the path metric.

### 5.2.4 Data Association

In general, the observation matrix includes contacts from target(s) and clutter, and hence calculation of the likelihood function $p(\mathbf{Z}_i | \mathbf{x}_i^{(l)})$ in (5.4) requires addressing a data association technique. Using an approach similar to the probabilistic data association filter [62], we

define the data association event $\theta_k^j$ such that at scan index $k$, the $j$th contact $\mathbf{z}_k^j$ is generated by the target and all other contacts are generated by clutter. The special case in which the target is not detected (e.g. all measurements are generated by clutter) is denoted by $\theta_k^0$. Using the data association hypotheses, the likelihood function is calculated from the total probability theorem

$$
\begin{aligned}
p(\mathbf{Z}_i|\mathbf{x}_i^{(l)}) &= \sum_{j=0}^{m_i} p(\mathbf{Z}_i|\mathbf{x}_i^{(l)}, \theta_i^j) p(\theta_i^j|\mathbf{x}_i^{(l)}) \\
&= \sum_{j=0}^{m_i} p(\theta_i^j|\mathbf{x}_i^{(l)}) \prod_{k=1}^{m_i} p(\mathbf{z}_i^k|\mathbf{x}_i^{(l)}, \theta_i^j),
\end{aligned}
\tag{5.7}
$$

where the second line is derived based on the assumption that clutter generated contacts are independent from target contacts and other clutter generated contacts. The probability distribution function of a target generated contact is calculated from the measurement model and is equal to the probability distribution function of the measurement noise. The clutter generated contacts are assumed to be uniformly distributed in the surveillance region with volume $V$. Therefore, the likelihood function for each contact $p(\mathbf{z}_i^k|\mathbf{x}_i^{(l)}, \theta_i^j)$ is given by

$$
p(\mathbf{z}_i^k|\mathbf{x}_i^{(l)}, \theta_i^j) = \begin{cases} p_{\mathbf{w}}(\mathbf{z}_i^j - \mathcal{H}(\mathbf{x}_i^{(l)})) & \text{for } k = j \text{ (a target generated contact)} \\ \frac{1}{V} & \text{for } k \neq j \text{ (clutter contact)} \end{cases}.
\tag{5.8}
$$

Substituting (5.8) into (5.7), we have

$$
p(\mathbf{Z}_i|\mathbf{x}_i^{(l)}) = \frac{1}{V^{m_i}} p(\theta_i^0|\mathbf{x}_i^{(l)}) + \sum_{j=1}^{m_i} \frac{1}{V^{m_i-1}} p_{\mathbf{w}}(\mathbf{z}_i^j - \mathcal{H}(\mathbf{x}_i^{(l)})) p(\theta_i^j|\mathbf{x}_i^{(l)}).
\tag{5.9}
$$

In modeling of the tracking problem, the number of clutter generated contacts in the surveillance region with volume $V$ is a Poisson distributed random variable with probability density

function $\mu_F(.)$ and parameter $\zeta V$, where $\zeta$ is the average number of clutter contacts per unit volume. Therefore, the prior probability of each data association event $\theta_i^j$ is given by

$$
p(\theta_i^j | \mathbf{x}_i^{(l)}) = \begin{cases} P_D \mu_F(m_i - 1) & \text{for } j \neq 0 \\ (1 - P_D)\mu(m_i) & \text{for } j = 0 \end{cases}
$$

$$
= \begin{cases} P_D \frac{e^{-\zeta V}(\zeta V)^{m_i - 1}}{(m_i - 1)!} & \text{for } j \neq 0 \\ (1 - P_D)\frac{e^{-\zeta V}(\zeta V)^{m_i}}{(m_i)!} & \text{for } j = 0 \end{cases} . \tag{5.10}
$$

In sonar applications, a contact is detected at the receiver when the amplitude of the received signal is greater than a predefined threshold level. In many real applications, the amplitude of a target-generated contact has a different statistical distribution than that of a clutter-generated contact. We denote the probability distribution function of the amplitude of a clutter-generated contact by $p_0(a)$ and that of a target-generated contact by $p_1(a)$. Using the amplitude information, the likelihood function of each contact (5.8) is updated as follows:

$$
p(\mathbf{z}_i^k | \mathbf{x}_i^{(l)}, \theta_i^j) = \begin{cases} p_{\mathbf{w}}(\mathbf{z}_i^j - \mathcal{H}(\mathbf{x}_i^{(l)}))p_1(a) & \text{for } k = j \\ \frac{1}{V}p_0(a) & \text{for } k \neq j \end{cases} . \tag{5.11}
$$

Substituting (5.11) into (5.7), the likelihood function is given by

$$
p(\mathbf{Z}_i | \mathbf{x}_i^{(l)}) = \prod_{k=1}^{m_i} p_0(a_i^k) \times \left[ \frac{1}{V^{m_i}} p(\beta_i^0 | \mathbf{x}_i^{(l)}) \right. \tag{5.12}
$$

$$
\left. + \sum_{j=1}^{m_i} \frac{1}{V^{m_i - 1}} \frac{p_1(a_i^j)}{p_0(a_i^j)} p_w(\mathbf{z}_i^j - \mathcal{H}(\mathbf{x}_i^{(l)}))p(\eta_i^j | \mathbf{x}_i^{(l)}) \right] .
$$

Using the amplitude information in the path metric calculations improves the performance

if the distribution functions $p_0(a)$, and $p_1(a)$ are separated from each other. In this case, the contact amplitude provides information about the class in which the contact falls. If this condition is met, the likelihood of a target generated contact is magnified. This increases the path metric of the true target path and decreases the possibility of exploring non-target paths in the search tree.

## 5.2.5 Dynamic Discretization Technique

In general, the estimate of the state vector from the observation sequence using the dynamic state space model (5.43) may take any value in the system state space. The Kalman filtering family of approaches estimates the state vector $\mathbf{x}$ on the continuous state space. Techniques such as particle filtering and the unscented Kalman filtering use a discrete set of particles to approximate the continuous density of the state space. Other techniques, such as point mass approaches, discretize the state space and estimate the state vector $\mathbf{x}$ from among a discrete set of points. The stack-based tree search algorithm requires the state space to be discretized to a finite set of possible states that can be mapped onto the branches of the search tree. In digital synthesis of non-linear filters [17] and point-mass techniques [18], several discretization techniques have been employed. All of these techniques discretize the whole state space and evaluate the posterior probability density function over the entire space. The stack algorithm, however, partially explores the state space and in each iteration, only needs to discretize the observable area, e.g. the area (within the state space) to which the target could move from its current state with sufficiently high probability. This is the key advantage of the stack algorithm in reducing computational complexity relative to point mass techniques.

One of the elements to be considered in discretization is how fine/coarse the quantization should be. Increasing the number of discrete points reduces the discretization error of the algorithm by providing finer discretization grids of the system state space but increases the computational complexity of the stack algorithm. The stack algorithm discretizes the observable area from the currently estimated target state and the number of discretization

levels defines the number of children at each node in the search tree. In this section, we propose a dynamic discretization technique to discretize the observable region in the vicinity of the target estimated path. We first divide the observable area into cells and evaluate the boundaries of each cell. Then, we propose a sampling algorithm for selecting discrete points that represent discretized cells. The proposed algorithm reduces the discretization error without increasing the number of discrete levels in the observable area.

**Generating Discrete Cells**

In the proposed tree search algorithm for tracking, instead of discretizing the entire surveillance area, only the observable area from each node of the tree is discretized. Evolution of the target states from each estimated target location is governed by the target motion model (5.1), hence it defines the observable area for each node of the search tree. Using the target motion model, the observable area for each node of the search tree is centered on the predicted expected value of the target state and its dimension is defined by the predicted error covariance of the target state estimation. To derive a close form for the observable area, we assume that the target motion model is linear and the state process noise is additive and Gaussian distributed. Using this assumption does not limit the applications of the proposed algorithm to linear/Gaussian models, since we only use this assumption to find the dimension of the state space that must be discretized. The observable area of each node of the search tree for a nonlinear/non-Gaussian state transition model may be evaluated using linear-Gaussian approximations of the target motion model.

Assume that the previously estimated target state $\hat{\mathbf{x}}_{i-1}$ is related to the true target state $\mathbf{x}_{i-1}$ by

$$\hat{\mathbf{x}}_{i-1} = \mathbf{x}_{i-1} + \tilde{\mathbf{x}}_{i-1}, \tag{5.13}$$

where $\tilde{\mathbf{x}}_{i-1}$ is the estimation error at time index $i - 1$. Using this model within the DSSM,

the prediction of the state estimation error is given by

$$
\begin{aligned}
\tilde{\mathbf{x}}_{i|i-1} &= \hat{\mathbf{x}}_{i|i-1} - \mathbf{x}_i & (5.14)\\
&= \mathcal{F}(\hat{\mathbf{x}}_{i-1}) - \mathcal{F}(\mathbf{x}_{i-1}) - G\mathbf{v}_{i-1}\\
&= \mathcal{F}(\mathbf{x}_{i-1} + \tilde{\mathbf{x}}_{i-1}) - \mathcal{F}(\mathbf{x}_{i-1}) - G\mathbf{v}_{i-1}.
\end{aligned}
$$

The general form of the probability distribution function of the random vector $\tilde{\mathbf{x}}_{i|i-1}$ depends on the state transition function $\mathcal{F}(.)$ and the probability distribution function of the state transition noise $\mathbf{v}_{i-1}$. For a linear Gaussian state transition model, the probability distribution function of the random vector $\tilde{\mathbf{x}}_{i|i-1}$ is Gaussian. For a nonlinear/non-Gaussian target motion model, its distribution depends on the distribution of the state process noise vector and may not have a closed form. In this case, one can implement statistical techniques [74] to approximate the estimated target state probability distribution function. To derive a closed form expression in this work, we assume that the state process noise is Gaussian with zero mean and covariance matrix $\Sigma_v$. We also assume that the state transition function is linear and is represented by $F$. For a nonlinear state transition, $F$ represents a linearized version of $\mathcal{F}(.)$ using its Taylor series expansion around the estimated target state. Again this assumption is used only to evaluate the boundaries of the discretization area and does not limit the applications of the stack algorithm for general non-linear non-Gaussian scenarios. Using this assumption for the system state space model, the predicted estimation error $\tilde{\mathbf{x}}_{i|i-1}$ is Gaussian with zero mean and covariance matrix

$$
\hat{\Sigma}_{i|i-1} = F\hat{\Sigma}_{i-1|i-1}F^T + G\Sigma_v G^T, \tag{5.15}
$$

where $\Sigma_{i-1|i-1}$ is the estimation error covariance matrix at time index $i-1$ given the contacts up to time $i-1$. Unlike Kalman filtering techniques that estimate the expected value and the error covariance of the estimation, the structure of the stack algorithm does not directly provide an estimate of the error covariance $\Sigma_{i-1|i-1}$. For this purpose, we

calculate the sample error covariance of the target state in the search tree. Consider all discrete target states at time index $i-1$ that are generated from the same origin $\hat{\mathbf{x}}_{i-2}$ in the search tree. We denote these states by $\hat{\mathbf{x}}_{i-1}^j$ for $j = 1, \cdots, S$ where $S$ is the number of possible target states at time $i-1$ generated from the same origin (or, equivalently, the number of discretization levels). The sample error covariance can be approximated using

$$\hat{\Sigma}_{i-1|i-1} = \sum_{j=1}^{S} \mathbf{P}_j (\hat{\mathbf{x}}_{i-1}^j - \bar{\mathbf{x}}_{i-1})(\hat{\mathbf{x}}_{i-1}^j - \bar{\mathbf{x}}_{i-1})^T, \tag{5.16}$$

where $\bar{\mathbf{x}}_{i-1}$ is the sample expected value given by

$$\bar{\mathbf{x}}_{i-1} = \sum_{j=1}^{S} \mathbf{P}_j \, \hat{\mathbf{x}}_{i-1}^j, \tag{5.17}$$

and $\mathbf{P}_j$ is the conditional probability of the $j$th state,

$$\begin{aligned} \mathbf{P}_j &= p(\hat{\mathbf{x}}_{i-1}^j | \mathbf{Z}_{i-1}, \hat{\mathbf{x}}_{i-2}) \tag{5.18}\\ &\propto p(\mathbf{Z}_{i-1} | \hat{\mathbf{x}}_{i-1}^j) p(\hat{\mathbf{x}}_{i-1}^j | \hat{\mathbf{x}}_{i-2}). \end{aligned}$$

The right hand side of (5.18) is computed in the metric calculation (5.6), and $P_j$ can be calculated by normalizing (5.18) to meet $\sum_{j=1}^{S} \mathbf{P}_j = 1$.

The approximated error covariance matrix in (5.15) is used for calculating the boundaries of the observable area and the discretized cells. The required state space discretization can be achieved by discretizing the elements of the state vector in the observable region, but discretization in this way may be inefficient due to possible correlation between the elements of the system state. In sonar target tracking, the target state vector typically contains target location and target speed information, where the target speed can be exactly known based on the previous target location, current target location, and time difference between

the scans. To avoid correlated discretized states, the principal components of the state vector are discretized. The principal components can be generated from the approximated covariance matrix $\hat{\Sigma}_{i|i-1}$ in (5.15). Let $\mu_i^1, \cdots, \mu_i^d$ and $\mathbf{e}_i^1, \cdots, \mathbf{e}_i^d$ be the eigenvalues and eigenvectors, respectively, of $\hat{\Sigma}_{i|i-1}$. We define $\nu_i^j$ as the projection of $\tilde{\mathbf{x}}_{i|i-1}$ onto its $j$th eigenvector $\mathbf{e}_i^j$. Assuming Gaussian distribution for $\tilde{\mathbf{x}}_{i|i-1}$, $\nu_i^j$ is a Gaussian random variable with zero mean and variance $\mu_i^j$. The random variables $\nu_i^j$ for $j = 1, \cdots, d$ are uncorrelated, because they are the projection of the state error matrix onto the orthogonal eigenvector basis. Therefore, the state space can be discretized by individually discretizing each random variable $\nu_i^j$.

We use the technique proposed by Max [75] for discretizing the uncorrelated individual random variables $\nu_i^j$. In this technique, the boundaries of the discretized cells $(u_l, u_{l+1})$, and the discrete points $\varepsilon_l$ are calculated to minimize mean square error, defined by

$$\sum_{l=1}^{L} \int_{u_l}^{u_{l+1}} (v^j - \varepsilon_l)^2 p_{\nu^j}(\nu^j) d\nu, \tag{5.19}$$

where $p_{\nu^j}(\nu^j)$ is the probability distribution function of $\nu^j$, and $L$ denotes the number of discretized regions. The cell boundaries and the discrete point values for a Gaussian distribution are provided in [75].

**Selecting Points in Each Cell**

Discretizing the state space introduces discretization error in the tree-search based tracker. In order to reduce this error, one can employ finer discretization, but this has a substantial negative impact on the speed of the tracker, since the size of the tree increases significantly. As an alternative, we maintain a small number of discrete cells and employ sampling to obtain a more accurate representation of the likelihood in each discrete cell.

The approach provided in (5.19) for calculating the discretized cell boundaries and their

discrete points does not necessarily guarantee to generate the metric peak in the discretized cell. We suggest a sampling discretization approach using the same cell boundaries as derived from (5.19) and the discrete points that are derived from sampling inside each cell. Using this technique, the discrete points can be selected in a finer grids within the cell that generates highest metric values. Hence, the target state space is discretized in a data-informed manner to reduce the discretization error of the target state space estimate.

Figure 5.1 represents the sampling discretization and compares it with the fixed discretization technique. As shown in this figure, rather than selecting a fixed state point to represent a discrete cell, the cell is sampled at a finer scale, and the most likely sample is selected to represent the cell. Using the cell boundaries $(u_l, u_{l+1})$ as in (5.19), the probability mass function in finer grades $(\varepsilon_l \in (u_l, u_{l+1}))$ is evaluated as follows:

$$\varepsilon_l \quad = \quad \arg\max_{\nu^j} p(\nu^j | \mathbf{Z}_i, \nu^j \in (u_l, u_{l+1}), \mathbf{x}_{i-1}), \tag{5.20}$$

and

$$p(\nu^j | \mathbf{Z}_i, \nu^j \in (u_l, u_{l+1}), \mathbf{x}_{i-1}) \propto p(\mathbf{Z}_i | \nu^j) p(\nu^j | \nu^j \in (u_l, u_{l+1}), \mathbf{x}_{i-1}). \tag{5.21}$$

The likelihood function $p(\mathbf{Z}_i | \nu^j)$ must be expanded over data association hypotheses as in (5.7). Considering that $p(\nu^j | \mathbf{x}_{i-1})$ is the probability distribution function of the process noise, the conditional probability density $p(\nu^j | \nu^j \in (u_l, u_{l+1}), \mathbf{x}_{i-1})$ is proportional to the truncated version of $p(\nu^j | \mathbf{x}_{i-1})$ in the intervals $(u_l, u_{l+1})$.

In finding the discrete point within the interval $(u_l, u_{l+1})$ that maximizes the posterior probability in (5.20), the posterior probability density is calculated by dividing the interval $(u_l, u_{l+1})$ into small intervals and selecting the center of each smaller interval as a candidate for the point mass. The posterior probability distribution is calculated for each candidate, and the one that maximizes the posterior probability is selected as a discrete point of the interval $(u_l, u_{l+1})$.
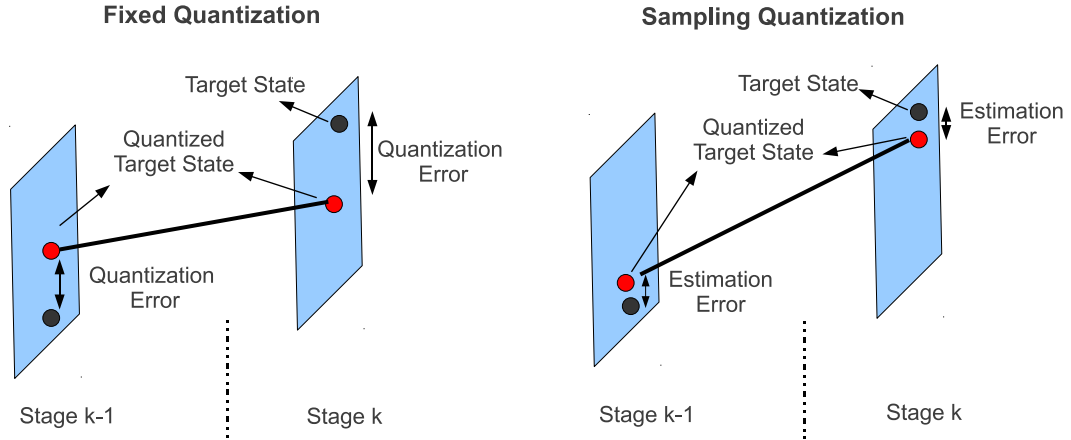
Figure 5.1: State space discretization using fixed discretization (left figure) and sampling discretization (right figure). In fixed discretization, the estimated target location in each region is deterministic; in sampling discretization, the estimated target location is sampled from the posterior distribution function of the target in that region.

Using the proposed sampling technique to choose discrete points in each cell allows for the use of fewer discretization levels without an increase in discretization error. Since decreasing the number of discrete levels generally reduces the number of the paths that are explored in the search tree, computational complexity is reduced, as well. While sampling discretization increases the complexity of the metric calculation for each path, the reduction in the number of paths explored outweighs the effects of additional metric calculations.

### 5.2.6 Simulations

To evaluate the performance of the proposed stack-based single target tracking algorithm, we perform simulations for two target scenarios. In the first scenario, the target follows a linear motion model, and in the second, the target follows a nonlinear motion model that reflects the presence of obstacles. For both scenarios, the target state vector is defined by $\mathbf{x} = [x \; y \; \dot{x} \; \dot{y}]^T$, where $(x, y)$ denotes the position of the target in Cartesian coordinates, and $(\dot{x}, \dot{y})$ denotes the target velocity in Cartesian coordinates.

The same observation model is employed in both scenarios. A single sensor is placed at

$(0,0)$ in Cartesian coordinates. The target measurement model is given by (5.2), where

$$h(\mathbf{x}) = \begin{bmatrix} \tan^{-1}(y/x) \\ \sqrt{x^2 + y^2} \end{bmatrix}. \tag{5.22}$$

Note that the target observation vector contains noisy measurements of the bearing and range of the target, which are nonlinear transformations of the Cartesian target position coordinates.

For evaluation purposes, the performance of the stack-based tracking algorithm is compared to that of the EKF and particle filtering (PF). The EKF technique employs the nonlinear state space model in calculating the predicted state estimate and the innovation error, but it linearizes the nonlinear functions based upon the first derivative of the nonlinear functions in calculating the Kalman gain and the updated covariance error of the estimation [76]. To reflect the fact that the initial target state is assumed to be known (e.g. $\hat{\mathbf{x}}_0 = \mathbf{x}_0$), the initial covariance matrix of the EKF algorithm is given by $P_0 = 5 * 10^{-5} I_4$, where $I_4$ denotes the $4 \times 4$ identity matrix. The probabilistic data association (PDA) filter is employed in conjunction with the EKF (known as EKF-PDA) to combat the effects of clutter [62]. EKF-PDA gates the measured contacts at each scan using a g-sigma ellipsoid. A g-sigma ellipsoid is an ellipse that encompasses a random vector with high probability. This ellipse is centered at the expected value of the random vector and its diagonals are equal to the $g$ times of the standard deviations in each direction. For a two dimensional random variable, the gate volume is $\pi g^2 |S|^{1/2}$, where $S$ is the summation of the noise covariance matrix and the covariance error matrix of the target state estimate in Cartesian coordinates. Selecting $g^2 = 16$ is sufficient to achieve a probability of close to 0.99 that the target, if detected, is inside the gate [77].

