Tracking the Operational Mode of Multi-Function Radar

TRACKING THE OPERATIONAL MODE OF MULTI-FUNCTION RADAR

By

JEROME DOMINIQUE VINCENT, B.Eng.
University of Madras, India, 2004

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Applied Science

McMaster University
August 2007

MASTER OF APPLIED SCIENCE (2007)	MCMASTER UNIVERSITY
(Electrical and Computer Engineering)	Hamilton, Ontario, Canada

TITLE:	**Tracking the Operational Mode of Multi-Function Radar**
AUTHOR:	Jerome Dominique Vincent B.Eng. University of Madras India, 2004
SUPERVISORS:	Dr. Simon Haykin and Dr. Thia Kirubarajan
NUMBER OF PAGES:	xii, 79

Abstract

This thesis presents a novel hybrid methodology using Recurrent Neural Network and Dynamic Time Warping to solve the mode estimation problem of a radar warning receiver (RWR). The RWR is an electronic support (ES) system with the primary objective to estimate the threat posed by an unfriendly (hostile) radar in an electronic warfare (EW) environment. One such radar is the multi-function radar (MFR), which employs complex signal architecture to perform multiple tasks. As the threat posed by the radar directly depends on its current mode of operation, it is vital to estimate and track the mode of the radar. The proposed method uses a recurrent neural network (echo state network and recurrent multi-layer perceptron) trained in a supervised manner, with the dynamic time warping algorithm as the post processor to estimate the mode of operation. A grid filter in Bayesian framework is then applied to the dynamic time warp estimate to provide an accurate posterior estimate of the operational mode of the MFR. This novel approach is tested on an EW scenario via simulation by employing a hypothetical MFR. Based on the simulation results, we conclude that the hybrid echo state network is more suitable than its recurrent multi-layer perceptron counterpart for the mode estimation problem of a RWR.

To My Family

Acknowledgments

I thank God for helping me complete this thesis and in all aspects of my life. I am greatly indebted to my supervisors, Dr. Haykin and Dr. Kirubarajan, for providing me an opportunity to be a graduate student. Their support and guidance to pursue this research work is deeply appreciated. I am also thankful to Dr. Dilkes from Defence Research & Development Canada (DRDC) for his involvement in this work.

I am grateful to the members of Adaptive Systems Laboratory, especially, I. Arasaratnam, Y. Xue and M. Ganapathy, and members of Estimation Tracking and Fusion Laboratory, especially, S. Thuraiappah and N. Nadarajah for their help whenever needed. I also wish to thank my roommates V. Rajakumar and G. Balasubramanium for all the support rendered.

Last but by no means least, I am deeply thankful to my family for their endless support, love and prayers.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we present a brief overview of the field of electronic warfare (EW) and its various divisions. We state and motivate the problem addressed in this thesis and also discuss the organization of the thesis.

1.1 Electronic Warfare

EW [1], in general, may be defined as a military action involving the use of electromagnetic energy to limit the hostile use of the electromagnetic spectrum by the enemy while retaining one's own friendly use of the electromagnetic spectrum. The control over the electromagnetic spectrum is crucial and may prove to be the decisive factor in the outcome of a battle. Although the field of EW is dynamically changing due to continually changing threats, it can be broadly classified into the following major categories:

- **Electronic Support Measure (ESM)** — refers to any passive military action aimed at real-time threat estimation. The primary actions of an ESM (e.g., RWR) include interception, identification, and analysis of hostile radiations that may serve as information for tactical decision-making.

- **Electronic Counter Measure (ECM)** — refers to any military action aimed at denying information that the enemy may obtain by effective use of the electromagnetic spectrum. ECM techniques may be subdivided into two sub-groups namely,
  - Passive ECM — includes the use of chaff, radar reflectors, stealth, etc.
  - Active ECM — sub-classed into soft-kill including jamming, deception, etc., and hard-kill including anti-radiation missiles, directed energy weapons, etc.
- **Electronic Counter-Counter Measure (ECCM)** — refers to actions taken to protect equipment and personnel, thereby reducing the impact of enemy ECM. The principles of ECCM may include techniques such as the frequency-hopping spread spectrum, which are generally embodied in the design of the equipment itself.

1.2 Multi-Function Radar (MFR)

The word radar is an acronym for *radio detection and ranging*. Radars are sensing equipments with a primary objective to detect the presence of a target and measure its range. They operate by transmitting electromagnetic energy into space and detecting the echo reflections from the target. Apart from detecting the target, information regarding its velocity, distance, etc., can also be extracted from the received echoes. These functions of radars along with their ability to perform them in extreme weather conditions make them indispensable to military applications. Over the years, several advancements in radar technology have resulted in radars evolving into more complex, sophisticated and powerful equipments. One such radar is the MFR, which poses a serious threat to the field of EW.

The MFR is capable of performing multiple functions such as search, acquisition, track, range resolution and track maintenance in a virtually simultaneous manner, hence its name. Complex signal architecture along with time-division multiplexing are employed to perform these multiple functions using an electronically controlled phased array antenna. A phased-array antenna contains a number of individual antenna elements, each connected to a phase shifter that is electronically controlled. The beam formation is accomplished by the principle of superposition of electromagnetic waves of different phases. As these phase shifters are controlled individually, beam steering can be done rapidly and efficiently.

Recent developments in technologies such as solid-state electronics, multiple-input multiple-output antennas, micro-controllers, software-defined radio have not only made the MFR a sophisticated military equipment but also facilitated the wide spread use of MFR in a variety of military applications.

1.3 Signal Intelligence (SIGINT)

Accurate and timely intelligence [1] on electromagnetic radiations from hostile sources is of vital importance to ensure efficient operation of each of the major divisions of EW. ESM is the division that is most dependent on the availability of prior intelligence. The analysis of data gathered from electromagnetic radiations may be classified into the following categories:

- **Communication Intelligence (COMINT)** — refers to intelligence obtained from the analysis of potentially hostile communication data.
- **Electronic Intelligence (ELINT)** — refers to the intelligence obtained from the analysis of any non-communicative electromagnetic radiation such as radar emissions. This is the most important source of information for the ESM.
- **Radiation Intelligence (RINT)** — refers to intelligence obtained from the analysis of unintended emissions from weapon systems. This form of intelligence may not be used frequently.

1.3.1 Electronic Intelligence (ELINT)

As mentioned earlier, ELINT [1] is the intelligence derived from non communicative sources. Radars are the main devices of interest for ELINT sensors. The electromagnetic radiations emitted by the radar are intercepted [5] by ELINT sensing devices. Analysis [4] of these signals provide vital information on the radar characteristics such as pulse per interval (PPI), pulse width, etc., which facilitate its identification and performance capabilities without being seen. This information coupled with information obtained from other non-electronic sources serve as the database for the ELINT library. The ELINT library is updated continually with the arrival of new information. In recent times, due to the large size of data and the need for ease of manipulation in real-time, storage of these sources of information has become an issue of concern. The design of an optimal structure for an ELINT library can be very challenging; however, techniques such as syntactic modeling, relational database modeling, etc., are available in the literature to tackle this issue. The use of ELINT library is very important and necessary for the efficient operation of ESM systems as it is prior information available about hostile radars. In a nutshell, ELINT can be thought of as the remote sensing of radar systems.

1.4 Radar Warning Receiver (RWR)

The RWR is a real-time electronic support system that is used for self protection. The primary function of RWR is threat estimation that may subsequently initiate tactical decision making. The RWR accomplishes its task through interception and analysis of hostile radiation from the enemy radar. As the RWR is a passive equipment, its interception and collection of radiations may not be detected by the enemy.

1.4.1 Signal Flow in a RWR

The signal processing in a typical RWR is shown in Figure 1.1. The RWR may achieve its objective by performing the following three important tasks:

- **Pulse Train De-interleaving** — The process of de-interleaving refers to isolating pulse trains associated with a particular radar from a set of overlapping pulse trains received from the radar within the RWR range of interception. This is the first step to threat recognition in a RWR, as it is difficult to identify the individual radars without the pulses being de-interleaved. A significant way to de-interleaving pulse trains is by exploiting their time of arrival (TOA).

- **Emitter Identification** — The de-interleaved pulse trains are then used to identify the particular radar emitter that radiated them. This is done by extracting features that are radar-specific such as pulse-repetition frequency (PPF), pulse width, amplitude, etc., from the pulses. These extracted features are then compared with the data stored in the ELINT library. Any matches found will identify the radar that radiated that particular pulse train.
- **Mode/Threat Estimation** — Once the particular radar is identified, the threat posed by the radar needs to be estimated as the threat directly depends on the radar's mode of operation. In the case of radars with hierarchical signal structure, the mode of operation can be estimated from the words extracted from the pulse trains.

1.5 Problem Statement

The problem addressed in this thesis is rooted in the concept of radar-target interaction. This is an important aspect of EW. It involves all the three main divisions of EW described earlier. There are two different perspectives to this concept.

From the radar's perspective it needs to search, detect, and track the target of interest. It needs to identify the critical parameters of the target such as its velocity, direction, etc. Besides these functions, the radar has to perform efficiently in the presence of any ECM initiated by the target. Therefore from the radar's view the target represents an uncooperative system that requires to be modeled.

From the target's perspective, say a military aircraft, it has to identify the operational mode of the radar, as the operational mode represents the function that the radar performs with respect to the target of interest. In this view, the radar itself

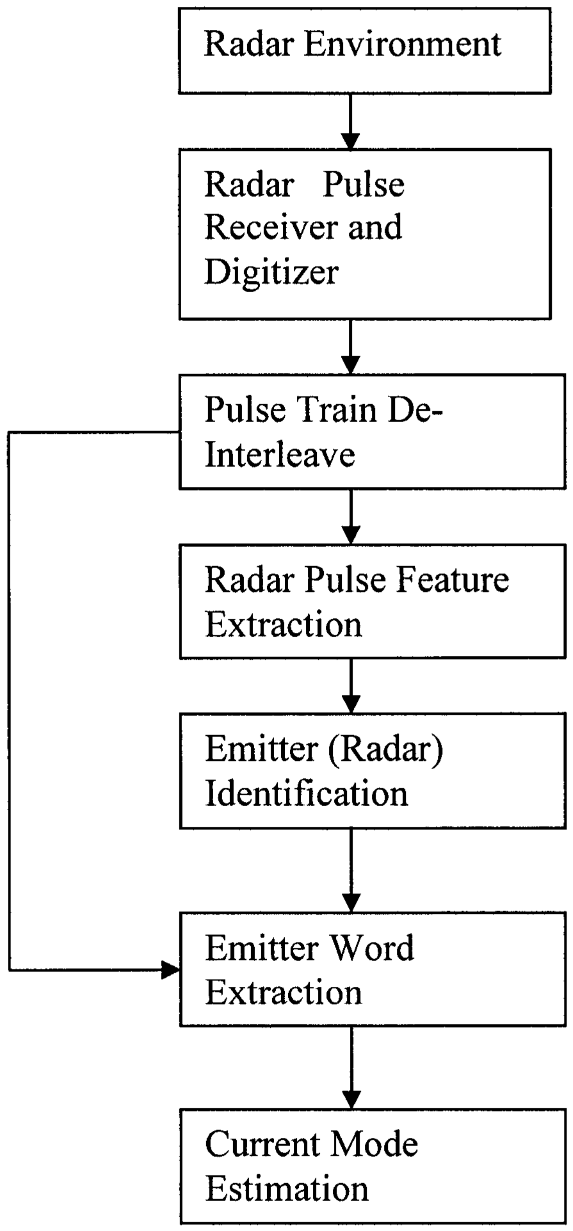
Radar Environment
Radar Pulse Receiver and Digitizer
Pulse Train De-Interleave
Radar Pulse Feature Extraction
Emitter (Radar) Identification
Emitter Word Extraction
Current Mode Estimation

Figure 1.1: Data processing in ESM System

becomes the target that needs to be tracked. Since the threat level directly depends on the operational mode, its knowledge is crucial for the target to estimate the threat posed by the radar. This is the topic of research addressed in this thesis. The motivation stems from the fact that, by estimating the operational mode, the threat can be estimated. This knowledge of threat posed by the radar, helps the target to initiate measures such as ECM or engage in evasive maneuvers to protect itself from the radar. This is the task of the RWR.