For simulating the particle filtering technique, we use the standard particle filtering approach as is proposed in [68]. To make a fair comparison among the algorithms, we use

the same importance sampling distribution function and likelihood function as is used in the stack algorithm. However, any other type of the importance sampling distribution function and the likelihood function in a PF algorithm can be applied in the stack algorithm for state space discretization and metric calculation, as well. The number of particles is set to 500. For scan $k$, the $l$th particle, denoted by $\mathbf{x}_k^{(l)}$, is updated using the target motion model in (5.1) through sampling from the following importance sampling distribution function:

$$q_k(\mathbf{x}^{(l)}) = \mathcal{N}(\mathbf{x}_k^{(l)}; \mathcal{F}(\hat{\mathbf{x}}_{k-1}^{(l)}), G^+\hat{\Sigma}_{k-1}G^{+T} + \Sigma_v), \tag{5.23}$$

where the notation $\mathcal{N}(x; a, b)$ represents that the random vector $x$ is Gaussian distributed with expected value $a$ and covariance matrix $b$. The term $\hat{\mathbf{x}}_{k-1}$ denotes the previous estimate of the target state vector, $\hat{\Sigma}_{k-1}$ denotes the estimated covariance matrix of the target, and $G^+$ is the pseudo-inverse of the transition matrix $G$. The likelihood function of the observations for the particle filtering technique is calculated using the same approach as in the stack algorithm (5.12) to make a fair comparison between the two algorithms. The re-sampling step is performed when the number of effective particles drops below one third of the total number of particles.

The number of clutter contacts generated at each scan is drawn from a Poisson distribution with parameter $\zeta V$. Due to the non-linearity of the target measurement model, the observation gate dimension depends on the error covariance of the measurement noise and the distance between the target and the source. Therefore, we define the clutter density as the average number of clutter contacts that appear in a unit volume, and we denote it by $\zeta$.

In our simulations, we leave the time and distance parameters unitless. Realistic units are dependent upon the application. The results presented here apply for any choice of time and distance units as long as these choices are carried into the definition of state transition noise variance, observation noise variance, etc. The maximum stack size used for simulation is limited to $L = 512$ entries.

**Linear Motion Model**

In the the first simulation scenario, the target moves according to a linear motion model given by

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + G\mathbf{v}_k, \tag{5.24}$$

where the covariance matrix of $\mathbf{v}_k$ is given by $\Sigma_v = 5 * 10^{-4} I_2$. Matrices $F$ and $G$ are given by

$$F = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G = \begin{bmatrix} \frac{\Delta_t^2}{2} & 0 \\ 0 & \frac{\Delta_t^2}{2} \\ \Delta_t & 0 \\ 0 & \Delta_t \end{bmatrix},$$

where $\Delta_t = 1$ is the time between scans. In this model, the state transition noise vector $\mathbf{v}_k$ determines the acceleration (in the $x$ and $y$ directions) over the time interval from $k\Delta_t$ to $(k+1)\Delta_t$. Each target path begins at point $(20, 20)$ in Cartesian coordinates. The initial velocity is drawn randomly from a zero-mean Gaussian vector with covariance matrix $0.02I_2$.

The observation noise $\mathbf{w}_k$ is assumed to be Gaussian with zero mean and covariance matrix of

$$\Sigma_w = \begin{bmatrix} \left(\frac{0.15\pi}{180}\right)^2 & 0 \\ 0 & 0.25^2 \end{bmatrix}, \tag{5.25}$$

where the first element of the diagonal is the variance of the bearing measurements and the second element is the variance of the range measurement.

A sample target path drawn from the motion model described above is shown in Figure 5.2. The observations (both from the target and from clutter) are plotted, as are the target position estimates generated by the stack-based tracker, the EKF-PDA, and the particle filtering (PF) techniques. An average of $\zeta = 1.0$ clutter contact is present in each
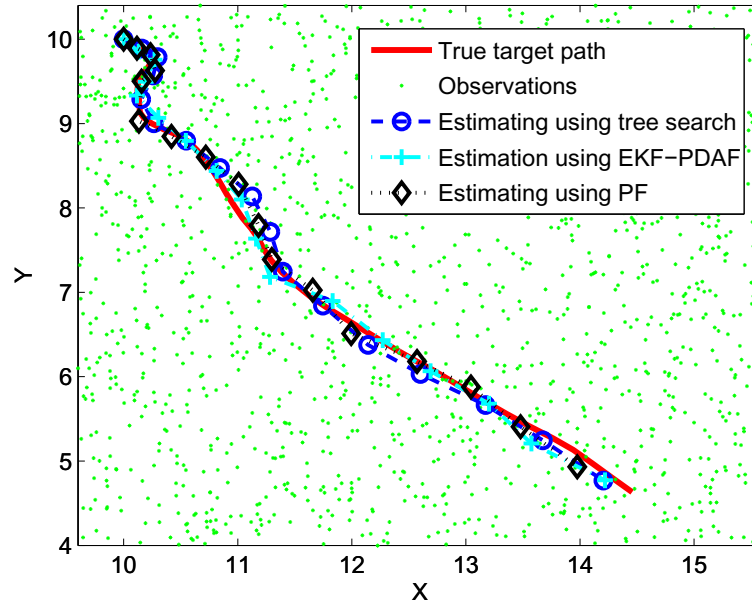
Figure 5.2: A sample target path with contacts (from target and clutter), stack-based tracker estimates, EKF-PDA estimates, and particle filtering estimates for the linear motion model with noisy observations of target bearing and range, $\zeta = 1$.

observation gate. We assume that the true target is detected in each scan with probability $P_D = 0.9$. In this sample realization, the stack-based tracker, the EKF-PDA, and the PF tracker successfully track the target through all 50 scans.

To evaluate the performance of the trackers, we have considered two criterion for each tracker: percentage of lost targets and the mean square error (MSE) of the target location estimate. To evaluate the percentage of lost targets, we must define a condition for determining that a target has been lost. A target is marked as lost if the estimated target location lies outside of a validation ellipse in any single scan. The validation ellipse is defined as an ellipse centered on the true target location with diagonals equal to the 4 times of the observation noise standard deviations. An example of the target path with validation ellipses is plotted in Figure 5.3.

All three techniques have been simulated over 50 scans for 200 independent realizations following the motion and the observation models given above. Generating independent
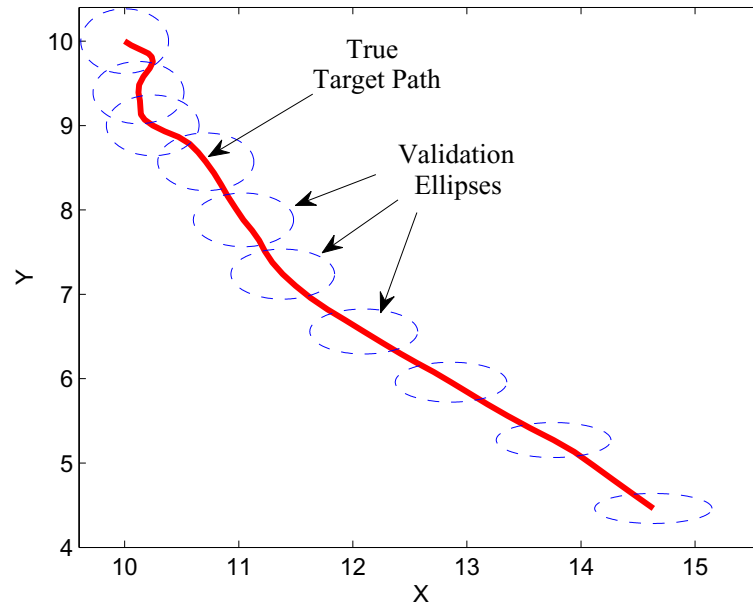
Figure 5.3: A realization of the target path and the estimation validation ellipses. The target tracking algorithm is considered to have lost a target if the estimated target location lies outside of the validation ellipse in any single scan.

realizations, the average number of lost targets is counted for each technique under different clutter density scenarios. Figure 5.4 shows the percentage of lost targets for different clutter density scenarios. While all three techniques could track the target for clutter density below 2, we see that the performance of the PF algorithm degrades faster than that of the stack algorithm and the EKF. In this case, we do not expect the stack-based tracker to outperforms the EKF, since the target motion model is linear and the nonlinear observation model is well approximated by its Taylor series expansion. When clutter density increases, the particles in particle filtering technique degenerate faster and the algorithm frequently needs to perform the re-sampling step. Therefore, the percentage of lost targets for particle filtering algorithm increases faster in early stages of increasing the clutter density.

Figure 5.5 shows the mean square error of the estimated target location averaged over 200 independent realizations of the target trajectory. The clutter density is fixed to $\zeta = 1.5$. For calculating the MSE, only the successfully estimated trajectories (not those associated
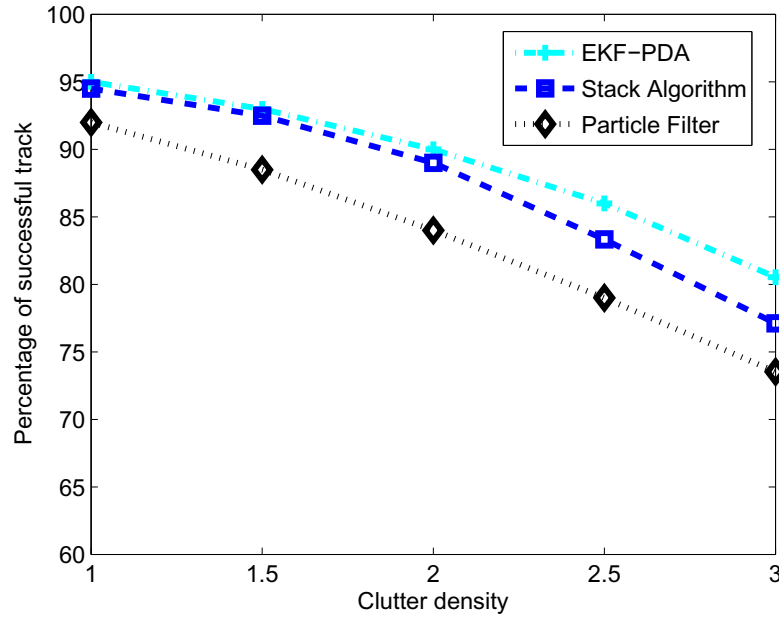
Figure 5.4: Performance comparison based on percentage of successful tracks versus clutter density.

with lost targets) are selected for averaging. We can see that the stack algorithm can estimate the track as precisely as the EKF technique. The low mean square error achieved by the stack algorithm is a result of the proposed sampling quantization technique, which significantly reduces the discretization error. In particle filtering, the estimated target location is provided by calculating the expected value of the particles. In high clutter environments, some particles may represent the false tracks and averaging over the particles increases the MSE of target location.

**Nonlinear Motion Model**

We also evaluate the performance of the stack-based tracker, PF, and the EKF-PDA in tracking a maneuvering target whose motion model is nonlinear. The motion model used in this evaluation is designed to simulate a maneuvering ship that travels in an area bounded by shoreline [78]. The target moves in a circular region of radius $r = 1$. The coastline is modeled by the perimeter of the circle, and a negative force moves the target back toward the
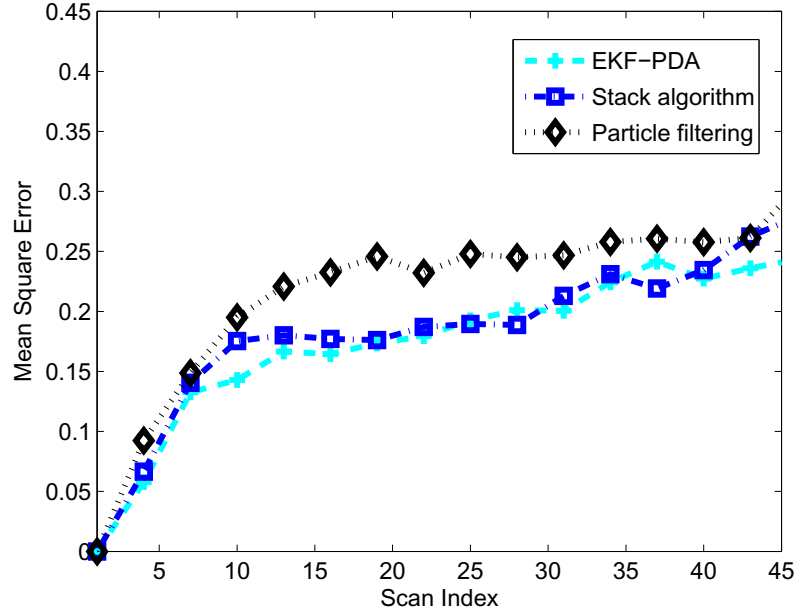
Figure 5.5: Performance comparison based on Mean Square Error of the estimated target location. The clutter density is fixed to $\zeta = 1.5$

center of the circle when it moves beyond the circle's perimeter. The target motion model is given by (5.1), where the covariance matrix of the state transition noise is $\Sigma_v = 10^{-2}I_2$, and

$$G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.26}$$

The nonlinear function $f(\cdot)$ is given by

$$f(\mathbf{x}) = \begin{bmatrix} x + \Delta_t \dot{x} \\ y + \Delta_t \dot{y} \\ \dot{x} + f_1(x,y) \\ \dot{y} + f_2(x,y) \end{bmatrix}, \tag{5.27}$$

where

$$
f_1(x, y) = \begin{cases} \frac{-10x}{\sqrt{x^2+y^2}} & \text{when } \sqrt{x^2 + y^2} > r \ \& \ x\dot{x} + y\dot{y} \geq 0 \\ 0 & \text{otherwise} \end{cases} , \tag{5.28}
$$

and

$$
f_2(x, y) = \begin{cases} \frac{-10y}{\sqrt{x^2+y^2}} & \text{when } \sqrt{x^2 + y^2} > r \ \& \ x\dot{x} + y\dot{y} \geq 0 \\ 0 & \text{otherwise} \end{cases} . \tag{5.29}
$$

The nonlinear functions governing velocity changes serve the purpose of keeping the target within the circle of radius $r$. When the target is within the circle, it accelerates in each direction of the Cartesian coordinate by an independent zero-mean Gaussian noise. When the target lies outside the perimeter of the circle and is moving away from the center of the circle, the target velocity is modified to direct the target back into the circle. Instances in which such nonlinear modifications are made pose challenges to the EKF-PDA, as will be seen in simulation results.

For simulation, a single sensor is present at $(0,0)$ (the center of the circle), and the sensor observes noisy measurements of bearing and range as described by (5.22). The time interval between scans is given by $\Delta_t = 0.15$, and the measurement noise covariance matrix is given by

$$
\Sigma_w = \begin{bmatrix} \left(\frac{\pi}{180}\right)^2 & 0 \\ 0 & 0.01^2 \end{bmatrix} . \tag{5.30}
$$

The initial position for each simulated target path is drawn from a uniform distribution within the circle, and the initial velocity for each axis ($x$ and $y$) is drawn from a zero-mean Gaussian distribution with variance 0.5.

One sample path drawn from the nonlinear motion model, along with the associated

Figure 5.6: A sample target path, stack-based tracker estimates, PF, and EKF-PDA estimates for the nonlinear motion model with noisy observation of target bearing and range, $\zeta = 1$. All techniques maintain track through 50 scans.

target observations and the stack-based tracker, PF, and EKF-PDA estimates, are shown in Figures 5.6 and 5.7. For this sample path, two different runs of the received target observations and clutter observations are simulated. For both plots, the clutter density is $\zeta = 1$, the probability of target detection is $P_D = 0.9$, and the stack algorithm is using exponential window with parameter $\lambda = 0.9$. In order to make the plots more readable, observations generated by clutter are not included. In Figure 5.6, all three algorithms are able to follow the target through all 50 scans. In Figure 5.7, the EKF-PDA loses the target track at the second change in direction (e.g. the first time the target travels outside the perimeter of the circle) and never regains an accurate estimate of the target location. This example highlights the advantage of the stack-based and PF tracker over a Kalman filter-based approach in the presence of severe nonlinearities.

In Figure 5.8, the performance of the stack algorithm is compared with that of PF
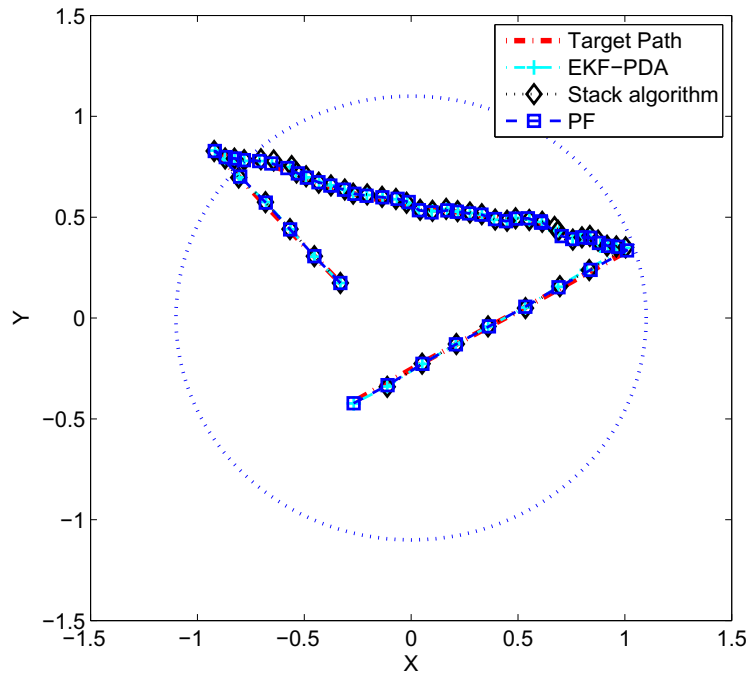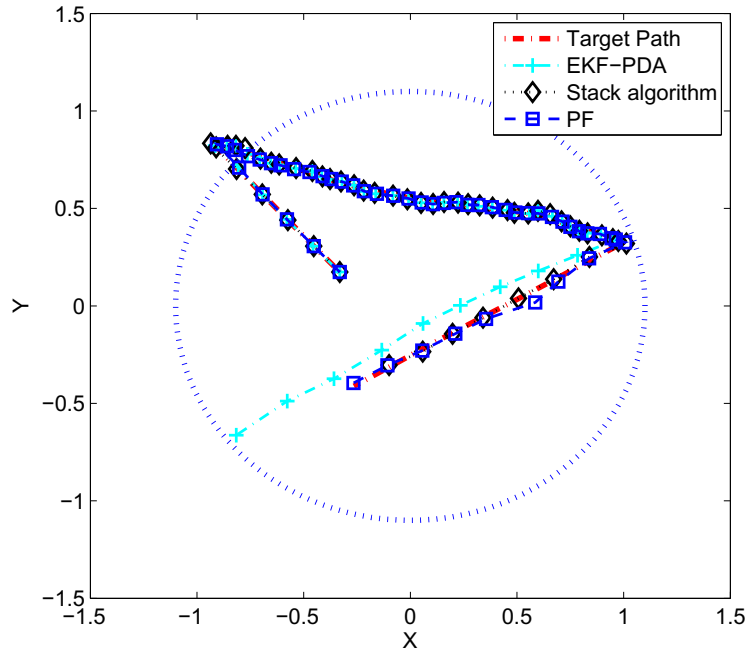
Figure 5.7: A sample target path, stack-based tracker estimates, PF, and EKF-PDA estimates for the nonlinear motion model with noisy observation of target bearing and range, $\zeta = 1$. The stack-based and the PF tracker maintain track through 50 scans, but the EKF-PDA is unable to follow the target's nonlinear motion in clutter and diverges in the second turn.

and EKF-PDA for different values of clutter density $\zeta$. In each scan, it is assumed that the target is measured with probability $P_D = 0.9$. Again, the sampling stack algorithm performs better than PF but it dramatically outperforms the EKF-PDA in the presence of clutter.

Figure 5.9 shows the MSE performance of the stack-based tracker, PF, and the EKF-PDA over 50 scans for 250 independent realizations of the nonlinear motion model with noisy observations of target bearing and range. The average MSE in target position estimation is evaluated for clutter density of $\zeta = 1$. At each scan, the true target is detected with probability $P_D = 0.9$. Only the mean square error of successful tracks (no lost targets) has been considered for averaging. The stack algorithm in this scenario works as well as the EKF-PDA, and it out performs the PF in estimation accuracy. This is because the stack
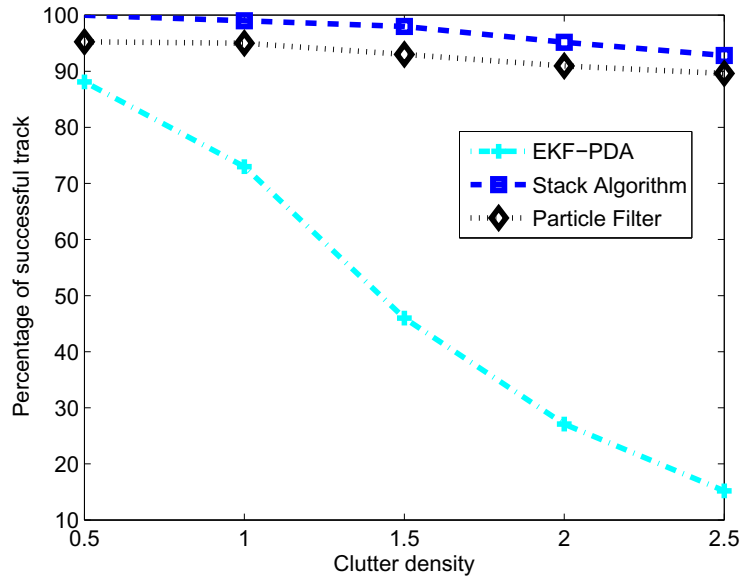
Figure 5.8: The performance of the stack algorithm is compared with that of EKF-PDA and PF. for a nonlinear target motion model. $\zeta$ values of 0, 1, and 2 are considered, and the true target appears in each scan with probability $P_D = 0.9$.

algorithm implements the sampling discretization technique which reduces the discretization error. In PF, however, the MSE is higher since the target state is estimated by averaging over all the particles which some may not follow the true target state.