In order to accomplish this task, the RWR needs to perform the following two functions:

1. Identify the observed radar.
2. Estimate its operational mode.

In this thesis, we assume the observed radar is identified [15], hence we focus on estimating the operational mode in an efficient and timely manner. In this investigation, a recurrent neural network (RNN) trained in a supervised manner, with dynamic time warping (DTW) algorithm as the post processor is employed to obtain the minimum distance estimates of the operational modes. These estimates are smoothed by the application of a grid filter to obtain the posterior estimate of the radar's current operational mode.

1.6 Organization

This thesis is organized as follows. Chapter 2 describes the essential elements to the proposed solution, which include a detailed version of the MFR layered signal architecture and the theory of DTW. Chapter 3 describes the two RNNs employed in the proposed solution, namely, the recurrent multi-layer perceptron (RMLP) and the echo state network (ESN). This chapter is dedicated to the description of their architecture and training philosophies. Chapter 4 describes in brief two methodologies that are known in the literature as solutions to the stated problem. This chapter also states some of their limitations. Chapter 5 describes the grid filter that is an important component of the proposed solution in a detailed fashion. Chapter 6 describes the proposed solution with the problem formulation, methodology, simulation and results. Chapter 7 concludes the thesis with a summary of major contributions and improvements from the proposed methodology over previous methodologies.

Chapter 2

Essential Elements in the Mode Estimation of MFR

This chapter introduces some of the essential concepts involved in the mode estimation of MFR. It describes the layered signal architecture of MFR, which is proposed to keep the radar signal processing complexity manageable, the idea of radar mode evolution, and a brief overview on the theory of DTW.

2.1 Hierarchial Signal Structure

As mentioned in Chapter 1, the advancements in electronics and software have made the MFR a sophisticated military equipment. The pulse structure of MFR has become more complex as a result of software controlled radar pulse generation. This complex pulse structure along with the capability to perform multiple tasks makes mode estimation of MFR difficult at the pulse level. Therefore, to overcome this difficulty and to keep the signal processing complexity manageable a hierarchial signal structure [6] is proposed as shown in Figure 2.1. Estimation of the operational mode is obtained through word level processing which is subsequent to the pulse level processing.

A *word* is composed of a specific group of pulses. It is divided into five sections (A-E), of which pulses are transmitted only in section B and section D. Section B is known as the *pulse doppler sequence.* The pulse doppler sequence is unique for each word. It has a set of pulses with a characteristic pulse per interval (PPI) for each word which makes them unique. Section D is known as the *synchronization burst sequence* that is common to all the words and sections A,C,E are known as *dead time zones* as no pulses are transmitted in these intervals. The MFR has a total of nine different words (W1 - W9) with similar pulse envelope.

A *phrase* is made up of a particular combination of four words. Each phrase represents a mode of operation of MFR but not in a unique way. Therefore, a particular phrase may represent more than one mode. In the same way several different phrases may represent a single mode of operation. For example the phrase [W6 W6 W6 W6] may represent two modes of operation such as acquisition and track maintenance. Therefore, the mapping between a mode and a phrase is many-to-many which makes the task of mode estimation for the MFR more complicated. Table 2.1 shows the different operational modes and their corresponding phrase combinations.

A *clause* is made up of combination of phrases. The number of phrases in a clause represents the number of operations performed by the MFR simultaneously in a time-multiplexed manner. The clauses are concatenated sequentially one after the other, same as in words and phrases. In that respect the last word of $(n)^{th}$ clause is followed by the first word of $(n+1)^{th}$ clause. However, when

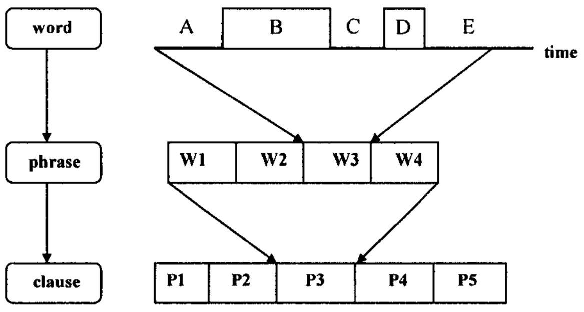

Figure 2.1: Layered signal architecture

the clauses are represented as stacks in a two-dimensional table, the phrases corresponding to each of the multiplexed tasks can be analyzed by reading the table from top to bottom along the phrase boundary. The boundaries of the phrases are represented by dotted vertical lines. Figure 2.2 shows the MFR performing five modes simultaneously in a time-multiplexed manner with respect to five targets.

Since there are efficient algorithms for the extraction of words from the received pulse trains [7], we will focus on mode estimation at the word level, which is less complex and less error prone than the pulse level.

Functional State	Phrase Content	Functional State	Phrase Content
Four-Word search	$[W_1W_2W_4W_5]$	TM	$[W_1W_7W_7W_7]$
	$[W_2W_4W_5W_1]$	(Track-	$[W_2W_7W_7W_7]$
	$[W_4W_5W_1W_2]$	Maintenance)	$[W_3W_7W_7W_7]$
			$[W_5W_7W_7W_7]$
Acquisition	$[W_3W_3W_3W_3]$		
	$[W_4W_4W_4W_4]$		
	$[W_5W_5W_5W_5]$		
NAT	$[W_1W_6W_6W_6]$		
(Non-Adaptive	$[W_2W_6W_6W_6]$		
Track)/TM	$[W_3W_6W_6W_6]$		
Range Resolution	$[W_7W_6W_6W_6]$		
	$[W_8W_6W_6W_6]$	Acq.,NAT or TM	$[W_6W_6W_6W_6]$
	$[W_9W_6W_6W_6]$		

Table 2.1: Typical phrase combinations of MFR corresponding to operational modes. There are a total of nine different words.

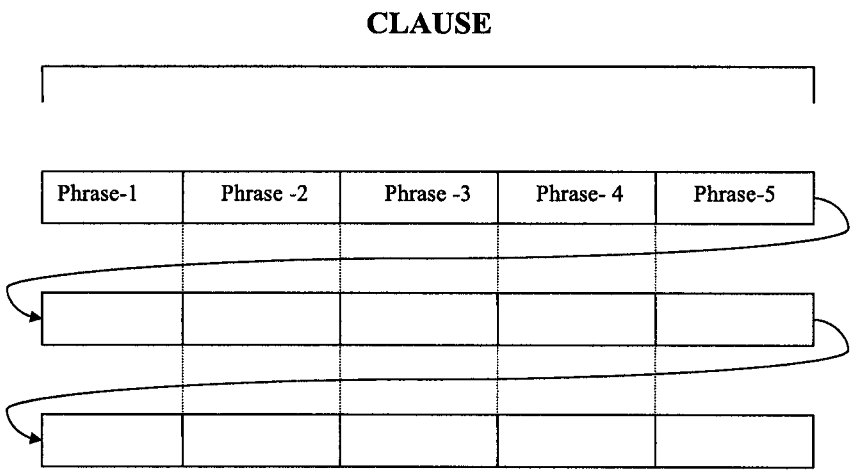

Figure 2.2: Output sequence of the MFR

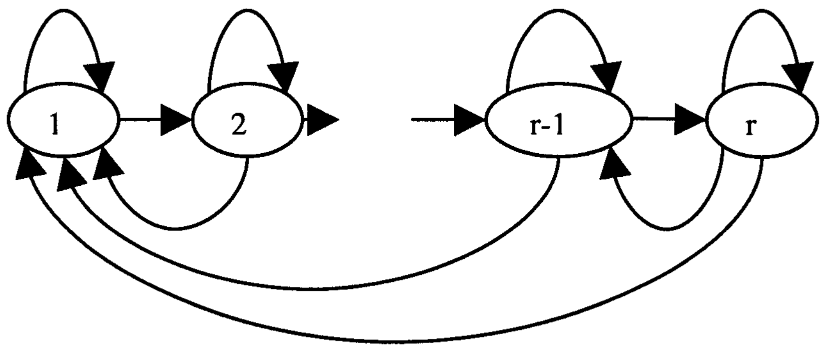

Figure 2.3: Mode evolution of an MFR in a typical first order Markov chain.

2.2 Mode Evolution of MFR

The knowledge of mode evolution is an a priori information for the mode estimation of the MFR. The mode evolution of MFR may be described by a Markov chain with known transition probabilities. A Markov chain is a sequence of events that follows the Markov property by which the state of a system at the $(n)^{th}$ time instance depends only on its state at the $(n-1)^{th}$ time instance. It can be formerly described by the following equation in which $X_1, X_2, \ldots, X_n$ are random variables.

$$Pr(X_{n+1} = x | X_n = x_n, \ldots, X_1 = x_1, X_0 = x_0) = Pr(X_{n+1} = x | X_n = x_n) \tag{2.1}$$

The modes of operation of MFR are finite and since they follow the Markov property in their evolution, the mode evolution process may be called as a Markov chain with a finite state space. It is represented in Figure 2.3. The states represent the modes of operation and the transition probability distribution is represented by a matrix with elements given by

$$Pr(X_{n+1} = i | X_n = j) \tag{2.2}$$

Therefore, the mode evolution of an MFR is a finite state Markov chain with known transition probabilities that are invariant in time.