## 5.3 Track Validation in the Stack-Based Tracker

In our development of the stack algorithm, we have assumed that the initial locations of the targets are estimated by an external track initiation algorithm. In addition to identifying targets, the track initiation algorithm may also initiate a considerable number of false tracks, particularly when clutter density is high. Similarly, when targets maneuver and/or probability of true target contact detection is low, the tracker may lose the target and instead follow clutter. Track validation is required to identify these cases in which a path followed by the tracker is not associated with a target.
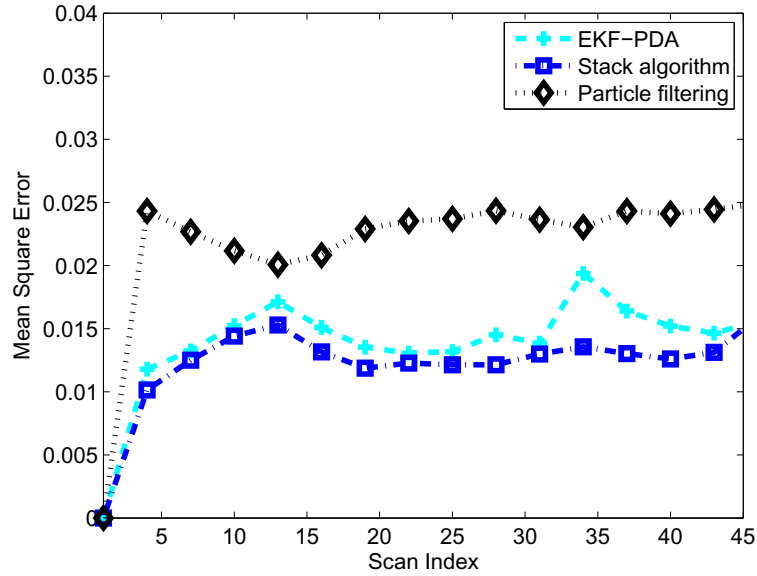
Figure 5.9: The MSE of the target location estimate for EKF-PDA, stack algorithm, and particle filtering. The MSE of the stack algorithm is close to that of the EKF and is significantly lower than the PF.

The Neyman-Pearson test can be used to determine track validity [79]. Define hypotheses $H_0$ and $H_1$ as

$$\begin{cases} H_0: & \text{target is absent} \\ H_1: & \text{target is present} \end{cases}. \tag{5.31}$$

The likelihood ratio of the contact matrix $\mathbf{Z}_i$ given the estimated target state $\hat{\mathbf{x}}_i$ is evaluated as

$$\gamma_i = \frac{p(\mathbf{Z}_i|\hat{\mathbf{x}}_i, H_1)}{p(\mathbf{Z}_i|\hat{\mathbf{x}}_i, H_0)}, \tag{5.32}$$

where the numerator $p(\mathbf{Z}_i|\hat{\mathbf{x}}_i, H_1)$ is calculated in (5.7). The denominator $p(\mathbf{Z}_i|\hat{\mathbf{x}}_i, H_0)$ is the likelihood that all contacts are generated by clutter and is equal to $(1-P_D)\frac{1}{V^{mo}}$. Therefore,

the likelihood ratio of a contact matrix $\mathbf{Z}_i$ that contains $m_i$ contact vectors $\mathbf{z}_i$ is equal to

$$\gamma_i = 1 + \frac{P_D}{(1 - P_D)\zeta} \sum_{j=1}^{m_k} p(\mathbf{z}_k^j | \mathbf{x}_k^{(l)}). \tag{5.33}$$

In conventional maximum likelihood target tracking techniques [79], the logarithm of likelihood ratio (LLR) is evaluated over a fixed-length window of scans, and an optimization technique finds the state vector which maximizes the LLR function. In this work, we employ the LLR function for validating the target paths within the tree. Under the assumption that recent contacts are more important than older contacts in evaluating the validity of a track, we apply an exponentially decaying window with parameter $\eta$, similar to the windowing used in defining the path metric for tree search. The resulting LLR is given by

$$\gamma \;\; = \;\; \frac{1 - \eta}{1 - \eta^{k_l - 1}} \times \sum_{i=2}^{k_l} \eta^{k_l - i} \log(\gamma_i). \tag{5.34}$$

A track is declared invalid and terminated when the LLR of the associated path drops below some threshold $\tau$. Analytical calculation of the threshold requires evaluation of the pdf of $\gamma$ for a true target path, denoted by $p_1(\gamma)$, and for a false track, denoted by $p_0(\gamma)$. The random variable $\gamma_i$ is a combination of multiple realizations of the random variables $m_i$ and $p(\mathbf{z}_i^j | \mathbf{x}_i)$, and the LLR is the summation of independent random variables $\gamma_i$. The Linderberg-Feller central limit theorem (CLT) [80, 81] suggests that the summation of independent random variables $\gamma_i$ may converge in distribution to a Gaussian random variable. Although the conditions of the CLT theorem are not met due to a limited sum over the random variables, but a Monte Carlo computer simulation of the distribution shows that the approximation is sufficiently close to be used for calculating the threshold. In Section 5.6, we discuss the Monte Carlo simulations and the statistical tests that are implemented to validate the approximations in more detail. Denoting the probability of declaring a true target track to

be valid by $P_{TT}$, the validation threshold can be calculated from

$$P_{TT} = \int_{\tau}^{\infty} p_1(\gamma) d\gamma, \tag{5.35}$$

and the probability of false alarm can be derived using

$$P_{FA} = \int_{-\infty}^{\tau} p_0(\gamma) d\gamma. \tag{5.36}$$

To evaluate the performance of the proposed track validation approach, we have simulated a single target following a linear Gaussian motion model with nearly constant velocity (NCV). The state consists of two-dimensional position and velocity, and the motion model (5.43) parameters are given by

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.37}$$

The process noise is zero mean with covariance matrix $0.25 I_{2\times2}$, where $I$ denotes the identity matrix. The sensor, which is located at the origin of the coordinate system, is assumed to collect noisy observations of the target position in Cartesian coordinates. The measurement noise, $\mathbf{w}_k$, covariance matrix is given by $0.04 I_{2\times2}$.

The probability of target detection is set to $P_D = 0.7$, and clutter density is $\zeta = 3$. In each realization of the simulation, the target starts from location (10, 10). Tracks are initiated using the Hough transform initiation technique [82]. The Hough transform is performed over cells of dimension $2 \times 2$, and the existence of a target in a cell is declared if 3 out of 5 successive scans provide a peak in the cell. The exponential window parameter

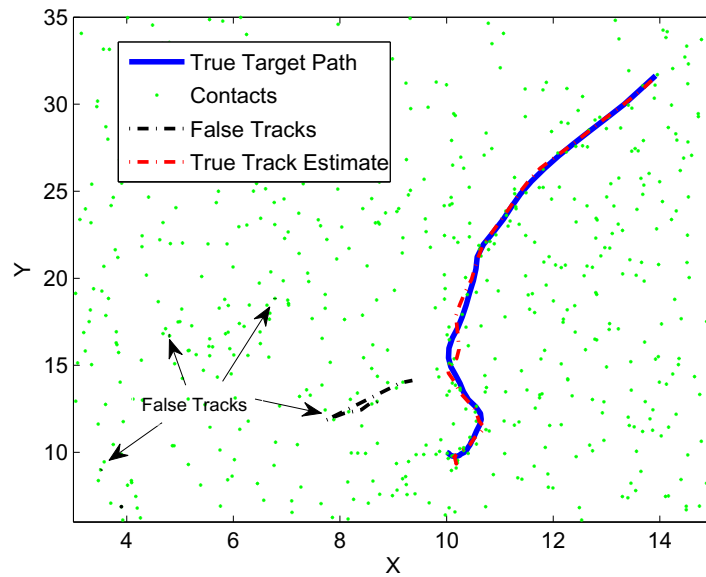Figure 5.10: A sample realization of linear Gaussian target motion tracked by the tree-search tracker. Likelihood-based track validation is used to terminate four false tracks.
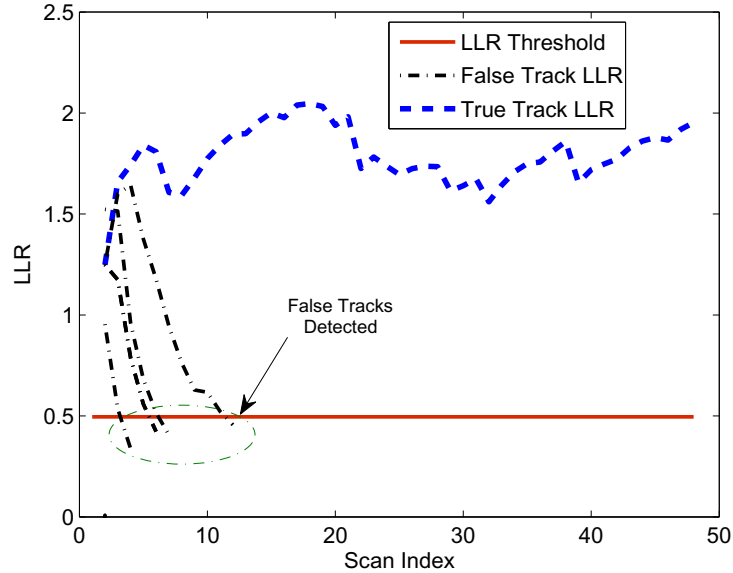


Figure 5.11: LLR values $\gamma$ of the true and false tracks shown in Figure 5.10 as a function of scan index. The $\gamma$ values associated with the false tracks fall below the termination threshold within 10 scans.

for track validation in (5.34) and for path metric calculation in (5.6) is set to $\lambda = \eta = 0.9$. The threshold for track validation is calculated from (5.35) by letting $P_{TT} = 0.9999$ and is equal to $\tau = 0.5$ which results in $P_{FA} = 0.014$. The stack is limited to 256 entries (paths).

A sample realization from this simulation is shown in Figure 5.10. When tracking begins, the initiation algorithm detects the target track, as well as four false tracks. (To simplify the figures, we perform track initiation only for the first five scans. In practice, false tracks would be initiated in later scans, as well.) The LLR values associated with each of the initiated tracks are plotted as a function of scan index in Figure 5.11. While the LLR of the true track remains well above the threshold $\tau$ for all 50 scans, the LLRs of the false tracks quickly drop below the threshold and are terminated within 10 scans.

The LLR-based track validation scheme also identifies when targets have been lost, an example of which is shown in Figure 5.12. The tree-search tracker follows the target well for the first 12 scans but then loses the target when several detections are missed. The track validation algorithm declares the target lost at scan 20, and the target track is reinitiated (again after several missed detections) at scan 28. The LLR values of the original and reinitiated tracks are plotted as a function of scan index in Figure 5.13. Note that the LLR begins to drop rapidly at scan 12 when the target is lost, though 8 scans are required before the LLR drops below the chosen threshold. The number of scans required to identify a lost target is strongly dependent on the LLR memory through $\eta$.

The simulation described above was performed for 200 randomly generated scenarios, and the average performance is presented in Tables 5.1 and 5.2. The results presented in Table 5.1 are derived by generating an environment with no target (only clutter contacts) and using the Hough Transform to initiate false tracks. These tracks are followed until their LLR falls below the threshold, and the length of each track when it is terminated by the track validation algorithm is recorded. For Table 5.2, both target and clutter contacts are available for the first 10 scan. After 10 scans, the target vanishes, and the number of scans required to declare the target lost is recorded. The parameter of the exponentially decaying window is selected as $\eta = 0.9$, yielding a memory of approximately 10 scans. Decreasing
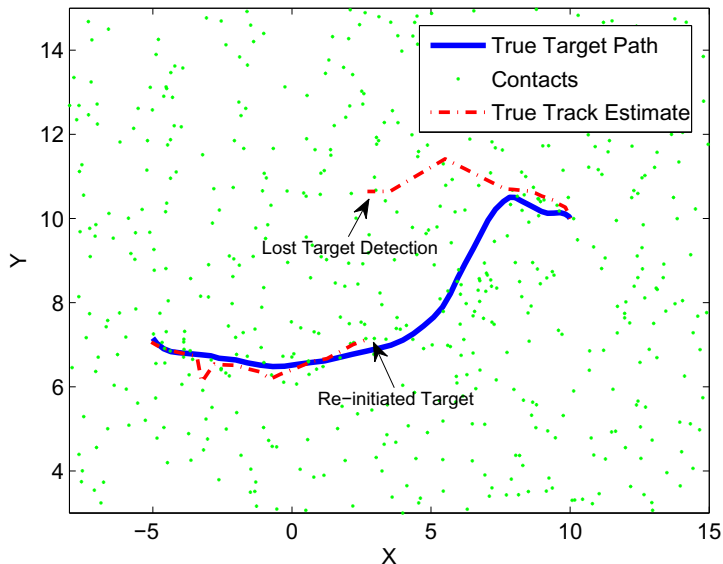
Figure 5.12: An example of a lost target identified via likelihood-based track validation. The track is identified as false 8 scans after the estimated path deviates from the target trajectory. The true target path is re-initiated via Hough transform techniques.
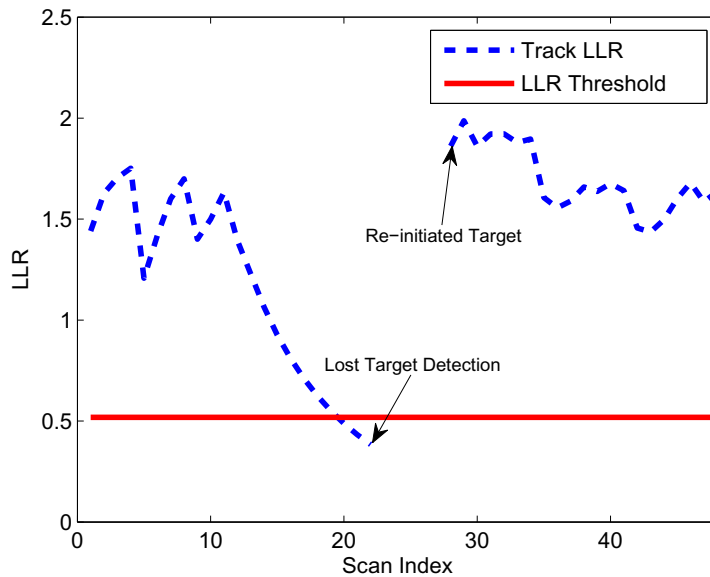


Figure 5.13: LLR values $\gamma$ of the target paths in Figure 5.12 as a function of scan index. The LLR of the path associated with the lost target falls below the termination threshold at scan 20.

$\eta$ decreases the number of scans needed to declare a target lost, but it also increases the probability of declaring a target lost when it is not, particularly when $P_D$ is small. When clutter density increases, the value of the threshold $\tau$ decreases to maintain the same $P_{TT}$. This increases the probability of retaining false tracks in the stack, as well as increasing the number of scans required to declare a lost target.

Table 5.1: Persistence of false tracks under LLR-based track validation

| Clutter density | False tracks remaining after 5 scans | False tracks remaining after 10 scans |
|---|---|---|
| 2 | 18 % | < 1% |
| 3 | 21 % | < 1% |
| 4 | 21 % | < 1% |
| 5 | 22 % | 1.2% |

Table 5.2: Scans required to declare targets lost when LLR-based track validation is employed

| Clutter density | Declared lost after 5 scans | Declared lost after 10 scans |
|---|---|---|
| 3 | 82 % | > 99% |
| 4 | 75 % | > 99% |
| 5 | 73 % | 98% |

## 5.4   Multistatic Tracking and the SEABAR'07 Dataset

Multistatic tracking has become a topic of recent interest in the active sonar community. In a multistatic scenario, multiple geographically distributed receivers are employed to observe the echo from a single source, and hence measurements are obtained from multiple source-receiver pairs. (The multistatic scenario may be extended to include multiple sources, as well, but we consider only a single source in this work.) The presence of multiple receivers provides the benefit of additional information for tracking, but it also presents the challenge of efficiently fusing multiple data sources. As will be shown, the stack-based tracker is

particularly attractive for multistatic tracking, as its path metric can be adapted to exploit information from multiple sources with little to no increase in algorithm complexity.

The focus of this section is the augmentation of tree-search based tracking to address multistatic problems and the evaluation of the resulting tracker on real sonar data. The remainder of this section describes the multistatic system model under consideration, metric derivation for multistatic target tracking, and evaluation of the stack-based tracker on the dataset from the SEABAR'07 experiment.

### 5.4.1 Multistatic Stack Tracker

A multistatic sonar target tracking system must combine the information provided from all the receivers to estimate the target path. Many distributed target tracking techniques are developed to perform whether estimation fusion or data association fusion [83]. In estimation fusion, multiple trackers are estimating the target state based on their own observations and the estimations are combined to estimate the target track. In data association fusion, the measurements provided from multiple receivers are combined to estimate the target track. The algorithms for estimation fusion and data association fusion are combined with the conventional target tracking techniques such as Kalman filtering and can be combined with the stack algorithm as well. In this work, we implement a centralized tree search tracker that approximates the posterior probability density function of the target path given the observations of all the receivers.

Figure 5.14 shows the architecture of a centralized tree search tracker. In a central fusion scenario, observations of all the receivers are transferred to a central target tracker where the tracking is performed. The central target tracker implements a tree search tracker and updates the search tree based on the observations from all the receivers. We assume that the source and receiver locations are known (at least approximately) to the tracker and that is a single source and $\Sigma_w$ receivers. Similar to the single target tracking model, let the vector $\mathbf{x}_k$ denote the state of the target at time (or scan) index $k$. In the model under consideration, each scan produces a set of contacts at each of the $\Sigma_w$ receivers. (A receiver

may not be functioning at a particular scan, in which case the associated set of contacts will be empty.) One could assume an equivalent model in which the contacts at each receiver are viewed as different scans, and hence one ping from the source results in $\Sigma_w$ scans of data.

Let the contacts observed by receiver $r$ at time $k$ be denoted by the $m_k^{(r)}$-column matrix $\mathbf{Z}_k^r$, where $m_k^r$ denotes the number of contacts observed by receiver $r$ in scan $k$ and each column of the matrix $\mathbf{Z}_k^r$ represents an individual contact. A target-generated contact appears in the measurement matrix of each receiver with probability $P_D^r$. For purposes of deriving data association probabilities, $m_k^r$ is assumed to follow a Poisson distribution. A full contact matrix for each scan is formed as $\mathcal{Z}_k = \begin{bmatrix} \mathbf{Z}_k^1 \ \mathbf{Z}_k^2 \ \cdots \ \mathbf{Z}_k^{\Sigma_w} \end{bmatrix}$. The target motion model is as given in (5.1), and the measurement models for the $r$th receiver is given by

$$\mathbf{z}_k^r \;=\; \mathcal{H}(\mathbf{x}_k) + \mathbf{w}_k^r, \tag{5.38}$$

where $\mathbf{w}_k^r$ denotes the observation noise at receiver $r$. The observation noise vectors $\mathbf{w}_k^r$ are assumed to be independent of the state transition noise and independent of the observation noise at other receivers and at different scans. While the stack-based tracker we consider assumes knowledge of $\mathcal{F}(.)$ and $\mathbf{G}$, a model of the target motion will not be known to the tracker in most practical scenarios; in fact, such a model may not exist for targets whose motion involves maneuvering, etc. In practical scenarios, however, motion models are assumed within the tracker, even though they are not followed by the target.