2.3 Dynamic Time Warping (DTW)

DTW is a sequence alignment (template matching) technique widely used in speech recognition, robotics, pattern recognition, etc. [8]. It is a well known post-processing tool for neural networks. It is used to measure the similarity between two given sequences. In DTW, the similarity is estimated as a measure of global distance between the two given sequences. As similarity is inversely proportional to distance, minimizing the distance signifies maximizing the similarity between the two sequences. An optimal alignment path is constructed between the two sequences to measure the global distance between them.

2.3.1 Construction of an Optimal Path

Consider $r(i),\ i = 1,2,\ldots,N$ and $t(j),\ j = 1,2,\ldots,M$ as the feature vectors representing the reference and test sequences, respectively. These two vectors need not be of same length. Now the task is to estimate the similarity between the two sequences by measuring the global distance between them. Therefore, to accomplish this, a two-dimensional grid structure is constructed with the elements of the

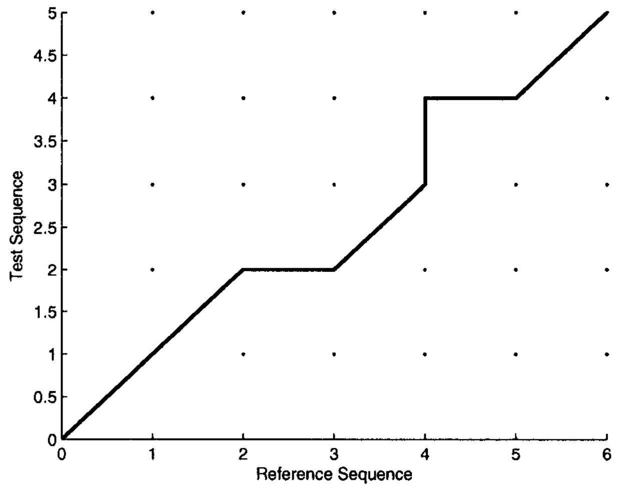

Figure 2.4: Warp path between the test and reference sequence from initial node to the final node.

reference sequence as the horizontal axis and those of the test sequence as the vertical axis. The dimension of the grid would be $(M \times N)$ in this case. Each element (i, j) in the grid is referred to as a node. Associated with each node (i, j) is a cost function. The function $d(i, j)$, represents the distance between the corresponding elements of the test and reference sequences, is the cost function associated with each node.

Now having constructed the grid, there are several choices of paths through the grid from the initial node (i_0, j_0) to the final node (i_N, j_M). Each path has a global cost D associated with it, defined as

$$D = \sum_{k=0}^{K-1} d(i_k, j_k) \tag{2.3}$$

Here K refers to the total number of nodes in that particular path. The path which minimizes this cost is the optimal path. This optimal path may be obtained by the principle of dynamic programming (DP). Figure 2.4 shows the warp path between the test and reference sequence.

2.3.2 Dynamic Programming (DP)

The principle of dynamic programming was first introduced by Bellman [8]. It states that an optimal path between the initial and final node is the concatenation of the optimal paths for each of the intermediate nodes that lie between the initial and final nodes. If (i_r, j_r) is an intermediate node between the initial node (i_0, j_0) and final node (i_N, j_M) and if the optimal path has to pass through it, then by Bellman's DP the optimal path is given by

$$(i_0, j_0) \xrightarrow{opt} (i_N, j_M) = (i_0, j_0) \xrightarrow{opt} (i_r, j_r) \oplus (i_r, j_r) \xrightarrow{opt} (i_N, j_M) \tag{2.4}$$

where $\xrightarrow{opt}$ refers to the optimal path from one node to another and $\oplus$ refers to the concatenation of these optimal paths. This principle of DP may be applied to find the optimal path through the grid resulting in minimizing the cost function given by (2.3). Therefore, the optimal path which minimizes (2.3) is given by

$$D_{min}(i_N, j_M) = \min_{(i_{N-1}, j_{M-1})} [D_{min}(i_{N-1}, j_{M-1}) + d(i_N, j_M)|(i_{N-1}, j_{M-1})] \tag{2.5}$$

where $d(i_N, j_M|i_{N-1}, j_{M-1})$ refers to the cost associated with transition from node (i_{N-1}, j_{M-1}) to node (i_N, j_M) which is added on to the minimum cost incurred up to node (i_{N-1}, j_{M-1}) . Step (2.5) is repeated right from the initial node to the final node. Thus by DP the overall cost is minimized if each of the transitional cost between the initial node to the final node is minimized. However this is done with some constraints which determine the direction and scope of node transition.

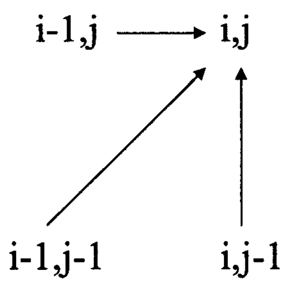

Figure 2.5: Local constraint indicating the allowable node transitions.

Constraints — The optimal transition from one node to another is defined by a set of constraints. They are described as follows:

1. **End Point Constraint**— defines the start and end points of the path. In the case described above, by the end point constraint the start point is the initial node (i_0, j_0) and the end point is the final node (i_N, j_M).

2. **Local Constraint** — defines the possible transitions that are allowed to reach a particular node. It is based on the principle of monotonicity as shown in Figure 2.5. For example, a transition to a node (i_N, j_M) is possible only from either one of $(i_{N-1},\ j_M)$, (i_{N-1}, j_{M-1}), (i_N, j_{M-1}) nodes. In other words, there are a set of predecessors for each node in the grid. These predecessors are located only to the left or south of the node of transition interest.

3. **Global Constraint** — defines a boundary for the nodes that are to be searched for an optimal path. The nodes that fall outside the boundary are not included in the search. In other words, this constraint specifies a region of search within the end points defined by the end point constraint.

Chapter 3

Recurrent Neural Networks (RNNs)

In this chapter, we introduce the concept of recurrent neural networks. We describe in detail two different architectures of RNNs, their training philosophies and algorithms. The two architectures described are the recurrent multi-layer perceptron (RMLP) and the echo state network (ESN).

3.1 Introduction

RNNs are a type of artificial neural networks which are characterized by the use of feedback. Feedback [10] is said to exist in a system if the system output influences the system input in part, by forming a closed loop transmission path from the output to the input of the system. RNNs are inspired by the biological neural network which is recurrent in nature. Due to the presence of feedback RNNs can perform input-output mapping of a system and hence are dynamically driven networks capable of mimicking dynamic systems with arbitrary precision. RNNs are widely used in areas such as controls, speech recognition, robotics, telecommunication, etc., for tasks that include prediction, pattern classification, system identification, filtering etc. Though there are different architectures and training algorithms for RNNs, in this thesis we will focus on two types of RNNs architectures, namely, the RMLP trained with the extended Kalman filter and the ESN.

3.2 Recurrent Multi-Layer Perceptron (RMLP)

The RMLP [10] [19] is one type of RNN architecture. The RMLP has layers of neurons between the input and output layers. These layers are called hidden layers. The RMLP must have at least one hidden layer of recurrent neurons. Figure 3.1 illustrates a RMLP with two hidden layers and an output layer. The dynamic behavior of the RMLP is described by a system of coupled equations [10] as follows.

$$x_I(n+1) = \varphi_I(x_I(n), u(n)) \tag{3.1}$$

$$x_{II}(n+1) = \varphi_{II}(x_{II}(n), x_I(n+1)) \tag{3.2}$$

$$x_o(n+1) = \varphi_o(x_o(n), x_{II}(n+1)) \tag{3.3}$$

where $\varphi_I(\cdot)$, $\varphi_{II}(\cdot)$ and $\varphi_o(\cdot)$ are the activation functions of the first hidden layer, second hidden layer, and output layer, respectively.

There are many training algorithms such as back propagation through time (BPTT), real time recurrent learning (RTRL), extended Kalman filter (EKF), etc., to train the RMLP. Each of these algorithms has its own merits. However, in this thesis we will focus on training the RMLP with the EKF algorithm as it is believed to give the best results [12].

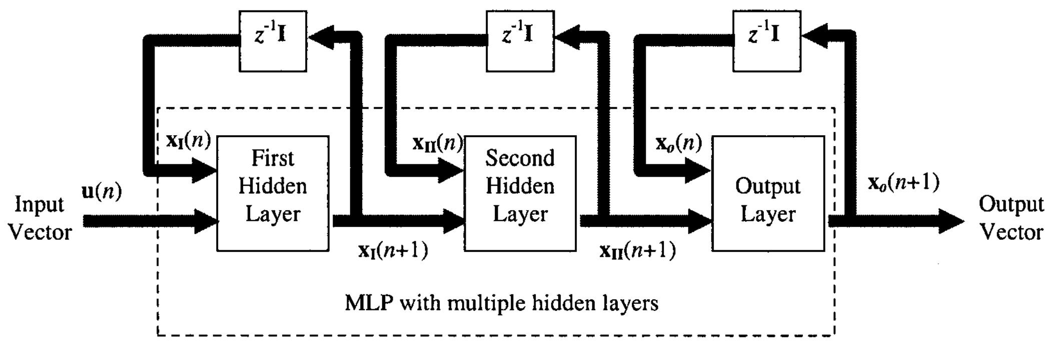

Figure 3.1: Structure of a Recurrent Multi-Layer Perceptron.

3.2.1 Training the RMLP with Extended Kalman Filter (EKF)

The EKF is a state estimation technique for non-linear systems. The technique is similar to the classical Kalman filter [22] but with an extension that uses linearization to handle non-linear systems. Once linearized, the classical Kalman filter that provides optimal estimates for linear systems is applied for state estimation. The application of the EKF to train the RMLP is motivated by the fact that it can be used to estimate the weights of the network [21]. Therefore, the weights of the network takes the notion of states in an EKF.