Similar to Section 5.2.3, we use an exponentially decaying window to generate a fair metric for comparing paths of different lengths:

$$b^{(l)} = \frac{1 - \lambda}{1 - \lambda^{k_l - 1}} \times \sum_{i=2}^{k_l} \lambda^{k_l - i} \log \left[ p(\mathcal{Z}_i | \mathbf{x}_i^{(l)}, \mathbf{x}_{i-1}^{(l)}) p(\mathbf{x}_i^{(l)} | \mathbf{x}_{i-1}^{(l)}) \right], \tag{5.39}$$
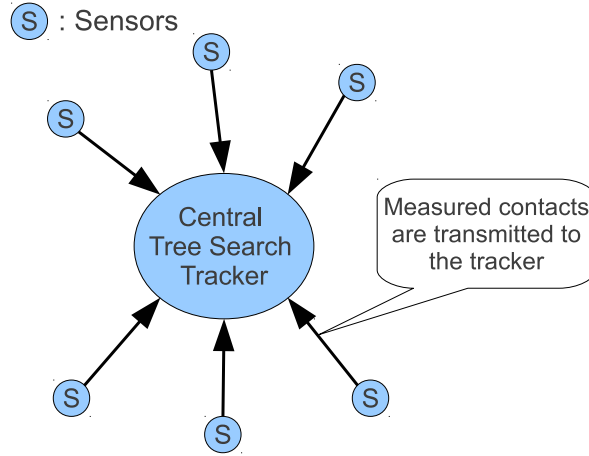
Figure 5.14: Architecture of a central fusion scenario. The information from all sensors is collected at a single fusion center at which centralized tree-search based tracking is performed.

where the state transition probability mass function $p(\mathbf{x}_i^{(l)}|\mathbf{x}_{i-1}^{(l)})$ is calculated in (5.5). Calculation of the path metric requires evaluation of the observation likelihood conditioned on the target state $p(\mathcal{Z}_i|\mathbf{x}_i^{(l)})$ for $i = 1$ to $k_l$. When multistatic geometries are employed, calculation of $p(\mathcal{Z}_l|\mathbf{x}_l^{(l)})$ must be modified to reflect the fusion of data streams obtained from multiple source-receiver pairs. Under the assumption that observations at any single receiver are conditionally independent of observations at all other receivers, we compute the multistatic observation likelihood as

$$p(\mathcal{Z}_i|\mathbf{x}_i^{(l)}) = \prod_{r=1}^{\Sigma_w} p(\mathbf{Z}_i^r|\mathbf{x}_i^{(l)}), \tag{5.40}$$

where $p(\mathbf{Z}_i^r|\mathbf{x}_i^{(l)})$ can be evaluated by expanding over all data association hypotheses (5.7). In fact, this is a simplifying assumption that generally does not hold true in practice. While conditioning on the target state eliminates dependence among contacts generated by the target (since the only remaining random element is observation noise), contacts that are

observed at different receivers but generated by the same clutter source will certainly be correlated. Hence, the chosen implementation is a simple first approximation to a more accurate approach to observation fusion. The new observation likelihood can be applied in discretization technique as discussed in Section 5.2.5.

## 5.4.2   Performance Evaluation on SEABAR'07 Dataset

The performance of the stack-based tracker for multistatic sonar has been evaluated on the SEABAR'07 experiment. The SEABAR'07 experiment took place in the Mediterranean Sea in October 2007 [84]. The experiment included a single source and three receivers. Both continuous wave (CW) and frequency modulated (FM) pulses were transmitted by the source at one-minute intervals. The result of two different experiments, known as A01 run and A56 run, is available for test. All three receivers were active during part, if not all, of run A01, and receivers 2 and 3 were active during run A56. The target was simulated by a towed echo repeater. In order to provide contact data that better model a true target, the SNR levels of the target contacts were modified to reflect aspect-dependent target strength [85]. The results presented here use contact data modified according to the BASIS target strength model [86].

The source and receiver locations, as well as the true target trajectory, were provided in terms of latitude and longitude. Conversion from time delay to contact range was performed as described in [87]. Source, receiver, and target locations were converted to a 2-D Cartesian coordinate system using the Haversine formula [88], and the origin of the coordinate system was placed at the source. The source and receiver locations for the SEABAR'07 experiment, along with the target trajectories for runs A01 and A56, are shown in Figure 5.15.

The stack-based tracker requires an initial estimate of the target state. We use the Hough transform for track initiation [82] to generate this information. The surveillance region is divided into 1-km$^2$ cells, each of which is evaluated for the presence of a target using peaks in the Hough bins. The common $M$-of-$N$ approach is employed, meaning that a track is initiated if $M$ out of $N$ observations in a sequence generate a peak. The track
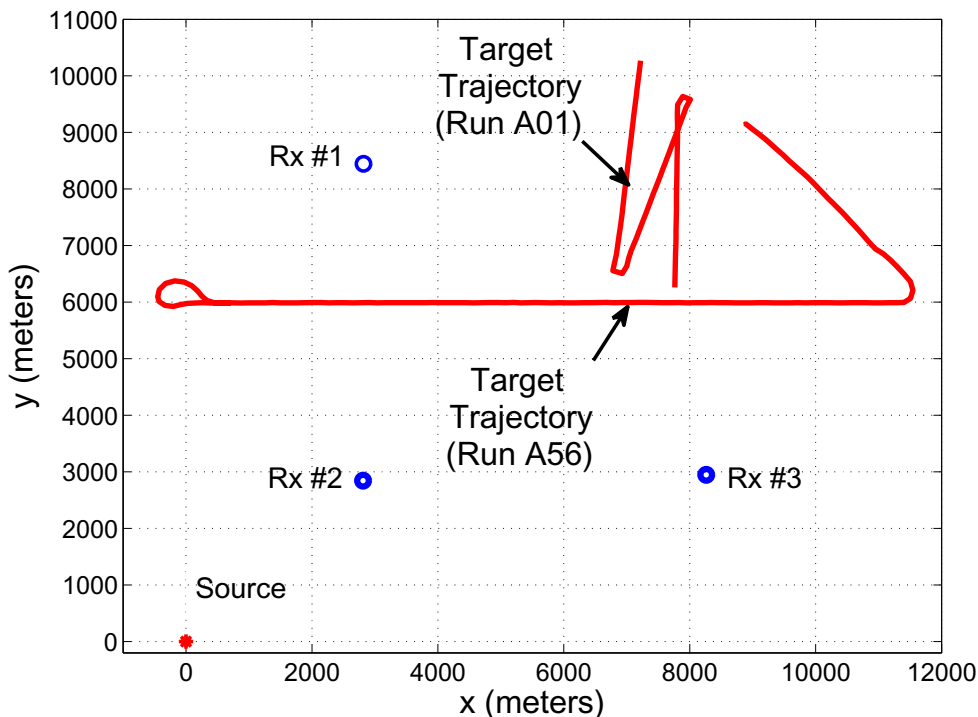
Figure 5.15: Geometry of the SEABAR'07 multistatic sonar experiment. The source, shown in red, is at $(0,0)$. The three receivers are shown as blue circles, and the target trajectories for runs A01 and A56 are shown in red.

initiation algorithm is repeated every 10 scans to detect newly arriving targets. The validity of each track is evaluated at each scan via the stack metric. During tracking, we use the track validation technique as discussed in Section (5.3) and if the LLR of an initiated target path falls below the threshold of the track validity test, the target is marked as invalid and is terminated by the algorithm. The value of $M/N$ used for track initiation depends upon the clutter density of the experiment. The values used for runs A01 and A56 are given in Table 5.3.

The target state vector is given by $\mathbf{x}_k = [x_k \ y_k \ \dot{x}_k \ \dot{y}_k]^T$, where $(x_k, y_k)$ denotes the position of the target in Cartesian coordinates, and $(\dot{x}_k, \dot{y}_k)$ denotes the target velocity in Cartesian coordinates. Computation of the Bayesian path metric requires that motion and measurement models be provided. Of course, no model is available to describe the motion

of the targets in the SEABAR experiment; a simple nearly-constant velocity (NCV) linear motion model [89] is assumed as a default. While extension to interacting multiple models [90] or other more sophisticated models may provide performance improvement, the stack-based search is able to hold track under the NCV model. The assumed motion model is given by

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + \mathbf{G}\mathbf{v}_k, \tag{5.41}$$

where

$$F = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G = \begin{bmatrix} \frac{\Delta_t^2}{2} & 0 \\ 0 & \frac{\Delta_t^2}{2} \\ \Delta_t & 0 \\ 0 & \Delta_t \end{bmatrix}. \tag{5.42}$$

The time between scans, $\Delta_t$ is equal to 1 minute. The observation vector for the FM contacts includes bearing and time difference of arrivals (TDOA) and for the CW contacts includes bearing, TDOA, and range rate. The observation function $\mathcal{H}(.)$ that relates the target location to bearing, TDOA, and range rate is derived in Appendix A. The covariance matrices of the $\mathbf{v}_k$ and $\mathbf{w}_k$, denoted by $\Sigma_v$ and $\Sigma_w$, respectively, are given in Table 5.3. An SNR threshold of 12 dB is used to determine which contacts are provided to the tracker.

For evaluation of the stack-based multistatic tracker, we first consider run A01 of the SEABAR'07 experiment. This run is particularly challenging, as the target engages in both a hairpin turn and a Crazy Ivan maneuver. The target begins near $(7000, 10000)$ and travels first through the hairpin turn followed by the Crazy Ivan. Figure 5.16 shows the performance of the stack-based tracker using only FM data from all three receivers. The stack-based tracker is able to maintain track throughout the target trajectory, including through the hairpin and Crazy Ivan turns. The tracker holds the hairpin turn particularly tightly, showing little to no deviation from the highly nonlinear maneuver. The initiation

Table 5.3: Stack-based tracker parameter values used for evaluation on run A01 of the SEABAR experiment.

| | |
|---|---|
| Contact SNR Threshold | 12 dB |
| $\mathbf{P}_D$ (Prob. of Detection) | 0.75 |
| $M/N$, run A01 | 3/5 |
| $M/N$, run A56 | 2/5 |
| $\Sigma_w$ (covariance of $\mathbf{w}_k$), FM | $\begin{bmatrix} 0.0019 & 0 \\ 0 & 2500 \end{bmatrix}$ |
| $\Sigma_w$ (covariance of $\mathbf{w}_k$), CW | $\begin{bmatrix} 0.0019 & 0 \\ 0 & 62500 \end{bmatrix}$ |
| $\Sigma_v$ (covariance of $\mathbf{v}_k$) | $\begin{bmatrix} 0.0005 & 0 \\ 0 & 0.0005 \end{bmatrix}$ |
| $\lambda$ (Forgetting factor) | 0.95 |
| Stack Size | 256 |
| Discretization Regions | 9 |
| Samples Per Region | 81 |

algorithm generates four false tracks at scan indices 21, 31, 41, and 61. The first and the third false tracks are terminated after 34 and 15 scans, while the second and fourth persist to the end of the experiment.

Figure 5.17 shows the performance of the stack-based tracker on run A01 using only CW data from all three receivers. Bearing, TDOA, and range rate information are used by the tracker. Like the FM case, the tracker follows the target trajectory throughout its maneuvers. The tracker does not follow the trajectory as closely in this case as it did when FM data was used since the measurement noise is much higher in CW data. In this experiment, three false tracks have been generated: one at the scan one and two at scan 21. The first false track is killed after 20 scans; one of the remaining false tracks lasts for 59 scans, and the other persists to the end of the experiment. Only parts of the false track are shown in Figure 5.17.

The performance of the stack-based tracker on run A01 when both FM and CW data are provided is shown in Figure 5.18. For simplicity, only CW data is used in the Hough

Figure 5.16: Tracking performance of the stack-based tracker on run A01 using FM contacts from all three receivers. The target trajectory is shown in red, the track estimate is shown in blue, and false tracks are shown in black.



Figure 5.17: Tracking performance of the stack-based tracker on run A01 using CW contacts from all three receivers. The target trajectory is shown in red, the track estimate is shown in blue, and false tracks are shown in black.

transform for track initiation, allowing bin size and thresholds to remain the same as those used when only CW or FM data alone is used for tracking. The tracker maintains track

Figure 5.18: Tracking performance of the stack-based tracker on run A01 using both FM and CW contacts from all three receivers. The target trajectory is shown in red, the track estimate is shown in blue, and false tracks are shown in black.
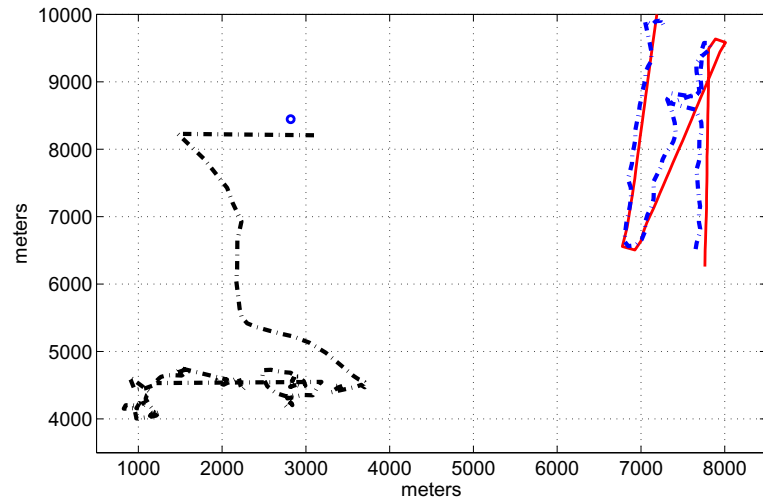
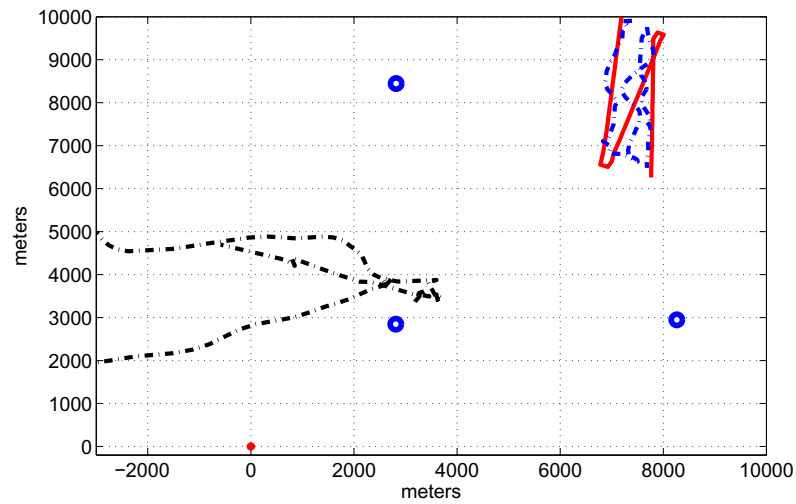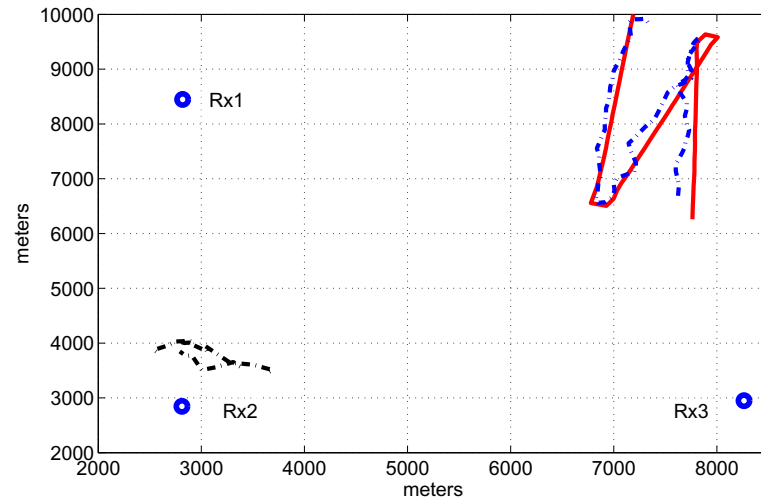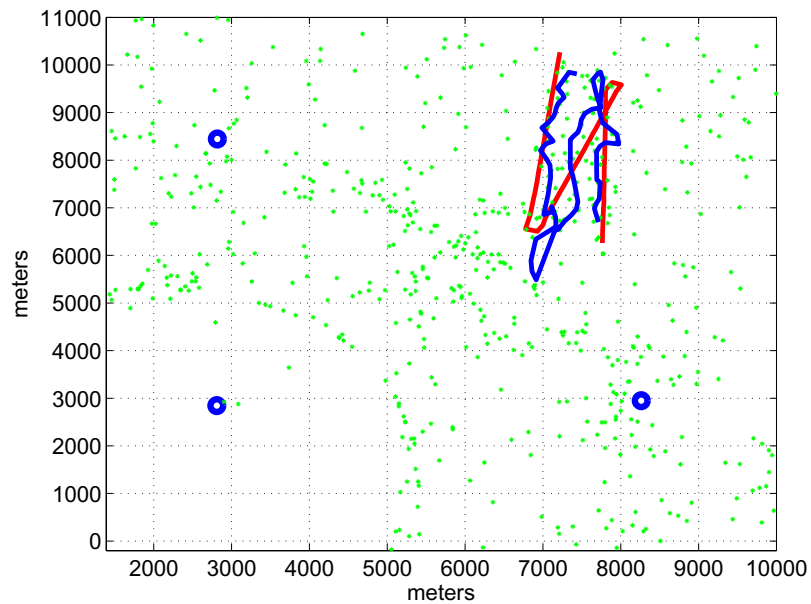

Figure 5.19: Tracking performance of the stack-based tracker on run A01 using only CW contacts observed at Receiver 2. The target trajectory is shown in red, and the track estimate is shown in blue. Contacts are shown in green.

throughout the trajectory, and performance is slightly better than FM alone. Since we have used only CW data for track initiation, the same false tracks appear here and in Figure 5.17. Using both FM and CW information, all false tracks are killed within 9 scans. When both data types are used, the multistatic likelihood is computed by treating each receiver as two data streams, one for each pulse type. Again, assumptions of independence across these contacts do not necessarily hold in practice, and methods for addressing correlations among contacts are a topic of further work.

To evaluate the performance of the stack-based tracker when only a single stream of contact data is available, we consider the scenario in which, for run A01, only CW data from Receiver 2 is provided. The results are shown, along with Receiver 2 CW contacts, in Figure 5.19. Even when only one receiver is used, the stack-based tracker succeeds in following the target through both maneuvers. In fact, the performance using CW data from a single receiver is comparable to (if not better than) the performance using CW data from all three receivers. This may indicate that the current technique for fusing data from multiple receivers (e.g. assuming independence) is causing minor performance degradation. Note that Receiver 2 does not observe contacts in the region of the false track seen in Figures 5.17 and 5.18, and hence no false tracks are initiated when only Receiver 2 data is used.

The performance of the stack-based tracker has also been evaluated on run A56 of the SEABAR'07 dataset. The target trajectory for run A56 begins to the left near $(600, 6000)$ with a loop and then travels straight for approximately 12 km before making a final turn. Figure 5.20 shows the performance of the stack-based tracker on run A56 when CW contacts are used for track initiation and both CW and FM contacts are used for tracking. Receivers 2 and 3, which were active for part or all of the run, are shown along with the target trajectory, the track estimate, and two false tracks. The stack-based tracker maintains track throughout the run, including the loop at the beginning and the turn near the end. Initiation with CW contacts results in four false tracks. Two are initiated at the first scan and last for 6 and 14 scans, respectively. Another false track is initiated at scan 51 and persists through 39 scans. At time index 91, a final false track is initiated and lasts 10

Figure 5.20: Tracking performance of the stack-based tracker on run A56. The target trajectory is show in red, the track estimate is shown in blue, and the false tracks are shown in black. Both CW and FM contacts are used for tracking. Only CW contacts are used for track initiation.

scans.

Finally, the performance of the stack-based tracker on run A56 when only CW data is provided for initiation and only FM data is provided for tracking is shown in Figure 5.21. The tracking performance using only FM data is very similar to that using both FM and CW data. In both cases, the target track is maintained throughout the run. Similar to Figure 5.20, four false tracks are identified; they are terminated in 8, 20, 12, and 23 scans, respectively.

## 5.5 Stack-based Multitarget Tracking

For many applications, particularly those that fall within the broader class of surveillance, the goal is not to track a single target over time but to detect and track all targets present in the field of view. For these applications, tracking techniques are required to estimate the
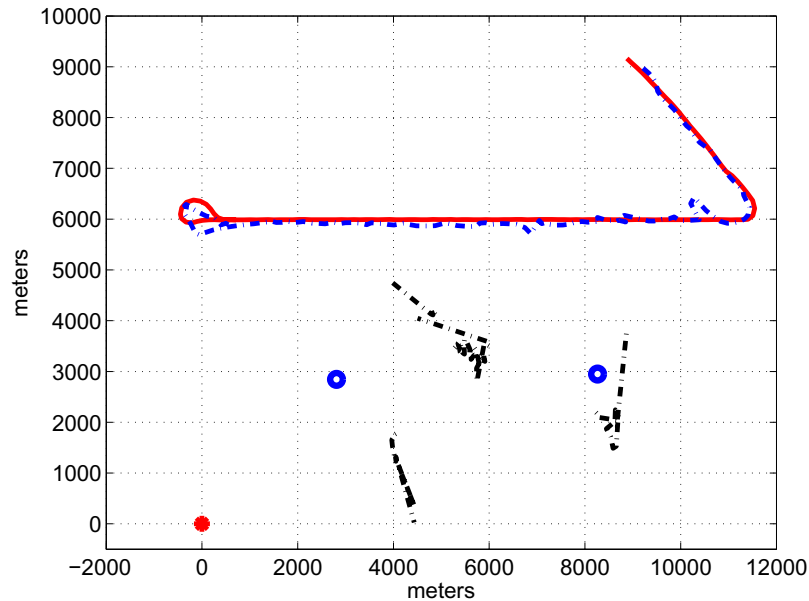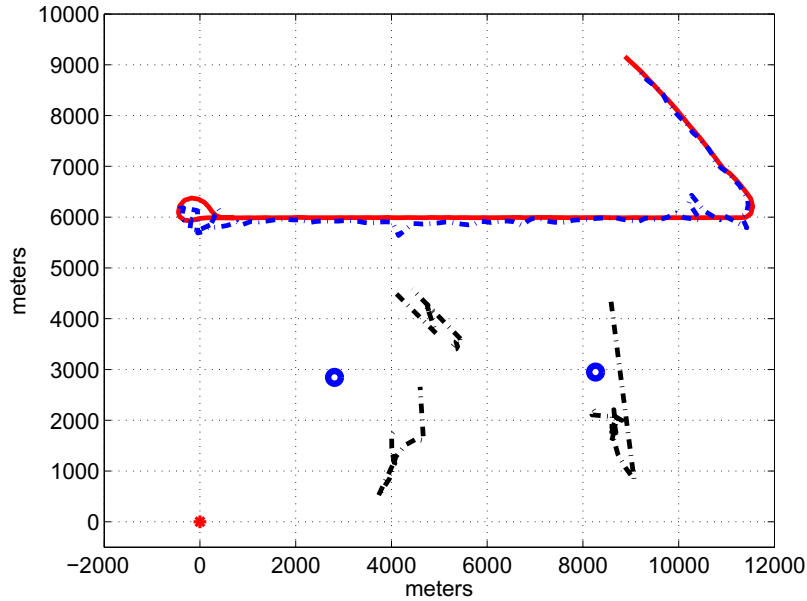
Figure 5.21: Tracking performance of the stack-based tracker on run A56. The target trajectory is shown in red, the track estimate is shown in blue, and the false tracks are shown in black. Only FM contacts are used for tracking, and only CW contacts are used for track initiation.

states of multiple targets and to perform association of contacts with targets. The focus of this section is on using tree-search techniques for multi-target tracking. We develop the stack algorithm for tracking multiple targets, and we evaluate its effectiveness in tracking the targets that are crossing or moving in close physical proximity.

## 5.5.1  System Model

To model the multi-target tracking problem, we consider a collection of targets traveling through a surveillance region. The number of targets in the region is denoted by $T$, and each target is assigned a unique identification number $t$, $t = 1, 2, \ldots, T$. Dynamic state space models govern the evolution of the targets' states and the observation of the targets.

The state of each target evolves according to the state process equation, given by

$$\mathbf{x}_k^t = \mathcal{F}^t(\mathbf{x}_{k-1}^t) + \mathbf{G}^t \mathbf{v}_k^t, \quad t = 1, \ldots, T. \tag{5.43}$$

A sensor associated with the tracking system periodically scans the surveillance area, and sensor observations are processed to form contacts, which may be generated by targets and/or clutter. In this section, we consider a single sensor problem to simplify the notations. Extension to allow for multiple sensors and multistatic sensing geometries is straightforward using the same approach presented in Section 5.4.

At each scan, the tracker forms a set of contacts $\mathbf{Z}_k = [\mathbf{z}_k^1 \ \mathbf{z}_k^2 \ \cdots, \ \mathbf{z}_k^{m_k}]$, where $m_k$ denotes the number of contacts obtained in scan $k$. A target-generated contact in the observation matrix is related to the target state vector by the observation model (5.2). The probability that target $t$ is detected by the sensor is denoted by $P_D^t$. As in previous sections, we assume that each contact is generated by at most one target. Other parameters of the DSSM and the distribution of the clutter generated contacts are the same as the single target model presented in Section 5.2.1.

## 5.5.2 Tree-Search Tracker Development for Multiple Target Tracking

The stack algorithm navigates a tree in which each path corresponds to a possible sequence of states visited by a target. The search tree is initiated using estimated target locations provided by an external track initiation algorithm. A target ID $t$ is associated with each initial target estimate; the ID is carried with all path extensions to differentiate among the paths associated to different targets. We describe an implementation in which a single search tree is used to track all targets in the surveillance area. Hence, the initiation step consists of storing all initial target state estimates in the stack. One could consider an equivalent implementation in which individual trees (and hence individual stacks) are used to track each target in view. Figure 5.22 provides a flow chart summarizing the tree search procedure for multiple target tracking.

Without loss of generality, consider the procedure as applied to the target with ID $t = 1$. All steps described are applied only to paths for which $t = 1$. Each time a new scan is completed and the corresponding contacts are obtained by the tracker, the tracker identifies the path with the largest metric, e.g. the most likely path. (If the stack has been
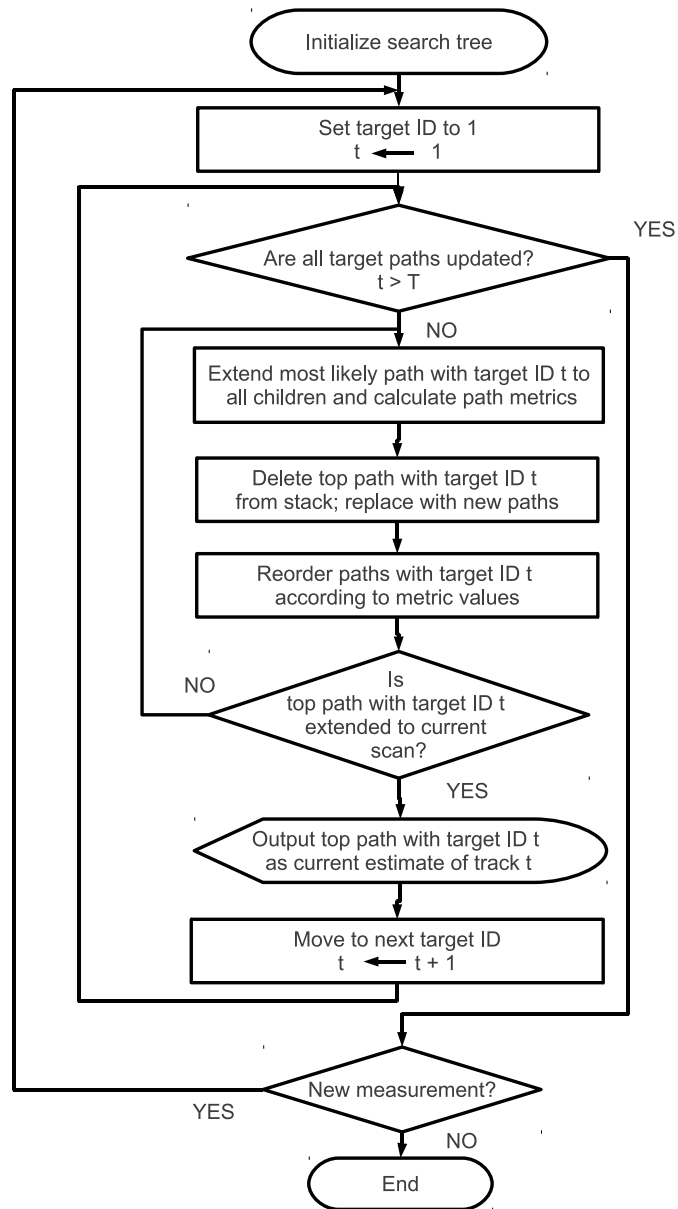
Figure 5.22: Flow chart describing multi-target tracking via stack-based tree search.

sorted by metric value, this path is at the top of the stack.) The most likely path is extended to all possible children; the metrics of the extended paths are calculated, and the paths are placed in the stack. The parent (shorter) path is removed from the stack. If the most likely path in the stack is extended to the current scan, its current state estimate is provided as tracker output, and the algorithm waits for the next set of contacts to be obtained. If

the most likely path contains fewer branches than the number of scans completed, it is again extended to its children, and the process continues until the most likely path reaches a length equal to the number of scans completed. Mathematical expressions for the path metric are derived in Section 5.5.3.

Multi-target tracking introduces the challenge of estimating the states of targets that travel in close proximity and may cross. When targets are near each other, joint association of contact data to targets is typically required to provide strong tracking performance. Figure 5.23 provides a pictorial representation of the stack algorithm tracking two targets that move into close proximity. At initiation of the stack algorithm, the two targets have distinct location estimates. The initial node for each target is extended to a set of possible children (discretized states), and the posterior likelihoods are calculated based on the contacts in scan 1. The most likely path for each target is again extended and metrics calculated based on the contacts in scan 2. Notice that, for target 1, the stack algorithm "looks back" and extends a second path from scan 1 to scan 2 before finding the most likely path in scan 2 to extend to scan 3. At scan 3, the regions covered by the paths associated with targets 1 and 2 overlap. In this situation, the state estimates for both paths must be considered when calculating the individual metrics of the paths associated with each target.

### 5.5.3 Derivation of the Path Metric for Multi-Target Tracking

In Section 5.2.3, we developed the path metric for a single target tracking scenario. When multiple targets are present, the path metric for each target is related to the states of the other targets and hence cannot be evaluated independently. To implement the MAP criterion, the path metric must be proportional to the posterior probability density of the target state conditioned on the observations from the sensor. At scan index $k$, the posterior pdf of the $l$th candidate path for target $t$, given the observation matrix $\mathbf{Z}_k$ and the best estimate (e.g. most likely path in the stack) of the states of all other targets at scan index
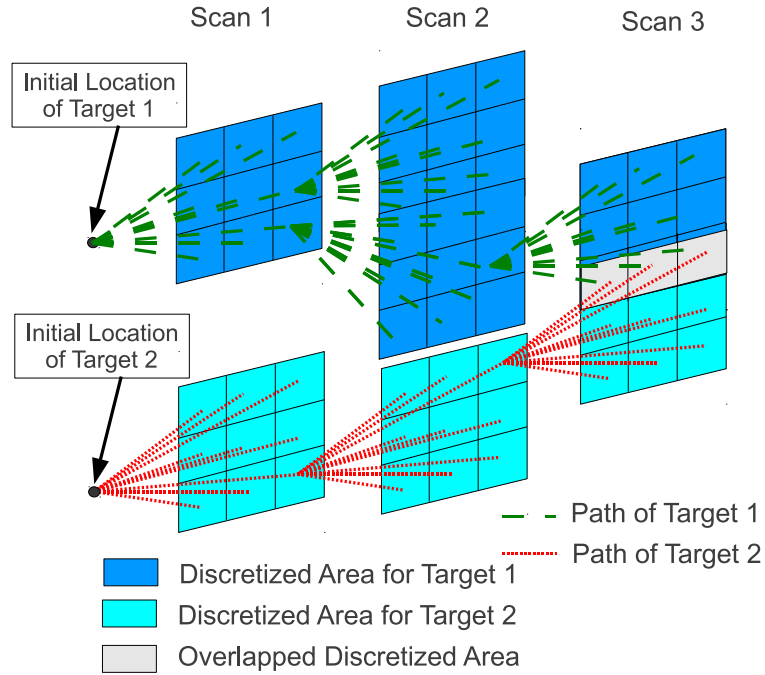
Figure 5.23: A graphical example of tree-search based multi-target tracking using the stack algorithm. The two targets move progressively closer to each other during the three scans shown.

$k - 1$, can be written as

$$p(\mathbf{x}_k^{t,(l)}|\mathbf{Z}_k, \hat{\mathbf{x}}_{k-1}^1, \cdots, \hat{\mathbf{x}}_{k-1}^T) \;\; = \;\; \frac{p(\mathbf{Z}_k|\mathbf{x}_k^{t,(l)}, \hat{\mathbf{x}}_{k-1}^1, \cdots, \hat{\mathbf{x}}_{k-1}^T)p(\mathbf{x}_k^{t,(l)}|\mathbf{x}_{k-1}^{t,(l)})}{p(\mathbf{Z}_k|\hat{\mathbf{x}}_{k-1}^1, \cdots, \hat{\mathbf{x}}_{k-1}^T)}.$$

Calculation of $p(\mathbf{Z}_k|\hat{\mathbf{x}}_{k-1}^1, \cdots, \hat{\mathbf{x}}_{k-1}^T)$ would require a summation over all possible discretized target states $\mathbf{x}_k$, but the tree search evaluates the likelihood of only a subset of the possible states. Note that this term is equal for all paths extended to scan index $k$. Particularly when the clutter density is high, the variation of the denominator term with $k$ will be negligible, and hence we omit it in generating the path metric. The path metric is constructed using the same definition as in (5.6) knowing that the likelihood function considers the estimated states of all the targets (5.44).

Due to the presence of multiple targets and clutter in the surveillance region, calculation of the likelihood function $p(\mathbf{Z}_i|\mathbf{x}_i^{t,(l)}, \hat{\mathbf{x}}_{i-1}^1, \cdots, \hat{\mathbf{x}}_{i-1}^T)$ must include data association. In Section 5.2.4, we have implemented a data association technique, similar to the PDAF technique, by expanding the likelihood function over all possible association hypotheses. The same approach can be applied for multiple target tracking and similar to the JPDAF, the association events must consider the existence of multiple targets. Letting $\beta$ denote a data association event (e.g. an association of contacts observed with targets in track), the likelihood function can be expanded over all possible association hypotheses as

$$p(\mathbf{Z}_i|\mathbf{x}_i^{t,(l)}, \hat{\mathbf{x}}_{i-1}^1, \cdots, \hat{\mathbf{x}}_{i-1}^T) = \tag{5.44}$$

$$\sum_{\beta} p(\mathbf{Z}_i|\mathbf{x}_i^{t,(l)}, \hat{\mathbf{x}}_{i-1}^1, \cdots, \hat{\mathbf{x}}_{i-1}^T, \beta)p(\beta|\mathbf{x}_i^{t,(l)}, \hat{\mathbf{x}}_{i-1}^1, \cdots, \hat{\mathbf{x}}_{i-1}^T).$$

When multiple targets are present in the surveillance region, we apply the constraint that a contact can be assigned to at most one target. (This is in contrast to algorithms such as MHT that allow one contact to be assigned to multiple measurements in order to reduce data association complexity.) This constraint is relevant, for example, in crossing targets scenarios or when two targets are moving in close proximity. If the constraint is not applied, two crossing targets may follow the stronger of the target contacts, resulting in both paths following a single (stronger) target and the weaker target being lost. For ease of explanation, consider two targets traveling in close proximity. Similar to existing approaches in evaluation of the likelihood function [91], there are four possible detection scenarios: neither target is detected, only one target is detected, and both targets are detected. The likelihood function

for target 1 in the two-target scenario is calculated as

$$p(\mathbf{Z}_i|\mathbf{x}_i^{1,(l)}, \hat{\mathbf{x}}_{i-1}^2) = \tag{5.45}$$

$$(1 - P_D^1)(1 - P_D^2)\mu_F(m_i)\frac{1}{V^{m_i}}$$

$$+\mu_F(m_i - 1)\frac{1}{m_i V^{m_i-1}}\left(P_D^1(1 - P_D^2)\sum_{j=1}^{m_i}p(\mathbf{z}^j|\mathbf{x}_i^{1,(l)}) + P_D^2(1 - P_D^1)\sum_{j=1}^{m_i}p(\mathbf{z}^j|\hat{\mathbf{x}}_{i-1}^2)\right)$$

$$+P_D^1 P_D^2 \mu_F(m_i - 2) \times \frac{1}{m_i(m_i - 1)V^{m_i-2}}\left(\sum_{j=1}^{m_i}\sum_{k=1,k\neq j}^{m_i}p(\mathbf{z}_i^j|\mathbf{x}_i^{1,(l)})p(\mathbf{z}_i^k|\hat{\mathbf{x}}_{i-1}^2)\right),$$

and the likelihood function for target 2 is computed similarly. Note that the likelihood function for each target is dependent on estimates of other target locations; the effect of other targets will be significant when they are close to the target under consideration.

One of the advantages of the stack algorithm is its ability to jump backward in the tree when a shorter path appears more likely than the paths extended to the current scan index. This feature gives tree-search based tracking the chance to "reconsider" when the trajectory followed begins to look less likely. The jump-back capability of the stack algorithm also poses a unique challenge for data association, however. The metric of any path in the tree depends upon the current and previous location estimates of all targets in the surveillance region. Hence, when the stack algorithm looks back and extends a shorter path in the tree, the resulting changes in the estimated location of a target impacts the metrics of paths associated with other targets. Recalculation of all metrics to reflect changes in previous estimates is computationally prohibitive. Instead, we propose applying the results of JPDAF-based data association only in forward path extensions. In other words, when a past state estimate changes due to path extension, we modify the metrics only of newly-extended paths and avoid recalculating existing path metrics. Since the dynamic state space discretization is also a function of the locations of all targets, implementing forward-only metric modification also avoids recalculation of discretization regions.

### 5.5.4 Simulation of Crossing and Maneuvering Targets with Track Validation

We examine the performance of the tree-search based tracker with both multi-target association and LLR-based track validation incorporated. We have simulated the algorithm for two crossing and maneuvering targets. The targets begin by following the NCV motion model with parameters given by (5.37). The initial state of the first target is set to $[9 \ 10 \ 0.075 \ -0.075]^T$, and the initial state of the second target is $[11 \ 10 \ -0.075 \ -0.075]^T$. The initial velocities are chosen such that the target paths cross with high probability. After 30 scans, the targets begin to turn in the direction opposite their velocity with a fixed acceleration; the turn continues for 10 scans. For the final 10 scans of the simulation, the targets return to following the NCV model with random (Gaussian) acceleration. For the simulations in this section, target detection probability is given by $P_D = 0.85$, and the clutter density $\zeta$ varies between 0.5 and 2. Because the crossing and maneuvering targets are much more challenging to track than the linear Gaussian target in Section 5.3, lower clutter density levels were chosen for evaluation. The sensor measures the target location in polar coordinates, e.g. bearing and range. The measurement function is given by

$$\mathcal{H}(\mathbf{x}_k) = \begin{bmatrix} \tan^{-1}(\frac{y_k}{x_k}) \\ \sqrt{x_k^2 + y_k^2} \end{bmatrix}. \tag{5.46}$$

The measurement noise $\mathbf{w}_k$ is assumed Gaussian with zero mean and standard deviation of 0.1 degrees in bearing and 0.01 distance units in range.

To track the targets, we have applied the tree-search tracker with and without track validation, and we have compared its performance to that of the EKF combined with the joint probability data association filter (EKF-JPDAF) [77], and the PF [68]. The proposed PF algorithm implements 2000 particles. The importance sampling function is selected using (5.23) and the likelihood function is calculated considering joint probabilistic data
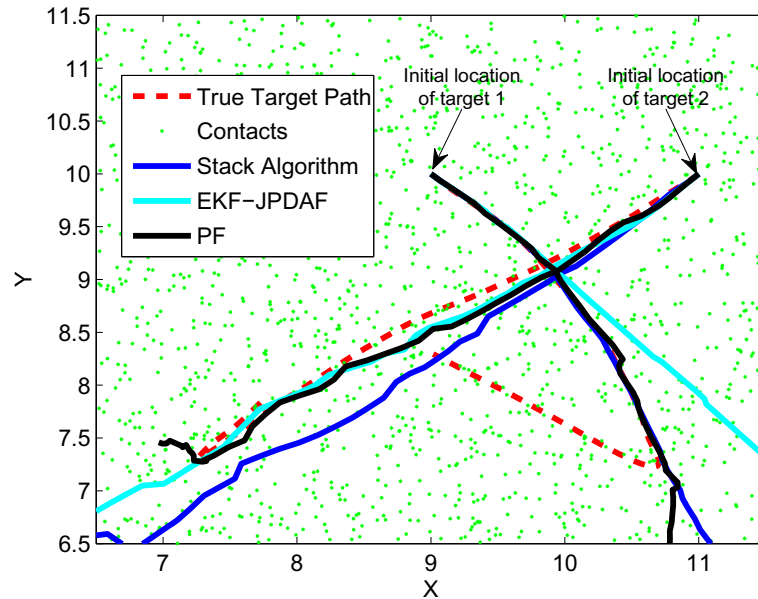
Figure 5.24: A sample realization of two crossing and maneuvering targets. Both the tree-search tracker and the EKF-JPDAF lose the targets when tight turns are initiated. Track validation is not integrated in the tree-search tracker in this example.

association (5.45). The re-sampling step is performed when the number of effective particles drops below one tenth of the total number of particles. As in Section 5.3, a stack size of 256 is used for all tree-search tracker simulations. The LLR threshold $\tau$ is set to 3, which corresponds to $P_{TT} = 0.99$ and $P_{FA} = 0.015$. The parameter of exponentially decaying window for track validation is selected as $\eta = 0.8$, yielding a memory of approximately 5 scans to declare a lost target. A sample realization of the crossing and maneuvering targets is shown in Figure 5.24. The clutter density is $\zeta = 1$ for this realization. The track estimates generated by the tree-search tracker without track validation, the EKF-JPDAF, and the PF are also shown. The tree-search tracker follows target 1 tightly and target 2 loosely until the two targets enter turns at scan 30. At this point, the tree search tracker loses both targets and instead follows clutter. The EKF-JPDAF follows target 2 closely until the turn, but it begins to lose target 1 even before the turn is initiated. The PF, however, is able to track both targets before the turn is initiated.
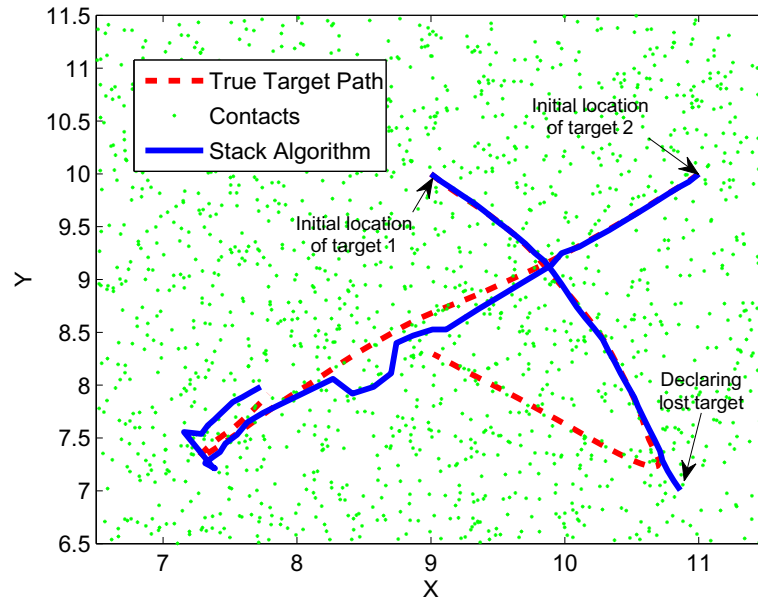
Figure 5.25: The same realization of crossing and maneuvering targets as shown in Figure 5.24, but likelihood-based track validation is now integrated in the tree-search tracker. With validation, the tracker can hold target 2 through the turn and can identify target 1 as lost when the turn begins.