Consider a RMLP with *W* synaptic weights and *P* output units. Then the state space model of the network may be given as [21] [10]

$$w(n+1) = w(n) \tag{3.4}$$

$$d(n) = c(w(n), u(n), v(n)) + \nu(n) \tag{3.5}$$

Equation (3.4) is known as the process equation and equation (3.5) is known as the measurement equation. In the process equation, *w(n)* is the weight vector that refers to the state and is assumed to be noiseless. Also it is assumed that the network is in an optimum condition with a weight transition matrix being equal to the identity matrix. In the measurement equation, *u(n)* and *v(n)* refers to the input vector and recurrent node vector, respectively. $\nu(n)$ denotes the measurement noise vector which is a zero mean, white noise process with a co-variance defined by

$$R(n) = E[\nu(n)\nu^T(n)] \tag{3.6}$$

$c(.)$ in (3.5) denotes the entire non-linearity right from the input layer to the output layer of the RMLP. Therefore, from (3.4) and (3.5) it can be seen that the non-linearity is associated only with (3.5) hence it is necessary to linearize it before applying the Kalman filter. The linearized version of (3.5) is given by

$$d(n) = C(n)w(n) + \nu(n) \tag{3.7}$$

where $C(n)$ is the measurement matrix of the linearized form of equation (3.5). The matrix consists of partial derivatives of p outputs of the whole network with respect to the W weights of the networks as shown below.

$$C(n) = \begin{bmatrix} \frac{\partial c_1}{\partial w_1} & \frac{\partial c_1}{\partial w_2} & \cdots & \frac{\partial c_1}{\partial w_W} \\ \frac{\partial c_2}{\partial w_1} & \frac{\partial c_2}{\partial w_2} & \cdots & \frac{\partial c_2}{\partial w_W} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial c_p}{\partial w_1} & \frac{\partial c_p}{\partial w_2} & \cdots & \frac{\partial c_p}{\partial w_W} \end{bmatrix} \tag{3.8}$$

The partial derivatives are with respect to the estimated weights $\hat{w}(n)$ computed by the EKF. These partial derivatives may be computed by using the BPTT [16], or the RTRL. One of these two algorithms has to be used to linearize (3.5) to (3.7). However, in this thesis we will use the truncated version of BPTT algorithm as illustrated in [19] to compute the $C(n)$ matrix since it is efficient as well as computationally affordable. The computation cost for RTRL is $O(N^4)$. This is much higher than that of truncated BPTT, which is $O(dN^2)$, where N is the number of recurrent units and d is the truncation depth. Besides computational cost it has been reported in the literature that best results in training RMLP may be obtained by using the EKF to estimate the weights of the network with truncated BPTT for linearization which involves the computation of $C(n)$ [12], [19]. Therefore with the computation of $C(n)$ equation (3.5) may be recast in the form of (3.7). Now the classical Kalman filter may be applied to estimate the weights of the RMLP.

Before deriving the Kalman filter equations, we mention two variations in the application of EKF, namely,

1. Global EKF

2. Decoupled EKF

In the case of global EKF, the entire linearized measurement matrix $C(n)$ is used as a whole in the derivation of the Kalman filter. This is computationally expensive but more accurate. On the other hand, in the case of decoupled EKF the network synaptic weights may be partitioned into, say, g groups with the i^{th} group containing j_i neurons. Then the partial derivatives of the $C(n)$ measurement matrix have to be rearranged according to the weights corresponding to each of the individual neurons in the network in a way that they are grouped as a single block within $C(n)$. $C(n)$ would then be a concatenation of the partial derivatives corresponding to each group i, where $i = 1, 2, \ldots, g$.

$$C(n) = [C_1(n), \cdots, C_g(n)] \tag{3.9}$$

This decoupled nature of EFK was first introduced in [18]. It is computationally more efficient than the global EKF with slight compromise in accuracy. The derivation for the decoupled version of EKF is given as follows [10], [18].

$$\Gamma(n) = \left[\sum_{i=1}^{g} C_i(n)K_i(n,n-1)C_i^T(n) + R(n)\right]^{-1} \tag{3.10}$$

$$G_i(n) = K_i(n,n-1)C_i^T(n)\Gamma(n) \tag{3.11}$$

$$\alpha(n) = d(n) - \hat{d}(n|n-1) \tag{3.12}$$

$$\hat{w}(n+1|n) = \hat{w}_i(n|n-1) + G_i(n)\alpha(n) \tag{3.13}$$

$$K_i(n+1,n) = K_i(n,n-1) - G_i(n)C_i(n)K_i(n,n-1) \tag{3.14}$$

where $\Gamma(n)$ is the conversion factor that relates the estimation error $e(n)$ to the innovations $\alpha(n)$ by [10]

$$e(n) = R(n)\Gamma(n)\alpha(n) \tag{3.15}$$

$G_i(n)$ is the Kalman gain that determines the correction used to update the weight estimate for group i of neurons. $\alpha(n)$ is the innovation, which is defined by the difference between the desired response $d(n)$ and the actual network output $\hat{d}(n|n-1)$. $\hat{w}_i(n|n-1)$ is the estimate of the weight $w_i(n)$ for group i of neurons and $K_i(n,n-1)$ is the error covariance matrix for group i of neurons.

3.2.2 Effect of Artificial Process Noise

The non-linear dynamical system described by (3.4) and (3.5) can lead to some numerical difficulties during estimation. It suffers from what is called as the divergence phenomenon in which the update matrix $K_i(n+1,n)$ may not be non-negative definite for every iteration of the algorithm due to numerical inaccuracies.

This has to be avoided as $K_i(n+1,n)$ represents a covariance matrix which has to be non-negative definite. This problem can be circumvented by incorporating artificial process noise in the process (3.4) as follows

$$w_i(n+1) = w_i(n) + \omega_i(n) \tag{3.16}$$

where $\omega_i(n)$ is the process noise which is assumed to be a zero mean white Gaussian noise of diagonal co-variance matrix $Q_i(n)$. With the incorporation of $\omega_i(n)$, (3.14) can be recast into

$$K_i(n+1,n) = K_i(n,n-1) - G_i(n)C_i(n)K_i(n,n-1) + Q_i(n) \tag{3.17}$$

Therefore by (3.17), $K_i(n+1,n)$ will remain non-negative definite as long as $Q_i(n)$ is large enough. Apart from overcoming the divergence phenomenon, by the addition of process noise there is less likelihood for the algorithm to be trapped in local minimum, which therefore results in improved speed of convergence and accuracy in the solution [18]. Hence, because of these benefits we incorporate artificial process noise in the training of RMLP with EKF algorithm.

3.2.3 Back-Propagation Through Time (BPTT)

The BPTT is a gradient decent algorithm to train a RNN. It is an extension of the well known back-propagation algorithm which is used to train feed-forward neural networks. As the back-propagation algorithm cannot be employed directly to RNN, it is used with an extension and is called back-propagation through time. The extension here lies in unfolding the RNN in time into a feed-forward network. Once this is accomplished, the traditional back-propagation algorithm is employed to compute the weight adjustments. The unfolding of the RNN is done in time. Some interesting properties relating a RNN with its unfolded counterpart are described in [10]. Once the RNN is unfolded in time into a feed-forward network the back-propagation algorithm is used to compute the sensitivity of the network, which is the partial derivatives of the cost function (ξ) with respect to the synaptic weights of the network. The cost function is defined by

$$\xi_{total}(n_0, n_1) = 1/2 \sum_{n=n_0}^{n_1} \sum_{j \in \mathcal{A}} e_j^2(n) \tag{3.18}$$

where $\mathcal{A}$ is the set of indices pertaining to those neurons in the network for which the desired response is specified. n_0 is the start time and n_1 is the end time. $e_j^2(n)$ is the error signal at the output of j neuron with respect to some desired response. However, in minimizing this cost function, the network has to remember information right up to the start time n_0 for every iteration. This is computationally too expensive and is not feasible in real time. Therefore, to overcome this limitation a truncation depth h is used in which any information older than h time steps will be ignored. This version is called as the truncated BPTT [20]. The cost function to be minimized for the truncated BPTT is defined as

$$\xi(n) = (1/2) \sum_{j \in \mathcal{A}} e_j^2(n) \tag{3.19}$$

By the traditional back-propagation algorithm, the sensitivity of the network is given by

$$\Delta w_{ji} = -\eta \frac{\partial \xi(l)}{\partial w_{ji}} \tag{3.20}$$

$$= \eta \sum_{l=n-h+1}^{n} \delta_j(l) x_i(l-1) \tag{3.21}$$

The above equation holds for all $j \in \mathcal{A}$ and $(n-h) < l \leq n$. The learning-rate parameter is represented by η. The input applied to the i^{th} synapsis of neuron j at time $(l-1)$ is given by $x_i(l-1)$ and $\delta_j(l)$ is the local gradient defined by

$$\delta_j(l) = \frac{\partial \xi(l)}{\partial \upsilon_j(l)} \tag{3.22}$$

In the case of $l = n$ the local gradient $\delta_j(l)$ is given by

$$\delta_j(l) = \varphi'(\upsilon_j(l)) e_j(l) \tag{3.23}$$

where $\varphi'(.)$ is the derivative of the activation function $\upsilon_j(l)$ applied to the neuron j. In case of $(n-h < l < n)$ the local gradient $\delta_j(l)$ is given by

$$\delta_j(l) = \varphi'(\upsilon_j(l)) \sum_{k \in \mathcal{A}} w_{kj}(l) \delta_k(l+1) \tag{3.24}$$

Equations (3.23), (3.24) are repeated from time n right back to time $n-h$, and with the computation of back propagation at time $n-h+1$ the adjustment weights are computed according to (3.21). Thus it can be seen from (3.23) and (3.24) that the error signals is used only in (3.23) which is the computation at time n. This indicates that the past records of the desired responses have not been stored which makes the truncated BPTT computationally feasible in real time.

3.3 Echo State Network (ESN)

The echo state network is an RNN recently introduced in [11, 13, 14]. As mentioned earlier all RNNs are characterized by the use of feedback, inspired by the biological neural network which is recurrent in nature. RNNs have dynamic memory and are used for black-box modeling of non-linear systems. The ESN differs from other RNNs in two ways:

1. **Architecture** — The ESN consists of a large number of recurrent units usually in the order of hundreds (100-1000). These recurrent units form a reservoir. But, in case of conventional RNN such as the RMLP, recurrent units are represented in the form of hidden layers and are not large in number.

2. **Learning Algorithm** — The ESN learning algorithm is supervised in nature. In the case of ESN, during training only the output synaptic connections are modified, whereas in other RNN learning methods the entire synaptic connections are altered. The training of ESN is done by linear regression.

Figure 3.2 shows an ESN with a single input unit, N internal units and a single output unit. These N internal units are recurrent in nature and thus form the reservoir of dynamics. The activations of the internal and output units are represented as $x(n)$ and $y(n)$, respectively. The input is represented by $u(n)$. The updating of

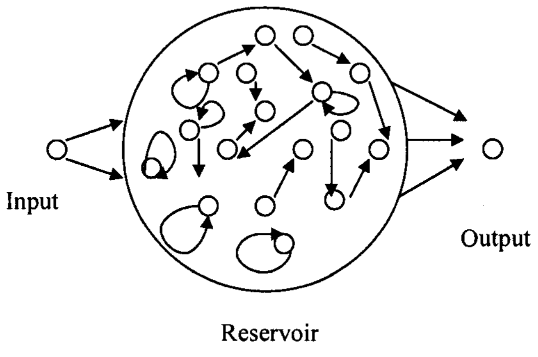

Figure 3.2: Structure of an ESN with a single input and a single output.

the activations of the reservoir and the output unit are given as

$$x(n+1) = f(Wx(n) + W^{in}u(n+1) + v(n+1)) \tag{3.25}$$

$$y(n+1) = f^{out}(W^{out}(x(n+1))) \tag{3.26}$$

where W is the internal weight matrix of dimension $N \times N$, W^{in} is the input weight matrix and W^{out} is the output weight matrix which has to be computed. These output weights are the only synaptic connections that are adapted during the training process once the network is chosen. The weights corresponding to the synaptic connections of the input and the reservoir remain unaltered during the training process. $u(n+1)$ represents the input vector, f denotes the transfer function of individual elements, and $v(n+1)$ is the optional noise vector.

3.3.1 Concept of Echo States

A network is an ESN if its dynamic reservoir has echo states. Echo states constitute a property that the network exhibits before it is trained. In fact, the echo state property is related to the properties of the reservoir matrix W. Therefore, a trained network is an ESN if its untrained version exhibits the echo state property. In this section, we paraphrase the discussion provided in [12, 14] on the properties and characterization of echo states. A formal definition of echo states as stated in [14] is given below.

Definition If an untrained network with weights W^{in}, W is driven by teacher input $u(n)$ from a compact interval U and has network states $x(n)$ from a compact set A, then the network has echo states if the network state $x(n)$ is uniquely determined by any left-infinite input $u(n-1)$, $u(n)$ where $n = -2, -1, 0$.

In an intuitive sense, the echo-state property states that if a network is run for a very long time from negative infinity (i.e. left-infinite property) then the current network states can be uniquely determined by the input history. As mentioned earlier the property of echo states depends on the algebraic properties of the weight matrix W of the dynamic reservoir. As there is no necessary and sufficient algebraic conditions [12] to show that the network has echo states given the weight matrix (W, W^{in}), we resort to a heuristic approach to obtain an echo-state network. Before describing the heuristics we discuss in proposition (3.3.1) the sufficient condition for the non-existence of echo states.

Proposition 3.3.1 *Consider an untrained network (W, W^{in}) with the state update according to (3.25) and tanh as the transfer function. This network does not have echo states with respect to the input interval U if the spectral radius of the dynamic* *reservoir weight matrix W is greater that unity, that is, if* $|\lambda_{max}| > 1$. *In this,* $|\lambda_{max}|$ *is the largest absolute value of an eigenvector of W.*

Though there is no sufficient algebraic condition for finding echo-state networks, in practice it is believed that a network exhibits echo states if Preposition 3.3.1 is not satisfied. In not satisfying Preposition 3.3.1 the spectral radius $|\lambda_{max}|$, which is the largest absolute value of eigenvector of W, is less than unity. Therefore this heuristic approach [12] as stated in Conjecture 1 is used to design an echo-state network.

Conjecture 1 Let δ and ε be two small numbers. Then for a dynamic reservoir of size N a random matrix W_0 is created by sampling the weights from a uniform distribution between [-1,1]. Then the matrix W_0 is normalized to W_1 by dividing W_0 by its spectral radius. Scaling W_1 to $W = (1-\delta)W_1$ where $(1-\delta)$ is the spectral radius of W; hence, the network (W^{in}, W) is an echo state network with probability $1-\varepsilon$.

It can be understood from Preposition 3.3.1 and Conjecture 1 that the key to achieving echo states is by having the spectral radius of the dynamic reservoir less than unity. It should also be noted that the echo-state property only depends on the properties of the dynamic reservoir weight matrix W and has nothing to do with the input weight matrix W^{in}. Therefore, the input weight matrix W^{in} can be freely chosen without affecting the echo-state property.

3.3.2 Training an Echo-State Network

Having described the property of echo states along with the heuristic conditions for an ESN, we present the idea behind training an ESN. First, we discuss the training principle of an ESN followed by the algorithm employed to train it. A detailed discussion on these topics is presented in [14], [12].

3.3.2.1 Principle

The main difference between an ESN and other RNN structures is that in the case of ESN only the output synaptic weights are modified during training whereas in other RNN structures the entire synaptic weights are modified. Therefore the key aspect of the ESN training lies in the computation of the output weights W^{out}. The principle employed in the computation of output weights is linear regression. The training principle is explained as follows.

Consider an ESN, which is to be driven by a set of inputs training data of length n_{max} represented by $u(n)$. Then, since the ESN satisfies the echo-state properties as described in the previous section, after some initial transient period the internal network states $x_i(n)$ of the dynamic reservoir may be represented as

$$x_i(n) \approx e_i(\ldots, u(n), u(n+1)) \tag{3.27}$$

where e_i represents the echo function of the i^{th} unit. The network weight matrix is assumed to be heterogeneous; therefore, the echo functions will differ from each other. Let the desired output be denoted by $d(n)$. Then, the network output is given by

$$y(n) = f^{out}\left(\sum_{i=1}^{N} w_i^{out} x_i(n)\right) \tag{3.28}$$

where w_i^{out} represents the i^{th} output connection which needs to be computed. The f^{out} is a $tanh$ function which is invertible. Hence, (3.28) may be rewritten as

$$(f^{out})^{-1} y(n) = \sum_{i=1}^{N} w_i^{out} x_i(n) \tag{3.29}$$

The echo functions as in (3.27) are substituted for $x_i(n)$ in (3.29) which gives

$$(f^{out})^{-1}y(n) = \sum_{i=1}^{N} w_i^{out} e_i(\ldots, u(n), u(n+1)) \tag{3.30}$$

Now the output weights w^{out} which constitute the key ingredient in the training of an ESN are computed by minimizing the training error in the mean square error (MSE) sense. The training error is given by

$$\varepsilon_{train}(n) = (f^{out})^{-1}d(n) - (f^{out})^{-1}y(n) \tag{3.31}$$

Substituting (3.30) in (3.31), the error is obtained as

$$\varepsilon_{train}(n) = (f^{out})^{-1}d(n) - \sum_{i=1}^{N} w_i^{out} e_i(\ldots, u(n), u(n+1)) \tag{3.32}$$

Therefore, the MSE, which is to be minimized to obtain W^{out}, is given as

$$MSE_{train} = \frac{1}{(n_{max} - n_{min})} \sum_{i=n_{min}}^{n_{max}} \varepsilon_{train}^2(n) \tag{3.33}$$

Now, (3.33) can be written as

$$MSE_{train} = \frac{1}{(n_{max} - n_{min})} \sum_{i=n_{min}}^{n_{max}} \left((f^{out})^{-1}d(n) - \sum_{i=1}^{N} w_i^{out} x_i(n) \right)^2 \tag{3.34}$$

where n_{min} refers to the length of the initial transient that is dismissed and not used in the computation of the output weight. Since the minimization of (3.33) is a linear regression task, the training of the ESN which mainly involves the computation of the output weight W^{out} matrix is by linear regression. If $f^{out} = 1$

as in the case of linear units then (3.34) may be written as

$$MSE_{train} = \frac{1}{(n_{max} - n_{min})} \sum_{i=n_{min}}^{n_{max}} \left(d(n) - \sum_{i=1}^{N} w_i^{out} x_i(n) \right)^2 \tag{3.35}$$

3.3.2.2 Algorithm

The training algorithm for an ESN may be described in four stages:

1. Procure an ESN
2. Network state updating
3. Computation of output weights
4. Testing

The first three stages constitute the training phase, and the last stage is the testing phase.

Procure an ESN — An ESN is one which satisfies the echo-state properties described earlier. Therefore, to accomplish this we follow the heuristic approach as discussed in Preposition 3.3.1 and Conjecture 1. The following heuristics guarantees an ESN.

- The network weight matrix W_0, is a sparse matrix generated randomly from a uniform distribution [-1,1]. It is then normalized to W_1 with the absolute value of its spectral radius.
- The Network weight matrix W which has the echo state property is then obtained as $W = \alpha W_1$ where $\alpha < 1$ is the spectral radius of W.

- The input weights W^{in} are randomly generated. Now the untrained network W^{in}, W is an ESN.

- The size of the weight matrix W is always between $T/10$ and $T/2$, depending on the complexity of the task. T is the length of the training data.

- The parameter α is of crucial importance for the successful training of an ESN. Since the spectral radius of the reservoir weight matrix is connected to the intrinsic timescale of the dynamics of the reservoir state, a diligent choice of α would improve the probability of successful training. Typical range of α lies between (0.78 - 0.98).

Network State Updating — Once the ESN is setup, it can be trained by presenting the input training sequence $u(n)$. The state update is done as follows:

- The initial network state is set to zero, $x(0) = 0$. Then the network is trained by presenting the input training data $u(n)$ and the network state update is performed by

$$x(n+1) = f(Wx(n) + W^{in}u(n+1) + v(n+1)) \tag{3.36}$$

- For each time n, after an initial transient, the network states $x(n)$ are collected as a row in a state collecting matrix M and the sigmoid-inverted desired output $tanh^{-1}d(n)$ is collected as a row in a output collecting matrix T.

Computation of Output Weights — This is the most important part of the training algorithm since in the case of ESN only the output weights are adapted. In the previous section, it was shown that the computation of W^{out} is a linear regression task. This can be achieved by multiplying the pseudo-inverse of M which stores the collected network states with T which stores the collected desired output:

$$(W^{out})^t = M^{-1}T \tag{3.37}$$

Thus the output weight W^{out} is obtained as the transpose of $(W^{out})^t$.

Testing — The network is trained once the computation of W^{out} is done. The computed W^{out} is then written in the output weights and the network (W, W^{in}, W^{out}) is ready to be tested. The trained network is tested by presenting a novel input sequence $u(n)$ and employing (3.25) and (3.26) as the update equations.

Chapter 4

Known Techniques for Mode Estimation

In this chapter, we briefly describe two stochastic modeling techniques which are known in the literature as solutions to the mode estimation problem of MFR. The first technique is based on syntactic modeling, which employs stochastic grammar. Here a hidden Markov model (HMM) filter is used to estimate the mode of operation of the MFR. Some limitations of this technique were overcome by the second method, the multi-model stochastic approach which is based on observable operator model (OOM). This approach exploits the principle of maximum likelihood to estimate the mode of operation of MFR. This technique also has some limitations of its own, which leads to the development of a more efficient model which is described in Chapter 6. The two stochastic modeling techniques, namely, syntactic modeling and OOM-based modeling are described briefly in the following sections.

4.1 Syntactic Modeling

The syntactic modeling technique [6] is based on formal language processing using grammars for modeling. Grammar is a well-known modeling tool in formal language processing. They help in efficient application of formal language. There are two types of grammar, namely, deterministic grammar and stochastic grammar. In MFR modeling, the radar is assumed to communicate in a formal language, and stochastic grammar is used to model the same.

4.1.1 Formal Language

A formal language L may be defined as follows.

Definition If A is a set of alphabets of some finite length defined by $A = a, b, c, \ldots$ then the language L defined over A is the set of finite length strings formed by concatenating the alphabets of A.

Two kinds of sets can be formed by concatenating the alphabets in A

$$A^+ = a, b, c, ab, bc, ca, aa, bb... \tag{4.1}$$

$$A^* = \varepsilon, a, b, c, ab, bc, ca, aa, bb... \tag{4.2}$$

where A^+ refers to positive closure of A and A^* refers to kleene closure of A as it contains ε. ε is an empty string which contains no alphabets. Formal language by itself has limited application. A better way to employ formal language is by the use of grammar.