To illustrate the benefits of integrating track validation within the tree-search based tracker, we consider the same realization but now with validation incorporated in the stack algorithm. The results are shown in Figure 5.25. To integrate track validation within the tree-search tracker (rather than treating it as an external element), we simply remove from the stack any target paths whose LLR is below the threshold $\tau$. Since the path likelihood is evaluated as part of the metric calculation in (5.44), calculation of the LLR does not impose any additional computational load on the tracker. In addition, culling the stack based on path LLR improves performance for fixed memory by freeing stack space for more promising paths. Figure 5.25 shows that the tree-search tracker with integrated track validation tracks target 2 much more tightly than without validation, and it is able to follow target 2 through the hairpin turn. While the enhanced tree-search tracker doesn't follow target 1 through the turn, the validation scheme does recognize that the target has been lost and terminates the track.

Two hundred realizations of the crossing and maneuvering targets scenario were simulated, and the average performance results are summarized in Table 5.4. The tree-search, the EKF-JPDAF, and the PF trackers were evaluated based on the percentage of targets lost, where a lost target was defined as one for which the distance between the true and estimated tracks was at least twice the standard deviation of the measurement noise. All simulated algorithms were able to follow all targets through the cross. The averaged results show that the tree-search tracker outperforms the PF and is able to reduce the number of lost targets by a factor of approximately three relative to the EKF-JPDAF. Target loss, when it occurred, typically occurred during the tight target turn, confirming the performance advantage of the tree-search tracker for following maneuvering targets.

Table 5.4: Tracking performance of the tree-search tracker with validation and that of the EKF-JPDAF, and PF.

| Clutter Density | Lost Targets Tree-Search Tracker | Lost Targets EKF-JPDAF | Lost Targets PF |
|---|---|---|---|
| 0.5 | 9 % | 18 % | 15 % |
| 1 | 16 % | 22 % | 20 % |
| 2 | 20 % | 32 % | 27 % |

Table 5.5: CPU time usage of the simulated stack-based tree search, EKF-JPDAF, and PF algorithms. The expected value (EV) and the standard deviation (Std) of the CPU time is calculated for each algorithm.

| Clutter Density | CPU Time Tree-Search | CPU Time EKF-JPDAF | CPU Time PF |
|---|---|---|---|
| 0.5 | EV: 107.9s Std: 71.5s | EV: 0.2s Std: 0.03s | EV: 44.8s Std: 0.3s |
| 1 | EV: 159.3s Std: 107.2s | EV: 0.75s Std: 1.5s | EV: 46.8s Std: 0.4s |
| 2 | EV: 264.9s Std: 142.8s | EV: 1.23s Std: 0.8s | EV: 49.2s Std: 0.4s |

In addition to performance comparison between the simulated algorithms, the expected value and the standard deviation of the CPU time usage is calculated for each tracker. The results are presented in Table 5.5. As we expect, the EKF-JPDAF is the fastest technique among the simulated algorithms. The CPU time usage of the PF is smaller than that of the

stack-based tree search algorithm, and it slightly increases in the environments with higher clutter density due to an increase in the frequency with which the re-sampling procedure must be performed. The CPU time of the stack-based tree search algorithm, however, more than doubles as clutter density increases from 0.5 to 2. One notable parameter in the CPU time usage of the stack-based tree search algorithm is its large standard deviation. This means that the CPU time usage notably varies for each realization of the simulated targets. In some realizations, positions of the clutter contacts and the observation noise may force the stack algorithm to expand more paths in the search tree. Although the stack-based tree search shows higher complexity in comparison with the PF and the EKF-JPDAF, it can still be implemented in real time for many sonar applications. The average CPU time for each estimation update of the target is equal to the average CPU time usage divided by the number of the scans. As an example in clutter density 2, it takes about 5.2 seconds for the stack algorithm to update the states of both targets. This time is acceptable for many sonar target tracking applications in which the time between the successive scans is often larger than one minute.

The CPU time usage does not represent an exact complexity comparisons, and its value depends on the implementation of the algorithms and the specifications of the simulating machine. Analytical complexity evaluation of the stack-based tracker algorithm is beyond the scope of this research and is considered as a future work.

## 5.6  Analysis of Path Metric Behavior

In some applications such as sonar target tracking, the mean square error (MSE) does not provide a complete evaluation of the performance of the tracker. In addition to estimation accuracy provided by the MSE, it is important to understand how often the tracker successfully follows the true target path. In fact, if the target is tracked, then the MSE becomes important and shows the accuracy of the estimation. In an environment with high clutter density, the tracker may follow clutter observations instead of the true target path. Therefore, it is very important to study the conditions in which the tracker loses the true

target path.

In this section, we seek an approach to evaluate the performance of the stack algorithm based on different system parameters. The stack algorithm is a tree search algorithm, and the estimation is performed by comparing the path metric of the existing paths of the search tree. Therefore, statistical analysis of the path metric provides a means to evaluate the performance of the stack algorithm. We first evaluate the probability distribution function of a general path metric. In Section 5.6.1, we analytically approximate the probability distribution function of a path metric under a no-clutter condition, perfect observability of the target, and the absence of discretization error. In Section 5.6.2, we extend the distribution approximations considering clutter contacts, discretization error, and non-unity target detection probability. Discussion of the results is provided in Section 5.6.3.

## 5.6.1   Statistical Analysis of the Path Metric in the Absence of Clutter

We seek to estimate the probability distribution function of a general path metric within the stack. We first start with the true target path and we ignore existence of clutter, discretization noise, and missed detections. The target dynamic state space model is assumed to be

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + G\mathbf{v}_k \tag{5.47}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{w}_k, \tag{5.48}$$

where $\mathbf{v}_k \sim \mathcal{N}(0, \Sigma_v)$ is the process noise vector with length of $L$ and $\mathbf{w}_k \sim \mathcal{N}(0, \Sigma_w)$ is the measurement noise vector with length of $D$. In the absence of clutter, there is no data association hypothesis, and the metric of the estimated path $\{\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_k\}$ using (5.6)

simplifies to

$$\bar{b}_k^{(i)} = \frac{1-\lambda}{1-\lambda^{k-1}} \tag{5.49}$$

$$\sum_{l=2}^{k} \lambda^{k-l} \left( \frac{1}{2} \left( G^+(\hat{\mathbf{x}}_l - F\hat{\mathbf{x}}_{l-1}) \right)^T \Sigma_v^{-1} \left( G^+(\hat{\mathbf{x}}_l - F\hat{\mathbf{x}}_{l-1}) \right) + \frac{1}{2} (\mathbf{z}_l - H\hat{\mathbf{x}}_l)^T \Sigma_w^{-1} (\mathbf{z}_l - H\hat{\mathbf{x}}_l) \right),$$

In deriving (5.49), we have multiplied the metric by $-1$ and eliminated the constant term $\frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \lambda^{k-l} \left[ \frac{L+D}{2} \log(2\pi) + \frac{1}{2}\log(|\Sigma_v|) + \frac{1}{2}\log(|\Sigma_w|) \right]$ since it only shifts the expected value of path metric and it is equal for all the paths. To define a non-target path, we define a difference state vector as the difference between the true target path and the estimated target path. We denote the true target path by $\{\mathbf{x}_1, \cdots \mathbf{x}_k\}$ and the state difference between the true target state and the estimated target state by

$$\mathrm{d}_i = \hat{\mathbf{x}}_i - \mathbf{x}_i. \tag{5.50}$$

We substitute (5.50) into (5.49) to yield

$$\bar{b}_k^{(i)} = \frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \lambda^{k-l} \left( \frac{1}{2} \left( G^+(\mathbf{x}_l + \mathrm{d}_l - F\mathbf{x}_{l-1} - F\mathrm{d}_{l-1}) \right)^T \Sigma_v^{-1} \right.$$

$$\left. \left( \mathbf{G}^+(\mathbf{x}_l + \mathrm{d}_l - F\mathbf{x}_{l-1} - F\mathrm{d}_{l-1}) \right) + \frac{1}{2}(\mathbf{z}_l - H\mathbf{x}_l - H\mathrm{d}_l)^T \Sigma_w^{-1} (\mathbf{z}_l - H\mathbf{x}_l - H\mathrm{d}_l) \right)$$

$$= \frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \lambda^{k-l} \left[ \frac{1}{2} \left( \mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l) \right)^T \Sigma_v^{-1} \left( \mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l) \right) \right.$$

$$\left. + \frac{1}{2} (\mathbf{w}_l - H\mathrm{d}_l)^T \Sigma_w^{-1} (\mathbf{w}_l - H\mathrm{d}_l) \right]. \tag{5.51}$$

Note that the target dynamic state space model (5.47), (5.48) has been applied to derive the metric based on random vectors $\mathbf{v}$ and $\mathbf{w}$. Defining the random variable $v_l$ as

$$v_l \;=\; \left(\mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)\right)^T \Sigma_v^{-1} \left(\mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)\right) \tag{5.52}$$
$$+ \left(\mathbf{w}_l - H\mathrm{d}_l\right)^T \Sigma_w^{-1} \left(\mathbf{w}_l - H\mathrm{d}_l\right),$$

the path metric is simplified to

$$\bar{b}_k^{(i)} = \frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \frac{1}{2} \lambda^{k-l} v_l. \tag{5.53}$$

The random variable $v_l$ is the sum of the quadrature of independent random vectors $\mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)$ and $(\mathbf{w}_l - H\mathrm{d}_l)$. Because of the Gaussian state space model assumption, the random vectors $\mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)$ and $(\mathbf{w}_l - H\mathrm{d}_l)$ are Gaussian with expected values of $G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)$ and $H\mathrm{d}_l$, and the covariance matrices of $\Sigma_v$ and $\Sigma_w$ respectively. Therefore, the random vector $v_l$ is a non-central Chi-square random variable with $D + L$ degrees of freedom and non-centrality parameter

$$\delta_l = \left(G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)\right)^T \Sigma_v^{-1} G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l) + (H\mathrm{d}_l)^T \Sigma_w^{-1}(H\mathrm{d}_l). \tag{5.54}$$

The metric defined by (5.53) is the weighted sum of the Chi-square distributed random variables $v_l$. The non-centrality parameter of the random variable $v_l$ depends on the vector $F\mathrm{d}_{l-1} - \mathrm{d}_l$, which can be viewed as the innovations element of the state difference vector. For the true target path, ignoring the discretization error, the state difference vector is zero, i.e. $\mathrm{d}_l = 0$ for $l = 1, \ldots, k$. We assume that the innovation state difference vector is independent across $l$ since it is affected by the state process noise vector $\mathbf{v}_l$, the observation noise vector $\mathbf{w}_l$, and clutter observations which are independent across $l$. Therefore, the random variables $v_l$ are independent across $l$.

The probability distribution function of the sum of infinite independent random variables has been studied well in advanced statistics [81]. Due to the exponential windowing, the sum in (5.53) is not an infinite sum, and the number of effective elements in the sum is governed by the window parameter $\lambda$. We first assume that the window length is infinite and we approximate the distribution of the path metric. Then, we discuss the effect of short window length on the distribution.

For a long window length ($\lambda \approx 1$), the sum in (5.53) is not limited by the window length and considering large $k$, we assume that the number of elements in the sum is infinite. It is shown in Appendix B that the non-identically distributed random variables $v_i$ satisfies the Lyapunov condition. Therefore, the Lyapunov central limit theorem can be applied and the path metric converges in distribution to Gaussian distribution with expected value and variance of

$$b_k^{(i)} \sim \mathcal{N}(\mu_{b_k}, \sigma_{b_k}^2)$$

$$\text{where} \quad , \quad \mu_{b_k} = \left[ \frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \lambda^{k-l} \left( -\frac{1}{2}(D + L + \delta_l) \right) \right]$$

$$\text{and} \quad , \quad \sigma_{b_k}^2 = \left[ \left( \frac{1-\lambda}{1-\lambda^{k-1}} \right)^2 \sum_{l=2}^{k} \left( \lambda^{k-l} \right)^2 \frac{1}{2} \left( (2D + 2L + 4\delta_l) \right) \right]. \qquad (5.55)$$

For a short window length, we can not apply the Lindeberg-Feller or Lyapunov central limit theorems since Lindeberg's condition [81] is not satisfied. We know from the statistics that if a sequence of independent random variables is not uniformly asymptotically negligible (u.a.n.), then Lindeberg's condition is not satisfied. Denoting the variance of $v_i$ by $\sigma_i^2$, we

find

$$\lim_{k \to \infty} \frac{\text{maximum variance of the contributed elements}}{\text{variance of the total sum}} = \frac{\sigma_v^2}{\sum_{l=1}^{k} \lambda^{2(k-l)} \sigma_v^2}$$

$$= \lim_{k \to \infty} \frac{1}{\sum_{l=1}^{k} \lambda^{2(k-l)}}$$

$$= 1 - \lambda^2 \neq 0,$$

and hence the sequence of variables $v_i$ is not u.a.n. We see that when $\lambda \approx 1$, then the path metric converges in distribution to Gaussian distribution. For the general case, the convergence depends on the value of $\lambda$. For $\lambda \approx 0$, we only have one element in the summation, and the distribution function of the path metric is equal to the distribution of $v_k$. In target tracking applications, however, the value of $\lambda$ is usually selected between 0.8 and 1, which suggests that the Gaussian approximation may be appropriate. We use the Kolmogrov-Smirnov (KS) statistical test [92] to compare the approximated distribution from (5.55) with that of the empirically generated distribution by Monte-Carlo simulation. To generate an empirical distribution and compare it with a Gaussian distribution, we simulate the tracking problem using the specific parameters that are defined as follows. The system state vector is defined as $\mathbf{x}_k = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}$ where $x$ and $y$ denote the target location in Cartesian coordinates and $\dot{x}$ and $\dot{y}$ denote the target speed in $x$ and $y$ dimension. The state space model parameters, $F$ and $G$ are fixed as in (5.25) and the observation transition matrix $H$ is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{5.56}$$

to meet a linear-Gaussian model. The covariance matrix of the state process noise and the measurement noise are given by $\Sigma_v = 0.0025 I_2$ and $\Sigma_w = 0.04 I_2$. The results of the Monte-Carlo simulation are shown in Figure 5.26 for the true target path and two different sets of window parameter $\lambda$ and data length $k$. We first set the values for $k$ and $\lambda$ as $k = 50$ and
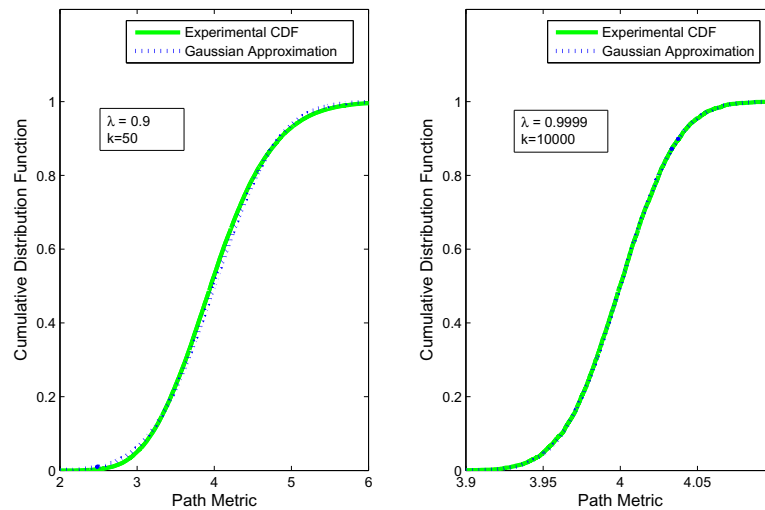
Figure 5.26: Comparison of the empirical cdf of the target path metric ($F_{D1}$) with the cdf of Gaussian distribution ($F_{D0}$) for different set of $k$ and $\lambda$. For large $k$ and $\lambda \approx 1$, the KS test results in $\max |F_{D1} - F_{D0}| = 0.005$. For $k = 50$ and $\lambda = 0.9$, the KS test results in $\max |F_{D1} - F_{D0}| = 0.017$, which still shows acceptable approximation.
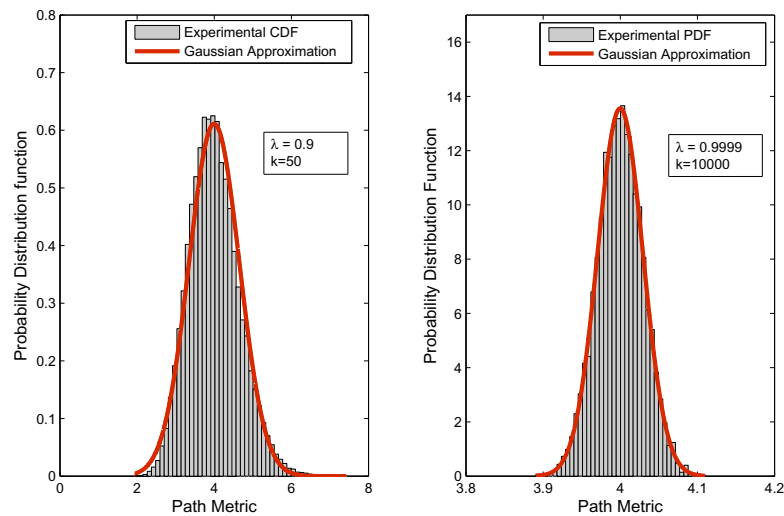


Figure 5.27: Comparison of the empirical pdf of the target path metric with the Gaussian pdf for different value of $k$ and $\lambda$. The Gaussian approximation is acceptable for practical values of $k$ and $\lambda$, and it converges well as $k$ increases and $\lambda \approx 1$.

$\lambda = 0.9$. We see that the empirical cumulative distribution function (cdf) is well matched to the analytical Gaussian cdf with parameters calculated from (5.53). The maximum absolute difference between the two cdfs is 0.017, which shows that the approximated cdf fits the data with precision higher than 98%. In the second set of parameters, we seek to test the convergence to the Gaussian probability distribution by increasing the window size and the data length. We set $k = 10000$ and $\lambda = 0.9999$, and the maximum absolute difference between the two cdfs decreases to 0.005. This shows that the distribution of the sum becomes closer to a Gaussian distribution.

Figure 5.27 shows the comparison of the empirical pdf and the Gaussian pdf. The expected value and the covariance of the Gaussian pdf is derived using (5.55). We can see that the empirical pdf of the path metric matches well with the Gaussian pdf also for small $k$ and $\lambda$.

In Figure 5.28, we fix the value of $k$ at 50, and we vary the value of $\lambda$ to see the effect of the window length on the distribution approximation. If we set the accuracy threshold to 96%, we see that by selecting $\lambda > 0.5$, the distribution can be well approximated by (5.53).

In the following section, we extend the results of this section to calculate the probability distribution function of the path metric considering clutter, discretization error, and missed target detections ($P_D \neq 1$).
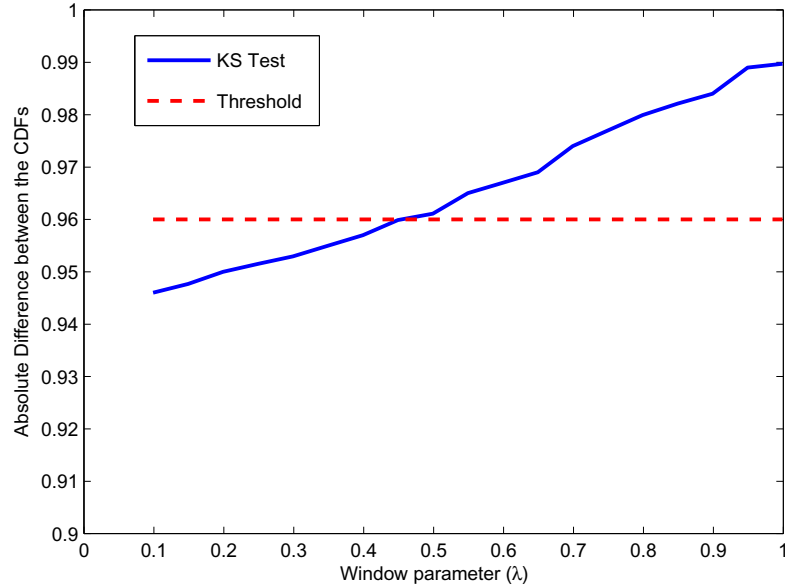
Figure 5.28: KS test results for different values of windowing parameter. We can see that the Gaussian approximation matches well for $\lambda > 0.5$, which is always met in our simulated stack-based tracker.