A - Alphabet/Set of terminal symbols	$A = \{a, b\}$
δ - Variables/Set of non-terminal symbols	$\delta = \{S_0, S_1\}$
Υ - Production rules	$S_0 \rightarrow aS_1/b$ $S_1 \rightarrow bS_0$
S_0 - Starting non terminal symbol	S_0

Table 4.2: Example for deterministic grammar

4.1.2 Grammar

Grammar may be regarded as a set of rules which may define the use of formal language. There are two types of grammar: deterministic grammar and stochastic grammar. Stochastic grammar is an extension of deterministic grammar as it incorporates a probability distribution which represents the uncertainty contained in the application.

Definition Deterministic grammar G is a four-tuple

$$G = (A, \delta, \Upsilon, S_0) \tag{4.3}$$

where A is the set of a alphabets. δ is the set of non terminal symbols of the grammar. Υ is the finite set of grammatical production rules (syntactic rules). S_0 is the starting non-terminal symbol.

A simple example of deterministic grammar is given below with notations defined in Table 4.2. Based on this grammar the following sequence is generated.

$S_0 \Rightarrow ab$

$S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abab$

$S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abaS_1 \Rightarrow ababS_0 \Rightarrow ababab$

$S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abaS_1 \Rightarrow ababS_0 \Rightarrow ababaS_1 \ldots$

The above described production rule is known as regular grammar. By exploiting the properties of production rules four classes of grammar are defined in a hierarchial fashion in [30]. These four classes of grammar listed below are known in the literature as Chomsky hierarchy of transformational grammar.

1. Regular Grammar
2. Context-Free Grammar
3. Context-Sensitive Grammar
4. Unrestricted Grammar

The syntactic modeling of MRF is rooted in the transformation of context-free grammar. Here, the production rule is defined in the form of $S \longrightarrow \beta$, in which the left hand side of the production rule must contain only one non-terminal symbol whereas the right hand side can be any string. Many practical applications contain some amounts of uncertainty. The uncertainty element may be represented in the form of probabilistic distribution. To illustrate uncertainty, we take the radar signal as an example. Radar signals are typically observed in the noisy environment which may cause sparseness in observation. This leads to an extension of deterministic grammar, which is known as stochastic grammar.

Definition Stochastic grammar G_s is a five tuple

$$G_s = (A, \delta, \Upsilon, P_s, S_0) \tag{4.4}$$

where P_s is the probability distribution over the set of production rules Υ. Rest of the notations have the same meanings as defined earlier. A more detained description of grammar and its various forms may be found in [24].

4.1.3 Finite State Automata (FSA)

Definition A FSA is a five-tuple

$$\Lambda = (Q, I, \delta, S_0, F) \tag{4.5}$$

where Q is the set of states of the FSA. I is the set of input symbols of the FSA. δ is the transition function of the FSA. S_0 is the initial state of FSA. F is the set of final (accepting) states of the FSA ($F \subset Q$).

It is shown that FSA is equivalent to regular languages, regular grammars and regular expressions [6]. Chomsky has shown that finite state language can be generated from CFG if the non-self embedding property is satisfied.

4.1.4 Hidden Markov Model (HMM)

In the syntactic modeling approach mode estimation of MFR is solved by employing a HMM filter. A HMM estimates the underlying state of a model with the help of noisy observations.

Definition A HMM is a three-tuple defined by

$$\lambda = \{A, B, \omega_0\} \tag{4.6}$$

where A is the Markov matrix/state transition matrix. B is the emission probability matrix. ω_0 is the initial state probability vector.

The transition and emission probabilities are generated by the syntactic concepts described in the previous section with probabilities as defined in stochastic grammar. Finally, a HHM filter is used to estimate the mode of the radar in a recursive manner. For a detailed description of HMM and the mode estimation methodology using HMM readers may refer to [6], [23] respectively.

4.1.5 Limitations

The following are the known limitations of this model as described in [3].

- The modeling of MFR is based on the context-free grammar, which needs to satisfy the non-self embedding property in order to generate FSA. Therefore, there is no guarantee that all radar grammars would pass this test.
- If a realistic radar is considered, the number of states in the word level HMM is quite large. Consequently, it becomes more expensive in terms of computational time and memory resource.
- Clustering of the HMM states needs to be performed to estimate the mode of the radar at any given time. This may lead to lack of accuracy in the radar mode estimate
- HMM filtering algorithm is valid for a stationary process. However, an input for the HMM filter is collected from the radar environment which is non-stationary.
- Finally, from the results presented in [6], there exists frequent mode jumps in the HMM estimate, which leads to unreliability and lack of accuracy.

Some of these limitations were overcome by the OOM model described in the following section.

4.2 Observable Operator Model (OOM) based Modeling

OOM is a stochastic modeling tool developed in [25]. It is comparable to the well-known HMM with respect to modeling dynamic systems and stochastic time series. One application example of OOM is to learn the underlying probability distribution of an unknown system based on its training data. In this section, we describe in brief the essence of the OOM and the multi-model approach for mode estimation of MFR. For a detailed description of OOM, readers may refer to [25] [26].

Definition An m-dimensional OOM is a three tuple, $A = (\Re^m, (\tau_a)_{a\in\Sigma}, \omega_0)$, where $\omega_0 \in \Re^m$ and $\tau_a : \Re^m \rightarrow \Re^m$ are linear maps represented by matrices, satisfying

1. $\mathbf{1}\omega_0 = 1$, where $\mathbf{1} = (1,\ldots,1) \in \Re^m$

2. $\mu = \sum_{a\in\Sigma} \tau_a$ has column sum equal to 1, where Σ is the alphabet.

3. $\forall a_0,\ldots,a_r$ it holds that $\mathbf{1}\tau_{\bar{a}}\omega_0 \geq 0$, where $\tau_{\bar{a}} \equiv \tau_{a_r a_{r-1}\ldots a_0} \equiv \tau_{a_r}\tau_{a_{r-1}}\cdots\tau_{a_0}$.

where m is the dimension of the vector space spanned by the prediction function and $\Re^m$ is the domain of the operators. τ_a is the operator indexed over the output symbol "a" of the stochastic process. ω_0 is the initial state vector and $\mathbf{1}\omega_0$ refers to the component sum of ω_0.

4.2.1 Why Observable Operators

Consider a system with state space Σ consisting of two states (A, B). Then the transition from one state to another is known as the trajectory of the system. This is defined by the application of a single operator T. Therefore, the trajectory would be a concatenation of states ....,A,B,A,A,.... visited by the system. In OOM theory, this trajectory is viewed in a complimentary fashion. The states are represented as operators (T_A, T_B). Suppose one of these two operators has to be chosen stochastically, then the trajectory refers to the transition from one operator to another. The trajectory is formed by concatenating the operators $T_A(T_B(T_A(T_A)...))....$ since the observable themselves are the operators the model is named as the observable operator model [26].

4.2.2 Multi-Model Approach

In this section, we briefly describe the algorithm employed to estimate the operational mode of MFR. This multi-model approach [3] makes use of pre-trained OOMs. The radar is assumed to operate within a finite number of modes which are known. Then for each mode of operation an OOM is built. The number of OOMs built is proportional to the number of modes of the radar. These OOMs may be called as mode specific OOMs. Each mode specific OOM is trained with word sequences corresponding to that particular mode of operation. The training is done by the OOM learning principle of Efficiency Sharping. This principle of training OOMs is believed to overcome some known limitations of the expectation maximization algorithm which is used to train HMMs [27].

The mode estimation of MFR is done by computing the likelihood of the incoming words with each of the mode specific OOM model. This is done in a frame-by-frame manner. The likelihood function $(\Omega_j(k))$ for a frame at time (k)

is given as

$$\begin{aligned}\Omega_j(k) &= P(words|model) && (4.7)\\ &= P(z(k)|M_j(k)) && (4.8)\\ &= 1[\tau_{z(k)}\omega_0]_j j = 1,\ldots,r && (4.9)\end{aligned}$$

where $\Omega_j(k)$ represents the likelihood function for a particular frame at time k. $z(k)$ refers to the frame of words at time k. M_j refers to the finite number of modes of MFR with $j = 1,\ldots,r$. τ and ω_0 refer to the operator and initial state vector of the OOM model associated with mode j respectively. Finally, the mode at time k is obtained by maximizing the likelihood function $\Omega_j(k)$. Therefore, the maximum likelihood estimate given by

$$M(k) = arg\max_{M_j} \Omega_j(k) \qquad (4.10)$$

The maximum likelihood estimate makes frequent jumps from one mode to another because of the absence of prior information. This makes the maximum likelihood estimate unreliable. This is overcome by the use of a grid filter, described in the next Chapter. The grid filter incorporates prior knowledge about the radar in the form of transition probabilities. The likelihood estimates of OOM are fed to the grid filter along with the mode transitional probabilities . The output of the grid filter is maximized at each time k to obtain the filtered estimate. The filtered estimate is known as the maximum a posteriori estimate which is free of mode jumps. A more elaborate version of this approach may be found in [3].

4.2.3 Limitations

Though the multi-model OOM approach is said to have overcome some of the limitations [3] of the syntactic modeling technique, it has some limitations of its own. Following are the limitations associated with this multi-model approach.

- The OOM modeling is a multi-model approach, which involves building individual OOM models for each mode of operation. The task of building a model for each mode of operation could be impractical especially when the number of operational modes is high.
- The words from the word extractor module are usually corrupted due to the unfavorable effects of the environment. The word corruption tolerance supported by the multi-model OOM approach is much less than 10% . This places a high demand on the performance of the word extractor module. Therefore, a model with improved word corruption tolerance is needed.
- The OOM multi-model approach fails to detect the mode of operation when the radar dwell time for that particular mode is less than eight phrases [3].
- The latency associated with the MAP estimate is large.

The above limitations of the multi-model approach lead to the design of a more reliable and robust technique for the mode estimation of MFR. This novel estimation technique, described in Chapter 6, overcomes the above mentioned limitations in an efficient manner.

Chapter 5

Introduction to Grid Filters

This chapter offers a brief description of grid filters. The grid filter is a recursive state estimation technique to estimate the state of dynamic systems. Systems in which the state changes over time are called dynamic systems. The grid filter is a Bayesian filter. A recursive Bayesian filter computes the posterior pdf of the state of the system in a recursive fashion by incorporating prior knowledge of the system. In the following sections, we introduce the concept of Bayesian tracking and the algorithm of grid filter.

5.1 Bayesian Approach for State Estimation

In order to analyze and make inference about a dynamic system, two models are required namely,

1. The system model which describes the evolution of the state with time.
2. The measurement model which relates the noisy measurement to the state.

These two models in general may be available in probabilistic form. This probabilistic state space model and the requirement for the updating of information on receipt of new measurement are ideally suited for the Bayesian approach [17]. The main concept of the Bayesian approach lies in the construction of the posterior probability density function (pdf) of the state with all the available information, including the measurements received up to the current time instant. Since the pdf is constructed with all the available statistical information, it is possible to obtain the optimal estimate of the state from the pdf. For these reasons, such a pdf may be said to be a complete solution to the state estimation problem of a dynamic system.

The state of the dynamic system has to be estimated at each time instant. This calls for a recursive type of filtering approach. In the case of recursive filtering, the data are processed in a sequential manner when they are received, so there is no need to store the past history of data. This is realized in two steps, namely, the prediction step and the update step. The prediction step is used to predict the state of the system forward from one measurement time to another by employing the system model. Since the state is usually subject to some random noise the prediction may not be accurate, hence an update step is used to modify the predicted pdf. This update step uses the latest measurement and is achieved by Bayes' theorem, hence the name "Bayesian" approach.

5.2 Bayesian Tracking

Having described the concept of the Bayesian approach for a dynamic state estimation problem, we will now see how the prediction and update equations are developed [17]. In order to describe a dynamic system, consider the following two models.

1. The system model describing the evolution of the state sequence $\{x_k, k \in \mathbb{N}\}$ of a system, is given by

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{5.1}$$

   where $f_k : \Re^{n_x} \times \Re^{n_v} \longrightarrow \Re^{n_x}$ is a non-linear function of x_{k-1} and v_{k-1}. $\{v_{k-1}, k \in \mathbb{N}\}$ is an independent and identically distributed (i.i.d) process noise sequence. n_x & n_v are dimensions of state and process noise vectors, respectively.

2. The measurement model, from which the state of the system is to be estimated recursively, is given by.

$$z_k = h_k(x_k, n_k) \tag{5.2}$$

   where $h_k : \Re^{n_x} \times \Re^{n_n} \longrightarrow \Re^{n_y}$ is possibly a non-linear function. $\{n_k, k \in \mathbb{N}\}$ is an i.i.d measurement noise sequence. n_y & n_n are dimensions of measurement and measurement noise vectors respectively.

The objective of the filter is to estimate the state x_k of the system, given all the available information (measurement) $z_k = \{z_i | i = 1, \ldots, k\}$. From a Bayesian perspective, this requires the calculation of the posterior density function $p(x_k|z_k)$. It is assumed that the initial state pdf $p(x_0|z_0) = p(x_0)$ is given. Then $p(x_k|z_k)$ can be sequentially obtained in two stages: the predict stage and the update stage.

The predict stage uses the system model to predict the state of the system at one time step ahead using past measurements. Assuming that the required pdf $p(x_{k-1}|z_{k-1})$ at time $(k-1)$ is available, prediction is given by the Chapman-Kolmogorov equation as follows:

$$p(x_k|z_{k-1}) = \int_{-\infty}^{\infty} p(x_k|x_{k-1})p(x_{k-1}|z_{k-1})dx_{k-1} \quad (5.3)$$

where $p(x_k|x_{k-1})$ refers to the state transition of the system.

The update stage uses the current measurement z_k to modify the state prediction at time $p(x_k|z_{k-1})$, and this is carried out via Bayes' rule:

$$p(x_k|z_k) = \frac{p(z_k|x_k)p(x_k|z_{k-1})}{p(y_k|z_k)} \quad (5.4)$$

where $p(y_k|z_k) = \int p(z_k|x_k)p(x_k|z_{k-1})dx_k$ is the normalization constant.

The current state estimate $p(x_k|z_k)$ is used as the past estimate in the prediction equation (5.3) for the next time instant $(k+1)$, forming a recursive propagation. This recursive propagation of the posterior probability density function is actually a conceptual solution and cannot be determined analytically. The analytical solution does exist but with certain restrictions. One such solution to this Bayesian recursive filtering is the grid filter, which is described in the next section.

5.3 Grid Filter Algorithm

The grid filter provides optimum solution to the Bayesian recursive equations (5.3) and (5.4) if the following two conditions hold:

- State space model is discrete

- Number of states in the state space is finite

Suppose the state space at time $k-1$ consists of discrete states $x^i_{k-1}, i = 1,\ldots,r$. For each state x_i, let the conditional probability of the state given measurements up to time step $(k-1)$, be denoted by

$$\omega^i_{k-1|k-1} = Pr(x_{k-1} = x^i_{k-1}|z_{k-1}) \tag{5.5}$$

Then the posterior pdf of the state at time $k-1$ is given by

$$p(x_{k-1}|z_{k-1}) = \sum_{i=1}^{r} \omega^i_{k-1|k-1}\delta(x_{k-1} - x^i_{k-1}) \tag{5.6}$$

where $\delta(.)$ is the Dirac delta. Substitution of (5.6) in (5.3) and (5.4) yields the prediction and update equation at time step (k), respectively, as follows:

$$p(x_k|z_{k-1}) = \sum_{i=1}^{r} \omega^i_{k|k-1}\delta(x_k - x^i_k) \tag{5.7}$$

$$p(x_k|z_k) = \sum_{i=1}^{r} \omega^i_{k|k}\delta(x_k - x^i_k) \tag{5.8}$$

where

$$\omega^i_{k|k-1} \simeq \sum_{j=1}^{r} \omega^j_{k-1|k-1} p(x^i_k|x^j_{k-1}) \tag{5.9}$$

$$\omega^i_{k|k} \simeq \frac{\omega^i_{k|k-1} p(z_k|x^i_k)}{\sum_{j=1}^{r} \omega^j_{k|k-1} p(z_k|x^i_k)}, i = 1,\ldots,r \tag{5.10}$$

The above equations assume that $p(x^i_k|x^j_{k-1})$ and $p(z_k|x^i_k)$ are known but they do not constrain the particular form of these discrete densities. This solution of grid filter is optimal only if the above mentioned assumptions hold.

Chapter 6

RNN-DTW hybrid for Mode Estimation of MFR

In this chapter, we formulate the mode-estimation problem and describe our novel methodology to track the operational mode of MFR. This methodology, the RNN-DTW hybrid, employs a RNN trained in a supervised manner with the DTW algorithm as the post processor [28] to estimate the operational mode. The approach is tested for an EW scenario via simulation by employing a hypothetical MFR. The simulation is performed with two types of RNN for comparison, namely the ESN and the RMLP which were described in Chapter 3.

6.1 Problem Formulation

For the RWR to estimate the operational mode, it must first classify the observed radar. In the context of this work, we assume the radar is already classified based on the characteristics of the received signal [15]. Therefore, having classified the

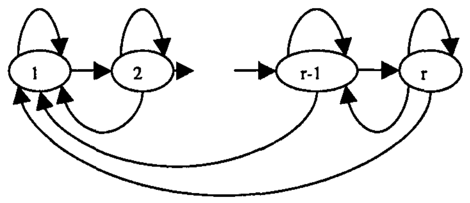

Figure 6.1: Mode Evolution of MFR in a typical first order Markov chain.

radar, the job of the RWR is to estimate its operational mode given the noisy observations and some prior knowledge about the MFR.

Let M_n be the discrete variable, which denotes the mode of the radar at time (n). The mode M_n is said to be in effect from time $(n-1)^+$ right up to time (n). This kind of mode transition is said to be "left continuous" since the mode of operation at time (n) takes effect right from time $(n-1)^+$. Such systems are known as linear transition systems.

The operational modes of MFR are assumed to be finite $M_i,\ i = 1,2,\ldots,r$ in number. Therefore if M_n is the mode at time (n), then

$$M_n \in M_i \tag{6.1}$$

The mode transition of MFR follows a typical first-order Markov process as shown in Figure 6.1. That is, the mode at time (n) depends only on the mode at time $(n-1)$. The mode transition probabilities T_{ij} are assumed to be known and are invariant in time as it is a homogenous Markov chain:

$$T_{ij} \quad = \quad P(M_n = M_j | M_{n-1} = M_i) \tag{6.2}$$

6.2 Methodology

The operational mode of MFR is estimated sequentially in two stages in the following order:

1. Minimum Distance Estimate
2. Maximum a-Posteriori Estimate

The computation of the minimum distance estimate is the most crucial part of the algorithm. A RNN-DTW hybrid is used for this purpose. The maximum a posteriori estimate, which is the second stage of the algorithm, is used to refine the minimum distance estimate. A grid filter is applied for this step. The following subsection describes the two stages in detail.

6.2.1 Minimum Distance Estimate

The pulses received from the radar are de-interleaved and given to a word extractor module. The words from the word extractor are fed to a trained RNN in a sliding window fashion. The words from the word extractor may usually contain erroneous words as a result of various corruption effects. A RNN is setup and trained to learn the underlying word generation mechanism of the MFR. The training is carried out in a supervised manner with the known sequence of words available for each mode of operation. In the simulation, we have tested the algorithm for two types of RNN: the ESN and the RMLP trained with the EKF algorithm. These two networks and their respective training principles were described in Chapter 3.

The output of RNN needs to be processed to obtain the modes. The DTW is used as a post processor. The stream of words from the RNN are fed into a windowing module. Here the word sequences are split into non-overlapping windows of fixed length. The choice of the window size is a design parameter. Mode estimation is carried out in a sequential window fashion with one window at a time. This offers the following advantages:

- Mode estimation by a sequential window fashion helps detect the radar mode transition in a precise manner.
- Sequential window processing helps keep the error growth under control in the case of a mismatched mode estimate at any given time.

Each window from the windowing module is considered as a test sequence T_n (at time n) and is compared with a finite set of reference sequences R_i, $i = 1,2,\ldots,r$ pertaining to each mode. Each reference sequence is a feature vector describing a particular operational mode. It is the result of concatenation of phrases which correspond to that particular mode of operation. The test sequence and reference sequences can be of varying lengths. The choice of their length is a design parameter. However, it is worthy to note that smaller size sequences increase the probability of timely mode estimation and timely detection of mode transition.