### 5.6.2 General statistical analysis of the path metric

In this section, we empirically approximate the probability density function of the path metric considering data association uncertainty, discretization error, and non-perfect observability of the target. Using (5.6), the path metric can be written as

$$
\begin{aligned}
b_k^{(i)} &= \frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \lambda^{k-l} \left[ -\frac{1}{2} \left( G^+(\mathbf{x}_l^{(i)} - F\mathbf{x}_{l-1}^{(i)}) \right)^T \Sigma_v^{-1} \left( G^+(\mathbf{x}_l^{(i)} - F\mathbf{x}_{l-1}^{(i)}) \right) \right. \\
&\qquad \left. + \log \left( \omega_0 + \sum_{j=1}^{m_k} \omega_j \frac{1}{(2\pi)^{L/2}|\Sigma_w|^{1/2}} \exp(-\frac{1}{2}\eta_l^j) \right) \right] \\
&= \frac{1-\lambda}{1-\lambda^{k-1}} \sum_{l=2}^{k} \lambda^{k-l} \left[ -\frac{1}{2}\epsilon_l + \log \left( \omega_0 + \sum_{j=1}^{m_l} \omega_j \frac{1}{(2\pi)^{L/2}|\Sigma_w|^{1/2}} \exp(-\frac{1}{2}\eta_l^j) \right) \right],
\end{aligned}
$$

(5.57)

where $\omega_j$ denotes the probability of the $j$th data association event as defined in Section 5.2.4 and $\sum_j \omega_j = 1$. The gating technique is imposed on the measurement dataset to filter out unlikely observations, and $m_l$ denotes the number of observations that lie inside the gate. Similar to the no-clutter case, a constant term is eliminated from the path metric for simplicity. The random variables $\epsilon_l$ and $\eta_l^j$ in (5.57) are defined as

$$\epsilon_l = \left(\mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)\right)^T \Sigma_v^{-1} \left(\mathbf{v}_l - G^+(F\mathrm{d}_{l-1} - \mathrm{d}_l)\right),$$

$$\eta_l^j = \left(\mathbf{w}_l - H\left(\mathbf{x}_l - \mathrm{d}_l\right)\right)^T \Sigma_w^{-1} \left(\mathbf{w}_l - H\left(\mathbf{x}_l - \mathrm{d}_l\right)\right).$$

For the true target path in the search tree, when discretization error is present, the state difference vector $\mathrm{d}_l$ is uniformly distributed in the discretized cell to represent the discretization error. For a non-target path, the state difference vector is the summation of uniformly distributed discretization error and the state estimation error, which is the difference between the true target state and the estimated target state.

The metric in (5.57) is the combination of multiple independent random variables $\epsilon_l$, $\eta_l^j$, and dependent random variables $\omega_j$ and $m_l$. While the random variables $\epsilon_l$, and $\eta_l^j$ are Chi-square distributed, the analytical derivation of the path metric distribution is challenging since the Chi-square random variables $\eta_l^j$ are randomly weighted by data association probabilities $\omega_j$, and the number of observations $m_l$ is Poisson distributed. However, the generalized central limit theorem suggests that the path metric may converge in distribution to a Gaussian random variable. Mathematical proof of the distribution convergence is only possible for long window length ($\lambda \approx 1$) and will be challenging due to the combination of multiple independent random variables in (5.57). Similar to the no-clutter case analyzed in Section 5.6.1, we approximate the probability distribution function of the path metric with a Gaussian distribution and use the KS test to verify the approximation.

We use the same parameters as in Section 5.6.1 to generate the empirical distribution of the path metric, but we set the clutter density to $\zeta = 1$ and the target detection probability
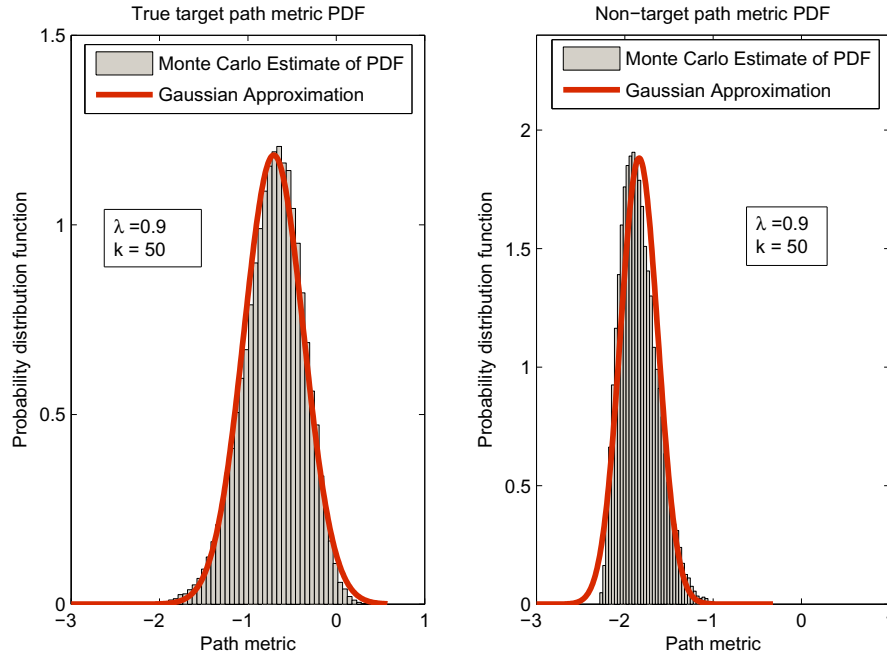
Figure 5.29: Approximation of the empirical path metric pdf by a Gaussian distribution for a true target path and a non-target path. The KS test for the non-target path results in max $|F_{D1} - F_{D0}| = 0.043$ and for a true target path results in max $|F_{D1} - F_{D0}| = 0.02$. It can be seen that the centers of the pdf functions are well separated, but the distributions do exhibit overlap in their tails.

to $P_D = 0.9$ to see the effect of clutter and missed target detections. We empirically generate the path metric of the true target path and of a non-target path. We consider a non-target path as a path that does not receive any true target observations in its validation gate, and hence its metric is a function of clutter-generated observations. The probability distribution functions of the path metrics for target and non-target paths are depicted in Figure 5.29. The KS test shows that the maximum difference between the empirical cdf and the Gaussian approximation cdf is 0.02 for the true target path and 0.043 for a non-target path. This shows that the accuracy of Gaussian approximation is about 96% for the a non-target path and about 98% for the true target path.

Due to non-linear transformations of multiple random variables in (5.57), the analytical calculation of the expected value and the variance of the approximated distributions is a

challenge; hence, we suggest using Monte-Carlo simulations to approximate the expected value and variance of the distribution. We use the Monte Carlo approximated expected value and variance of the Gaussian distribution to generate the Gaussian pdf in Figure 5.29. In the following section, we implement the results of the distribution approximation to analyze the behavior of the path metric as a function of discretization error, observation noise, and clutter density.

### 5.6.3  Characterizing Path Metric Sensitivity

In this section, we use the results of Section 5.6.2 to study how the behavior of the path metric (and hence the stack algorithm) varies with model and environmental parameters. For performance evaluation, we are interested in understanding the conditions under which the tracker fails in tracking a target. Using the stack algorithm, the estimated target path has the largest metric among all the paths explored within the search tree. To evaluate the probability of losing the target path, we need to consider the conditions in which the path metric of a non-target path is larger than that of the true target path. The stack algorithm does not explore all possible paths in the search tree, and hence one must also consider whether or not a particular path in the search tree has been explored. The number of paths that are explored in the search tree depends on many system parameters, such as clutter density, target observability at the receiver, observation noise covariance, initial estimation of the target state, and the target trajectory. Therefore, evaluating the probability of losing the target is a challenging problem, and the proposed adaptive discretization technique adds an additional element to this challenge.

Considering the challenge of performance evaluation in the stack algorithm, we seek an approach to simplify the problem. We first assume that all paths are explored in the search tree. This assumption simplifies the problem since we are not required to calculate the probability that a specific path has been explored in the search tree. Then instead of calculating the probability of losing a target, which is dependent upon the particular target trajectory, we consider a simplified criteria, denoted by $p_l$: the probability that the path

metric of the true target path is smaller than that of the non-target path with the largest expected path metric. Although $p_l$ is not equivalent to the probability of losing a target, it helps us to study the conditions under which the tree-search tracker can not discriminate between the true target path and the non-target paths within the search tree.

For calculating $p_l$, we denote the true target path metric by $b^t$ and the path metric of the non-target path with largest expected path metric by $b^l$. Assuming that $b^t$ is independent of $b^l$, $p_l$ is given by

$$
\begin{aligned}
p_l &= P(b^l > b^t) \\
&= \int_{-\infty}^{\infty} p_{b^t}(\tau) \int_{\tau}^{\infty} p_{b^l}(\nu) d\nu \ d\tau,
\end{aligned}
\tag{5.58}
$$

where $p_{b^t}(b)$, and $p_{b^l}(b)$ are probability distribution functions of $b^t$ and $b^l$, respectively. The non-target path with largest expected path metric is identified from (5.57). In (5.57), the effect of random variable $\eta_l^j$ will be negligible for non-target paths, since a non-target path sees only the clutter observations, which are assumed to be uniformly distributed in the surveillance region. Therefore, the path with $\epsilon_l = 0$ has the largest expected path metric value among all non-target paths. For a linear-Gaussian model defined in (5.47), (5.48), this path is the constant velocity target path. Figure 5.30 shows an example of the constant velocity path and the true target path. Suppose that the target makes a slight turn to the right in its trajectory. In this case, the constant velocity path (straight path) becomes a non-target path and it has the largest expected path metric among the non-target paths of the search tree.

Characterizing path metric sensitivity, we consider the same system parameters as in Section 5.6.2 and calculate the value of $p_l$ across a range of values of observation noise, clutter density, and target detection probability. Figure 5.31 shows variations of $p_l$ as a function of clutter density. The clutter density varies from low ($\zeta = 0.5$) to very high ($\zeta = 3.5$). The observation noise covariance matrix is set to $\Sigma_w = 0.004 \times I_2$, and $p_l$
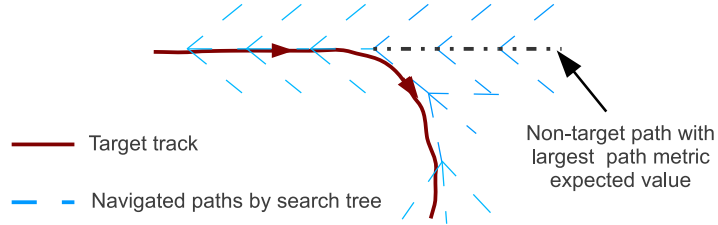
Figure 5.30: An example of true target path and the constant velocity path that has the path metric with largest expected value among non-target paths. When the target turns, the expected value of its path metric decreases. This increases the chance of selecting the constant velocity path.

is calculated for different target detection probabilities $P_D$. We see that in low clutter density, the probability that the stack-based tracker selects the true target path is close to unity, since $p_l$ is close to zero. The value of $p_l$ increases with increasing the clutter density and decreasing the probability of target detection. In this case, the probability that the stack based tracker selects a non-target path increases. As an example, with $P_D = 0.7$ and $\zeta = 2.5$, the constant velocity path has a larger metric than the target path with probability 0.1; when the probability of detection is increased to $P_D = 0.9$, the probability of selecting the constant velocity path is about 0.0005 .

In Section 5.2.5, we develop a sampling based discretization scheme to reduce discretization error of the proposed stack-based tracker. In designing a discretization technique for the stack-based tracker, we need to consider the effects of the discretization error on the path metrics. Figure 5.32 shows the effect of discretization error on variations of $p_l$, again as a function of clutter density. The probability of target detection is set to $P_D = 0.85$. In low observation noise ($\Sigma_w = 0.004 \times I_2$), values of $p_l$ for the sampling discretization technique are smaller than that of the fixed discretization technique and the ability of the stack-based tracker in identifying the true target path improves when using the sampling discretization. In higher observation noise ($\Sigma_w = 0.02 \times I_2$), however, the observation noise dominates the discretization error and the ability of the stack-based tracker in identifying the true target path is more affected by the observation noise.
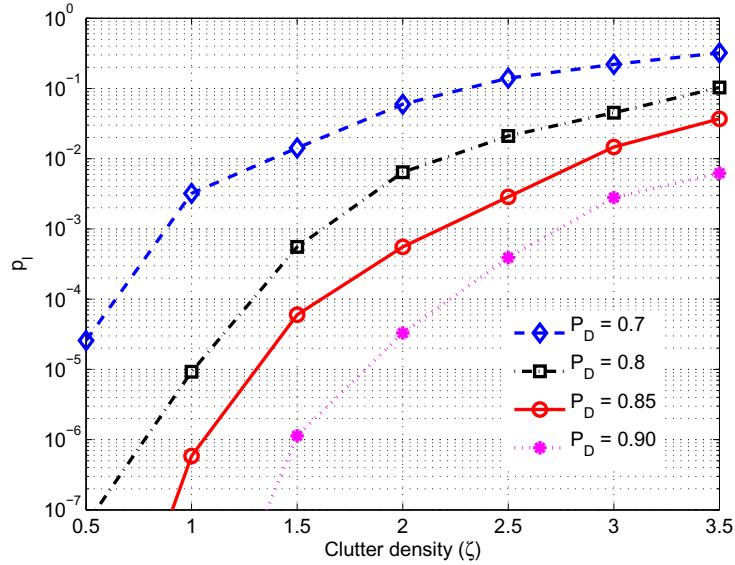
Figure 5.31: Variations of $p_l$ as a function of clutter density for different target detection probabilities. The observation noise covariance matrix is set to $\Sigma_w = 0.004 \times I_2$.

Figure 5.33 shows the effect of observation noise on the path metric sensitivity as a function of clutter density. The probability of target detection is set to $P_D = 0.85$. Three scenarios have been considered for the observation noise. In the first scenario, the observation noise covariance matrix is set to $\Sigma_w = 0.004 \times I_2$. In this case, the stack algorithm is able to detect the true target path with high probability. In the second scenario, the observation noise covariance matrix is set to $\Sigma_w = 0.02 \times I_2$ and in the third scenario, the observation noise is set to $\Sigma_w = 0.04 \times I_2$. We see that high observation noise reduces contributions of the true target observations in the true target path metric, and hence the probability that the stack-based tracker identifies the target path decreases.

Using path metric sensitivity analysis, we calculated the probability that the stack algorithm fails to track the true target path and instead follows the constant velocity path. In our evaluation, we assumed that the constant velocity path is present in the search tree and we didn't consider applying the track validation technique discussed in Section 5.5.4. From the point of view of the track validation technique, the constant velocity path doesn't have any advantage in comparison to other non-target paths, since the tracks are validated

using the likelihood function of the observations. Therefore, applying the track validation technique improves the ability of the stack based tree search algorithm to maintain the true target path specially in high clutter density and high measurement noise.
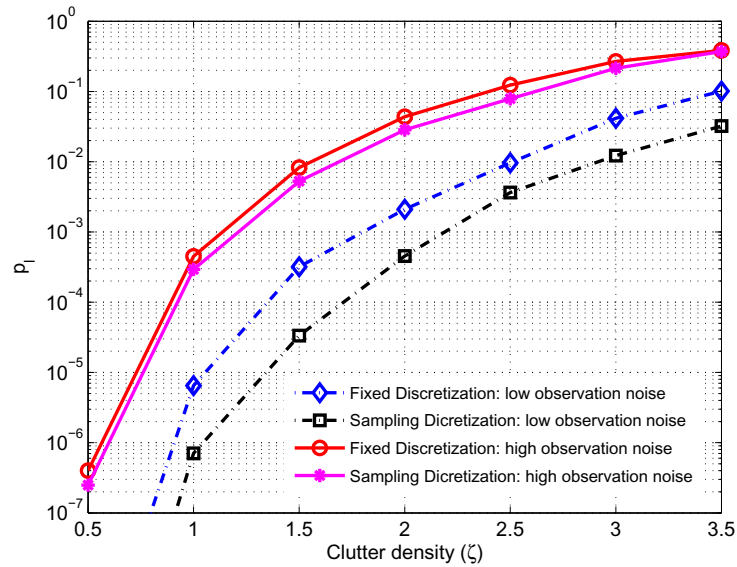
Figure 5.32: Effect of discretization error on values of $p_l$ as a function of clutter density. While the sampling discretization technique has much smaller values of $p_l$ in low observation noise ($\Sigma_w = 0.004 \times I_2$), in higher observation noise ($\Sigma_w = 0.02 \times I_2$), $p_l$ is more affected by the observation noise than the discretization error.
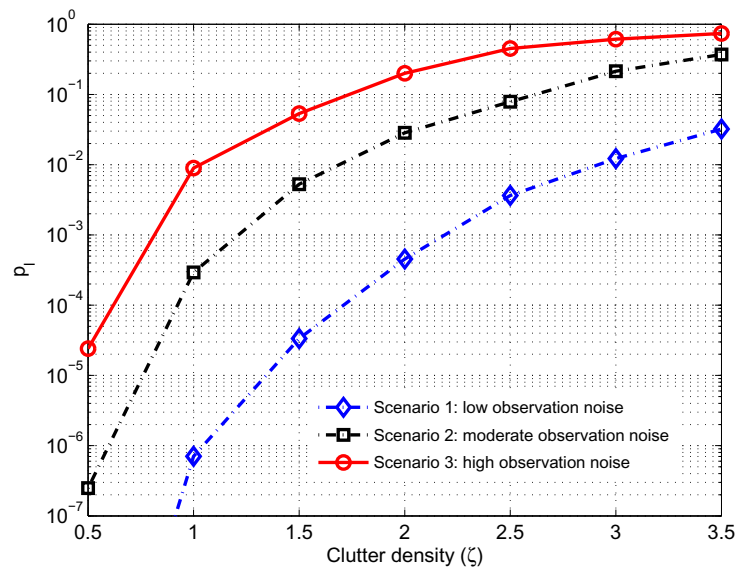


Figure 5.33: Effect of observation noise on values of $p_l$ as a function of clutter density. The observation noise is set as follows: scenario 1: $\Sigma_w = 0.004 \times I_2$, scenario 2: $\Sigma_w = 0.02 \times I_2$, and scenario 3: $\Sigma_w = 0.04 \times I_2$. Probability of target detection is set to $P_D = 0.85$.

# Chapter 6: Conclusions

We have developed tree search techniques for approximating the MAP solution to general detection and estimation problems that fit within a dynamic state space model. The proposed techniques use noisy observations to sequentially estimate the system state in a discrete or discretized continuous state space. We showed that the applications of the tree-search algorithms are not limited to linear-Gaussian state space models and the proposed techniques perform well in the presence of severe nonlinear dynamic state space models.

We applied the proposed techniques in wireless communication applications including blind channel equalization and modulation classification. In blind channel equalization, we considered the joint channel equalization and decoding problem. Using a forgetting factor matched with the variation rate of the channel, we successfully applied the algorithm for time-varying channels. The algorithm was compared with LMS-Viterbi algorithm and the combined particle filtering equalizer-Viterbi decoder algorithm. We addressed the challenges of implicitly estimating a dispersive fading channel whose effective order may also vary with time. Simulation results show that the proposed detector can achieve lower BER than competing blind detection schemes, even when only the proposed algorithm faces uncertainty in the channel order.

For modulation classification applications, we applied the tree search technique for joint channel estimation and data detection considering all possible modulation schemes of the transmitted sequence. We implemented the M-algorithm for tree search to reduce the complexity of the search tree. The statistical analysis of the estimation error was successfully used for classifying the modulation scheme of an unknown signal. The proposed algorithm is compared with Gibbs sampling technique, and the simulation results reveal superior performance of the proposed algorithm in classification of high density modulation schemes using a low number of observation sequences.

We also applied stack-based tree search techniques to the challenge of multiple target tracking. We presented a novel approach to target tracking in which each possible sequence of target states is mapped to a path through a tree, and the stack algorithm for depth-first tree search is employed to identify the most likely sequence of the states. We then proposed a sampling technique to identify points in each of the discretized cells for state space discretization. The proposed sampling technique reduces algorithm complexity by allowing for coarser state space discretization and thereby generating a smaller search tree. While the sampling based approach does require evaluation of the posterior density at multiple points in each discretized region, the net effect of reduced tree size results in a significant improvement in algorithm speed. Performance results reveal that the sampling stack tracker performs better than particle filtering technique and significantly outperforms the EKF-PDA in the presence of clutter and possibility of missed target.

An LLR-based track validation technique embedded in the tree-search based tracker has been developed to terminate false tracks and lost targets within a few scans. Using the LLR-based track management technique, we extended the stack-based tracker to address multistatic sonar tracking and multiple target tracking. The performance of the algorithm has been evaluated on data obtained from the SEABAR'07 experiment. Performance results show that the stack-based tracker is able to follow the target trajectory closely and to maintain track through highly nonlinear maneuvers. We evaluated the ability of the proposed tree search algorithm in discriminating the true target path with other non-target paths by approximating the probability distribution function of the path metric. We studied variations of different system parameters such as probability of missed-target, clutter density, and variance of the observation noise on the ability of the stack to maintain the true target path. The results of this analysis are important in design and implementation of the proposed technique in different applications.

Implementation of the stack algorithm for applications with a large number of unknown parameters, such as modulation classification, must be performed with caution due to possible local maximums in the calculated path metric. Since the stack algorithm partially

explores the search tree, it may follow a path through a local maximum of the posterior probability mass function, and hence the true solution may not be expanded by the search tree. In these applications, the M-algorithm (or a similar algorithm) that manually explores more paths is preferable, since the probability that the true solution path is explored by the search tree increases.

## 6.1   Future Work

Based on the empirical and simulation results of the proposed tree-search techniques for detection and estimation problems, the algorithms show promise for further development and evolution. In wireless communication systems, we assumed that system parameters such as variance of the additive noise are known. Extension of the proposed algorithms for estimating the measurement noise variance and applying the estimate in the path metric of the search tree is a topic for future work. Additionally, we proposed a simple approach for dealing with unknown channel length, but the approach does not explicitly estimate channel length. Further research in this area could include directly estimating the channel length and applying the results in the path metric. This technique may reduce the performance loss due to channel length uncertainty.

The proposed approach in classification of different QAM modulation schemes can be extended to modern communication systems which employ OFDM (Orthogonal Frequency Division Multiplexing) for transmission. Applying the proposed modulation classifier to each sub-carrier of an OFDM modulated signal may be cumbersome, and development of faster tree search techniques is required. Additionally, the proposed blind channel equalization and modulation classification techniques can be applied in communication systems with multiple receivers. A multiple receiver communication system is interpreted as either a single receiver with multiple receive antennas or multiple receivers located in separate locations with the ability to communicate with each other. For these applications, the path metric must be modified in a way to incorporate the measurements across all receivers.

The proposed stack-based target tracking algorithm has the potential to be developed

in different scenarios. The proposed algorithm assumes that the number of existing targets and the initial location of each target are known. In reality, a track initiation algorithm must be applied to initialize the possible target tracks in the region. We used a simple approach for track initiation of SEABAR'07 dataset, and we proposed a track management technique for identifying false tracks and validating the target tracks. Future extension of the proposed target tracking technique could include using tree search techniques for track initiation. In this scenario, a tree search technique can be applied to locate the high peaks of the likelihood ratio function for initiating possible target tracks.

In developing a stack-based tree search technique, we assumed that the target state space model is provided for the tracker. In a real target tracking system, this information may not be provided or the system parameters may change during target tracking. Extension of the proposed algorithm in applications with uncertain target state space models is interesting in sonar and radar target tracking, since the target may maneuver in many of these applications. The extension can be performed in either developing the path metric using possible different dynamic state space models for a target or developing a track management technique to monitor the target trajectories.

In all applications we considered, the complexity of the full tree search algorithm is reduced by implementing partial tree search algorithms such as the stack algorithm or the M-algorithm. While the computational complexity of the M-algorithm is deterministic, for the stack algorithm complexity varies with observation noise and other system parameters. However, the computational complexity of the stack algorithm with the same size path memory is lower than or equal to that of the M-algorithm. Statistical complexity analysis of the stack-based target tracking algorithm is considered as an extension of this research. The results of the computational analysis would be useful in better design and development of the stack algorithm for specific detection and estimation applications.

# Appendix A: SEABAR'07 Dataset Observation Model

The SEABAR07 experiment took place in the Mediterranean Sea in October 2007 [84]. The experiments included one transmitter and three receivers. In the SEABAR'07 dataset, the observations are provided in the form of the time difference of arrival (TDOA), angle of arrival (bearing), and range rate (CW only). This appendix describes the measurement fields that are provided in SEABAR'07 dataset and describes the relationship between the system state vector and the observation vector. The geometrical derivations are provided by Coraluppi [87] for a multistatic sonar system.

The SEABAR'07 dataset is available in separate folders for FM signals and for CW signals. The following fields are available for each contact:

- `time`: shows the received time of the contact

- `source_position`: location of the transmitter in attitude and longitude format.

- `receiver_position`: location of the receiver in attitude and longitude format

- `source_velocity`: the velocity of the transmitter

- `receiver_velocity`: the velocity of the receiver

- `sound_speed`: speed of the sound in water, equal to 1500 m/s for all contacts.

- `type`: shows the type of the measured informations. The first element of the measurement vector is the time difference between the transmitted signal at the transmitter and the received signal at the receiver. The second element is the bearing of the received signal. For CW signals, there is an additional range rate measurement that shows the speed of the repeater.

- **measurement**: measurement matrix.

- **covariance**: covariance error of each measured vector in measurement field.

- **truth**: 0 if the measurement is related to clutter and non-zero for measurements from the target; this value is provided intentionally to evaluate the performance of the tracker and cannot be used as a measurement in simulations.

- **contact_id**: shows the contact identification code. Each contact has its own contact identification code that is unique within a scan.

- **snr**: shows the SNR of the received contact.

We consider a single source and $R$ receivers present in the region of interest. We assume that the source and receiver locations are known (at least approximately) to the trackers, and we denote them in a Cartesian coordinate system by $(x_s, y_s)$ and $(x_r, y_r)$ respectively. Let the vector $\mathbf{x}$ denote the state of the target, including the target location $(x, y)$ and the target speed $(\dot{x}, \dot{y})$. The measurements in the form of Bearing, time difference of arrival (TDOA), and range-rate are related to the target state by

$$
\begin{bmatrix}
\text{Bearing} \\
\text{TDOA} \\
\text{Range Rate}
\end{bmatrix}
=
\tag{A.1}
$$

$$
\begin{bmatrix}
\tan^{-1}\left(\frac{x-x_r}{y-y_r}\right) \\
\frac{1}{C}\left(\sqrt{(x-x_s)^2 + (y-y_s)^2} + \sqrt{(x_r-x)^2 + (y_r-y)^2} - \sqrt{(x_r-x_s)^2 + (y_r-y_s)^2}\right) \\
\frac{\dot{x}(x-x_s)+\dot{y}(y-y_s)}{2\sqrt{(x-x_s)^2+(y-y_s)^2}} + \frac{\dot{x}(x-x_r)+\dot{y}(y-y_r)}{2\sqrt{(x-x_r)^2+(y-y_r)^2}}
\end{bmatrix},
$$

where $C$ denotes the sound speed in the water.

# Appendix B: Distribution of the Sum of Non-central Chi-square Distributed Random Variables

In probability theory, the distribution of the sum of independent random variables has been studied through the central limit theorem. While the conventional central limit theorem requires independent random variables to be identically distributed, the generalized central limit theorems consider independent random variables that are not identically distributed. The Lindeberg-Feller and Lyapunov central limit theorems consider the conditions in which the sum of independent random variables converges in distribution to a Normal (Gaussian) distribution. In this appendix, we check the Lyapunov conditions for sum of independent non-centralized chi-square distributed random variables, and we prove that the distribution converges to a Normal distribution. First we recall the Lyapunov central limit theorem.

**Lyapunov Central Limit Theorem:**  Lets $x_i$ denote the set of independent random variables with finite expected value $\mu_i$ and variance $\sigma_i^2$. Define $S_n^2 = \sum_{i=1}^{n} \sigma_i^2$. The sum of $(\mathbf{x}_i - \mu_i)/\sigma_i$ converges in distribution to a standard Normal random variable if the Lyapunov condition

$$\lim_{n \to \infty} \frac{1}{S_n^{2+\delta}} \sum_{i=1}^{n} E\left[|x_i - \mu_i|^{2+\delta}\right] = 0 \tag{B.1}$$

is satisfied for some $\delta > 0$.

If $x_i$ denotes a non-central chi-square random variable with $k_i$ degree of freedom and non-centrality parameter of $\lambda_i$, then its second central moment (variance) is $2(k_i + 2\lambda_i)$ and

its third central moment is $8(k_i + 3\lambda_i)$. Setting $\delta = 1$ in (B.1), we have

$$
\begin{aligned}
\lim_{n\to\infty} \frac{1}{S_n^3} \sum_{i=1}^{n} E\left[|x_i - \mu_i|^3\right] &= \lim_{n\to\infty} \frac{\sum_{i=1}^{n} 8(k_i + 3\lambda_i)}{(\sum_{i=1}^{n} 2(k_i + 2\lambda_i))^{3/2}} \\
&< \lim_{n\to\infty} \frac{8n \max(k_i + 3\lambda_i)}{(2n \min(k_i + 2\lambda_i))^{3/2}} \\
&= \lim_{n\to\infty} M \frac{1}{\sqrt{n}} \quad \text{where } M < \infty \\
&= 0.
\end{aligned}
\tag{B.2}
$$

We can see that the sum of independent non-central distributed random variables satisfies the Lyapunov condition if $M \propto \frac{\max(k_i + 3\lambda_i)}{(\min(k_i + 2\lambda_i))^{3/2}}$ is finite. Therefore, the sum converges in distribution to a Normal distribution.

# Bibliography

138

# Bibliography

[1] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation.* Prentice Hall, 2000.

[2] H. L. V. Trees, *Detection, Estimation, and Modulation Theory: Part I.* John Wiley and Sons, 2001.

[3] L. Stone, T. Corwin, and C. Barlow, *Bayesian Multiple Target Tracking.* Artech House, 1999.

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME J. Basic Eng.*, vol. 82, pp. 34–45, March 1960.

[5] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Dover Publications, 1979.

[6] M. Norgaard, N. K. Poulsen, and O. Ravn, "New developments in state estimation of nonlinear systems," *Automatica*, vol. 36, pp. 1627–1638, November 2000.

[7] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, pp. 401–422, March 2004.

[8] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. on Automatic Control*, vol. 45, pp. 477–482, March 2000.

[9] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Trans. on Automatic Control*, vol. 45, no. 5, pp. 910–927, May 2000.

[10] I. Arasaratnam and S. Haykin, "Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 953–977, May 2007.

[11] ——, "Cubature Kalman filters," *IEEE Trans. on Automatic Control*, vol. 54, pp. 1254–1269, June 2009.

[12] D. L. Alspach and H. W. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Trans. on Automatic Control*, vol. 17, no. 4, pp. 439–448, August 1972.

[13] H. W. Sorenson and D. L. Alspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, vol. 7, pp. 465–479, July 1971.

[14] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, pp. 19–38, September 2003.

[15] E. Punskaya, *Sequential Monte Carlo Methods for Digital Communications*. PhD Thesis, University of Cambridge, 2003.

[16] C. J. Bordin and L. A. Baccala, "Deterministic particle filters for joint blind equalization and decoding on frequency selective channels," in *Proceedings of IEEE Statistical Signal Processing Workshop*, July 2005, pp. 375–380.

[17] R. Bucy and K. Senne, "Digital synthesis of nonlinear filters," *Automatica*, vol. 7, pp. 287–298, May 1971.

[18] M. Simandl, J. Kralovec, and T. Soderstrom, "Advanced point-mass method for nonlinear state estimation," *Automatica*, vol. 42, pp. 1133–1145, July 2006.

[19] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM Journal of Research and Development*, vol. 13, no. 6, pp. 675–685, November 1969.

[20] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. IEEE Press, 1999.

[21] R. E. Blahut, *Algebraic Codes for Data Transmission*. New York, NY: Cambridge University Press, 2003.

[22] I. M. Jacobs and E. R. Berlekamp, "A lower bound to the distribution of computation for sequential decoding," *IEEE Transactions on Information Theory*, vol. IT-13, no. 2, pp. 167–174, April 1967.

[23] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[24] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Pearson Education, Inc., 1985.

[25] Y. Sato, "A method of self-recovering equalization for multilevel amplitude-modulation systems," *IEEE Transactions on Communications*, vol. 23, no. 6, pp. 679–682, June 1975.

[26] F. C. Zheng, S. McLaughlin, and B. Mulgrew, "Blind equalization of nonminimum phase channels: higher order cumulant based algorithm," *IEEE Transactions on Signal Processing*, vol. 41, pp. 681–691, Feb 1993.

[27] J. K. Tugnait, "Blind equalization and estimation of digital communication for channels using cumulant matching," *IEEE Transactions on Communications*, vol. 43, pp. 1240–1245, Feb-Mar-Apr 1995.

[28] L. Tong, G. Xu, and T. Kailath, "Blind identification and equalization based on second-order statistics: A time domain approach," *IEEE Trans. on Information Theory*, vol. 40, pp. 340–349, March 1994.

[29] F. Alberge, M. Nikolova, and P. Duhamel, "Blind identification/equalization using deterministic maximum likelihood and a partial prior on the input," *IEEE Trans. on Signal Processing*, vol. 54, pp. 724–737, February 2006.

[30] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Transactions on Information Theory*, vol. 18, pp. 363–378, May 1972.

[31] N. Seshadri, "Joint data and channel estimation using blind trellis search techniques," *IEEE Trans. on Comm.*, vol. 42, pp. 1000–1011, February-April 1994.

[32] H. Kubo, K. Murakami, and T. Fujino, "Adaptive maximum-likelihood sequence estimation by means of combined equalization and decoding in fading environments," *IEEE Trans. on Selected Areea in Communications*, vol. 13, pp. 102–109, January 1995.

[33] X. Li, "Blind sequence detection without channel estimation," *IEEE Trans. on Signal Processing*, vol. 50, pp. 1735–1746, July 2002.

[34] Q. Dai and E. Shwedyk, "Detection of bandlimited signals over frequency selective rayleigh fading channels," *IEEE Trans. on Communications*, vol. 42, pp. 941–950, April 1994.

[35] C. J. Bordin and L. A. Baccala, "Particle filter algorithms for joint blind equalization/decoding of convolutionally coded signals," in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, March 2005, pp. iii/497–iii/500.

[36] J. K. Nelson and A. C. Singer, "Bayesian ML sequence detection for ISI channels," in *Proc. of the IEEE Conf. on Info. Sciences and Systems*, March 2006, pp. 693–698.

[37] R. H. Clarke, *A Statistical Theory of Mobile-Radio Reception.* Bell Syst. Tech. J., 1968.

[38] E. Eleftheriou and D. Falconer, "Tracking properties and steady-state performance of RLS adaptive filter algorithms," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 34, pp. 1097–1110, October 1986.

[39] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. on Automatic Control*, vol. AC-19, no. 6, pp. 716–723, 1974.

[40] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[41] Y. R. Zheng and C. Xiao, "Simulation models with correct statistical properties for Rayleigh fading channels," *IEEE Trans. On Signal Processing*, vol. 51, pp. 920–928, June 2003.

[42] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: Classical approaches and new trends," *IET Communications*, vol. 1, pp. 137–156, April 2007.

[43] B. Ramkumar, "Automatic modulation classification for cognitive radios using cyclic feature detection," *IEEE Circuits and systems magazine*, vol. 9, pp. 27–45, 2009.

[44] A. Polydoros and K. Kim, "On the detection and classification of quadrature digital modulations in broad-band noise," *IEEE Transactions on Communications*, vol. 38, pp. 1199–1211, August 1990.

[45] N. Benvenuto and T. W. Goeddel, "Classification of voiceband data signals using the constellation magnitude," *IEEE Transactions on Communications*, vol. 43, pp. 2759–2770, November 1995.

[46] C. U. Huang and A. Polydoros, "Likelihood methods for MPSK modulation classification," *IEEE Transactions on Communications*, vol. 43, pp. 1493–1504, Feb, March, April 1995.

[47] W. Wei and J. M. Mendel, "A new maximum-likelihood method for modulation classification," in *IEEE Asilomar Conference*, 30 Oct - 2 Nov 1995, pp. 1132–1136.

[48] ——, "Maximum-likelihood classification for digital amplitude-phase modulations," *IEEE Transactions on Communications*, pp. 189–193, Feb 2000.

[49] A. Swami and B. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on Communications*, vol. 48, pp. 416–429, March 2000.

[50] O. A. Dobre, Y. Bar-Ness, and S. Wei, "Higher-order cyclic cumulants for high order modulation classification."

[51] A. O. Hero and H. Hadinejad-Mahram, "Digital modulation classification using power moment matrices," in *IEEE conference on Accoustic Speech and Signal Processing*, May 1998, pp. 3285–3288.

[52] H. Roufarshbaf and H. Amindavar, "High-speed voiceband QAM constellation classification in multipath environment," in *IEEE Workshop on Neural Networks for Signal Processing*, Sep 2002, pp. 455–464.

[53] T. A. Drumright and Z. Ding, "QAM constellation classification based on statistical sampling for linear distortive channels," in *IEEE transactions on Signal Processing*, May 2006, pp. 1575–1586.

[54] ——, "Gibbs sampling classification of QAM signals in frequency selective channels," *IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 833–837, Nov 2002.

[55] Y. Bar-Shalom, "Tracking methods in a multitarget environment," *IEEE transactions on Automatic Control*, vol. 23, no. 4, pp. 618–626, August 1978.

[56] G. W. Pulford, "Taxonomy of multiple target tracking methods," *IEE Radar, Sonar, and Navigation*, vol. 152, no. 5, pp. 291–304, October 2005.

[57] R. W. Sittler, "An optimal data association problem in surveillance theory," *IEEE Transactions on Military Electronics*, vol. 8, pp. 125–139, April 1964.

[58] R. G. Sea, "An efficient suboptimal decision procedure for associating sensor data with stored tracks in real-time surveillance systems," in *Proceedings of the IEEE Conference on Decision and Control*, December 1971, pp. 33–37.

[59] R. A. Singer and J. J. Stein, "An optimal tracking filter for processing sensor data of imprecisely determined origin in surveillance systems," in *Proceedings of the IEEE Conference on Decision and Control*, December 1971, pp. 171–175.

[60] R. A. Singer and R. G. Sea, "New results in optimizing surveillance system tracking and data correlation performance in dense multitarget environments," *IEEE Transactions on Automatic Control*, vol. 18, pp. 571–581, December 1973.

[61] S. Blackman, *Multiple target tracking with radar applications*. Artech House, 1986.

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

[63] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, pp. 451–460, September 1975.

[64] Y. Bar-Shalom, T. E. Fortmann, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. OE-8, pp. 173–184, July 1983.

[65] J. K. Wolf, A. J. Viterbi, and G. S. Dixon, "Finding the best set of K paths through a trellis with applications to multitarget tracking," *IEEE transactions on Aerospace and Electronic Systems*, vol. AES-25, no. 2, pp. 287–296, March 1989.

[66] H. Gauvrit, J. P. L. Cadre, and C. Jauffret, "A formulation of multitarget tracking as an incomplete data problem," *IEEE transactions on Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1242–1257, October 1997.

[67] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, January 2004.

[68] C. Hue, J. P. LeCadre, and P. Perez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Transactions on Signal Processing*, vol. 50, pp. 309–325, February 2002.

[69] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

[70] R. Mahler, "PHD filters of higher order in target numbers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, pp. 1523–1543, Octobor 2007.

[71] B. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo methods for multitarget filtering with random finite sets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1224–1245, Octobor 2005.

[72] B. Vo and W. K. Ma, "The Gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, November 2006.

[73] A. Ben-Israel and T. N. Greville, *Generalized Inverses*. Springer-Verlag, 2003.

[74] G. Casella and R. L. Burger, *Statistical Inference*, 2nd ed., ser. Duxbury Advanced Series, 2001.

[75] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, March 1960.

[76] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*. Prentice Hall, 1993.

[77] T. E. Fortmann and Y. Bar-Shalom, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. OE-8, no. 3, pp. 173–184, July 1983.

[78] S. Haykin, "Cubature Kalman Filters," Web accessible presentation, accessed on March 1, 2009.

[79] W. Blanding, P. Willett, and Y. Bar-Shalom, "Offline and real-time methods for ML-PDA track validation," *IEEE Trans. on Signal Processing*, vol. 55, no. 5, pp. 1994–2006, May 2007.

[80] P. Billingsley, *Probability and Measure Theory*, 3rd ed. Wiley-Interscience.

[81] A. F. Karr, *Probability*. Springer-Verlag, 1992.

[82] B. D. Carlson, E. D. Evans, and S. L. Wilson, "Search radar detection and track with the Hough transform. I: System concept," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, pp. 102–108, January 1994.

[83] M. Liggins, C. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thomopoulos, "Distributed fusion architectures and algorithms for target tracking," *Proceedings of the IEEE*, vol. 58, no. 1, pp. 95–107, January 1997.

[84] F. Ehlers, "SEABAR'07 Cruise Report," 2008, NATO Undersea Research Center, La Spezia, Italy.

[85] D. Grimmett, "Specular-cued multistatic sonar tracking on the SEABAR'07 dataset," in *Proceedings of the 12th International Conference on Information Fusion (FUSION)*, July 2009, pp. 1576–1583.

[86] R. F. Gragg, "The BASIS-3D acoustic target strength model," Naval Research Laboratory, Washington, D.C., Tech. Rep. NRL/FR/7140-07-101052, 2002.

[87] S. Coraluppi, "Multistatic sonar localization," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 964–974, October 2006.

[88] P. C. Irwin and L. C. Wilbur, *Plane and Spherical Trigonometry*. New York, London: McGraw-Hill Book company, 1934.

[89] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation.* New York: John Wiley and Sons, 2001.

[90] H. A. P. Blom and Y. Bar-Shalom, "Multisensor tracking of a maneuvering target in clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, pp. 176–189, March 1989.

[91] W. Blanding, P. Willett, and Y. Bar-Shalom, "Multiple target tracking using maximum likelihood probabilistic data association," in *Proceedings of the 2007 IEEE Aerospace Conference*, March 2007, pp. 1–12.

[92] J. Shao, *Mathematical Statistics.* Springer, 2003.

# Curriculum Vitae

Hossein Roufarshbaf started his PhD with George Mason University in 2006, where he became a PhD candidate in 2010. He received his MSc from Amirkabir Univeristy of Technology (Tehran Polytechnique) in 2002 and his BSc from Isfahan University of Technology in 1999, all in Electerical Engineering. Before starting his PhD, he was with SIEMENS working on optical communication systems and next generation networks. During 2000-2004 he was with Iran Telecommunication Research Center (ITRC) where he became a faculty member in 2003. He received outstanding graduate student award from Volgenau School of Engineering in 2011.