The comparison criterion is the distance. The DTW technique described in Chapter 2 estimates the distance of each reference sequence from the test sequence:

$$D_i(T_n,R_i) = min\left[\frac{\sum_{j=1}^{K} W_j}{K}\right]_i \tag{6.3}$$

where W_j refers to the optimal warp path from the initial node to the final node of the grid constructed between the test sequence and each of the reference sequence as described in Chapter 2. This optimal warping minimizes the global distance between the two sequences. Therefore, the minimum distance estimate (M_n^{MD}), which is the mode at time (n), is obtained as

$$M_n^{MD} = \arg \min_{M_1 \le M_i \le M_r} D_i(T_n, R_i) \tag{6.4}$$

6.2.2 Maximum a-Posteriori Estimate

It can be seen from the simulation results that the minimum distance estimate makes occasional mode jumps. This is due to the lack of prior knowledge of the radar. A logical method to overcome this occasional mode jump is by incorporating prior knowledge about the radar into the mode estimation algorithm. This is achieved by applying a grid filter, described in the previous chapter, to the computed minimum distance estimates. The grid filter, which actually smooths the minimum distance estimate, exploits the known transitional probabilities (T_{ij}) of the radar, which is a prior knowledge. The solution of the grid filter is optimal under the assumption that the state space is discrete and finite [17]. The choice of grid filter is motivated by the fact that in our problem the operational mode represents the state of the filter and thus satisfies the assumption. The governing equations of the grid filter modeled in the mode estimation frame-work are given as

$$\omega^i_{n|n-1} = \sum_{j=1}^{r} \omega^j_{n-1|n-1} T_{ij} \tag{6.5}$$

$$\omega^i_{n|n} = \frac{\omega^i_{n|n-1} D_i(T_n, R_i)^{-1}}{\sum_{j=1}^{r} \omega^j_{n|n-1} D_i(T_n, R_i)^{-1}} \tag{6.6}$$

where T_{ij} is the known mode transition probability, which is used as the prior knowledge of the radar. $\omega^{j}_{n-1|n-1}$ is the initial mode probability for each mode of operation. $\omega^{i}_{n|n-1}$ is obtained from the first step of the grid filter, which is (6.5). $D_i(T_n, R_i)$ is the minimum distance estimate obtained from the first step of the mode estimation algorithm. Its inversion in the normalized form gives the probabilistic version of the M^{MD} estimate.

The grid filter provides the posterior estimate of the radar's operational mode. Therefore, the current mode M_n of the radar is obtained as a maximum a-posteriori estimate by maximizing the posterior mode probability and is given by

$$M_n^{MAP} = \arg \max_{M_r \geq M_i \geq M_1} \omega^{i}_{n|n} \tag{6.7}$$

Figure 6.2 illustrates the entire mode estimation methodology. As mentioned earlier, this novel approach is tested via simulation for two different RNN networks. The simulation and results are presented in the next section.

6.3 Simulation

In this section, we explain the computer simulations conducted to test our methodology in the following hypothetical EW scenario. Imagine a MFR stationed on the ground. There are five modes of operation, namely, search, acquisition, non-adaptive track, range resolution and track maintenance that the MFR can perform. Now assume an aircraft fitted with a RWR is approaching the MFR air space. As the aircraft enters the radar detection territory, the radar will begin to engage the aircraft and evolve from one mode to another in a Markovian process as described earlier in Chapter 2. Now it is the job of the RWR to estimate and track the mode

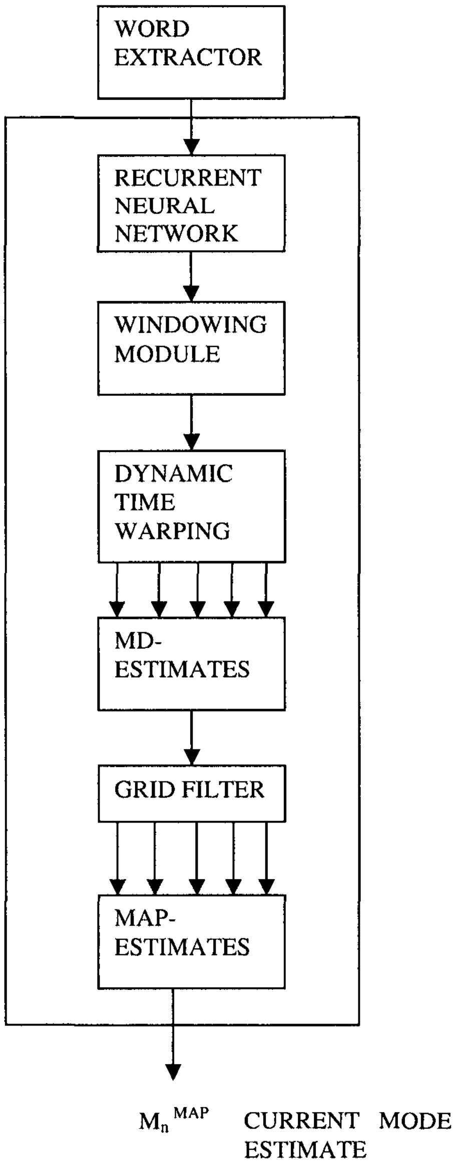

Figure 6.2: Pictorial representation of the entire mode estimation operation.

evolution of the MFR as this information is vital for the aircraft to estimate the threat posed by the radar.

A hypothetical radar was setup, which depicts the above mentioned scenario. For each mode of operation a corresponding phrase from Table 2.1 is assigned. The phrase selection is done randomly by generating random numbers from a uniform distribution of (0 to 1). These phrases are then concatenated to form a sequence of words. This word sequence is then corrupted randomly by introducing mismatched words. This is done to acknowledge the undesirable effects of corruption in the environment and the word extractor module. The sequence of words is fed as input to the pre-trained RNN. Mode estimation is then carried out as described in the previous section. Two thirds of the word sequence are used as the training data and the rest as the test data. The experiment was carried out separately with ESN and RMLP. It is worthy to note while using ESN there are several heuristics [14] to consider as it is not the case with RMLP. However, recent research has shown that it is not difficult to overcome the heuristics associated with ESN [29].

In this experiment, the dimension of the reservoir was 450 units for the ESN, and in the case of RMLP two hidden layers were used with ten recurrent units in the first hidden layer and ten non-recurrent units in the second layer. For each RNN, the simulation was carried out with corruption levels of approximately 10%, 13%, 15% and 18%. The performance results were first obtained for instantaneous trials of each corruption levels since it represents the actual mode estimation scenario for the RWR. Furthermore ensemble averaging of the mode estimate over a large number of trial was also performed for each of the corruption levels to show the robustness of the approach. As there is a delay (latency) associated with the posterior estimate, the delay rate is an important performance criterion. Therefore, the confidence limits of delay rate are also obtained for a large number of instantaneous trials. The performance of the two approaches were found to be comparable and encouraging. The results observed from the simulation graphs are discussed below.

6.3.1 Results

The results obtained via simulation are compared here. The following results compare the algorithm in terms of reliability, accuracy, delay (latency) and computational time between the ESN and RMLP.

- The minimum distance estimate has some mode jumps in both neural network methods. The grid filter estimate is more reliable as it circumvents these mode jumps but at the cost of a delay in detecting the mode transition.
- Both RNN approaches show higher degree of accuracy in the ensemble average estimate, with the mode jump of minimum distance estimate and the delay rate of the maximum a-posteriori estimate being greatly reduced.
- The delay rate for 10% word corruption level averaged over a large number of instantaneous trials is 8.85% in the case of the grid filter based ESN/DTW hybrid and 7.03% in the case of grid filter based RMLP/DTW hybrid as shown in Table 6.3. The table shows the mean delay with 95% confidence limits. It can be seen that there is not much difference in the delay rate between the two methods.
- The word corruption tolerance supported by the RNN-DTW hybrid algorithm is 18%. Previous mode estimation techniques discussed in Chapter

Neural Net Model	mean (μ)	mean confidence interval
ESN/DTW Hybrid	8.84	8.36 - 9.24
RMLP/DTW Hybrid	7.03	6.65 - 7.35

Table 6.3: Delay rate for 10% word corruption averaged over large number of instantaneous trials

4 support word corruption tolerance of less than 10%. This significant increase in the tolerance limit induces leverage in the accuracy of the word extractor module.

- The minimum radar dwell time that can be estimated is 5 phrases in the case of the RNN approach, whereas it is 8 phrases in the case of the OOM approach discussed in Chapter 4.
- The computational time for training and testing in ESN/DTW hybrid is approximately 28s for the posterior estimate as compared to 145s in RMLP/DTW hybrid, on a 1.73 GHz Intel Pentium processor using MATLAB 7.1. This illustrates the practicality of employing ESN for this application.

The Table 6.4 shows the delay rate with 95% confidence limits for each of the corruption levels. The performance graphs for each of the corruption levels are shown below.

Neural Net Model	Corruption Level	mean (μ)	mean confidence interval
ESN/DTW Hybrid	10%	8.84	8.36 - 9.24
RMLP/DTW Hybrid	10%	7.03	6.65 - 7.35
ESN/DTW Hybrid	12%	12.04	11.44 - 12.64
RMLP/DTW Hybrid	12%	12.81	12.17 - 13.45
ESN/DTW Hybrid	15%	13.88	13.19 - 14.57
RMLP/DTW Hybrid	15%	18.30	17.38 - 19.22
ESN/DTW Hybrid	15%	14.21	13.5 - 14.92
RMLP/DTW Hybrid	18%	17.14	16.28 - 18.27

Table 6.4: Delay rate for each of the word corruption levels averaged over large number of instantaneous trials

ESN based mode estimation for 10% corruption level

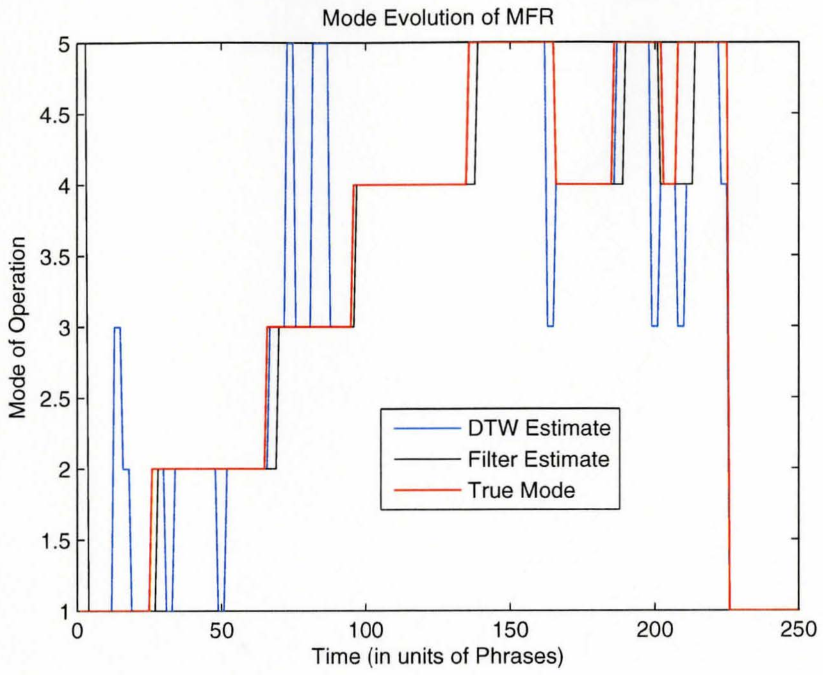

Figure 6.3: ESN based Instantaneous MD & MAP Estimates for 10% corruption level $\forall s$ True Mode Evolution

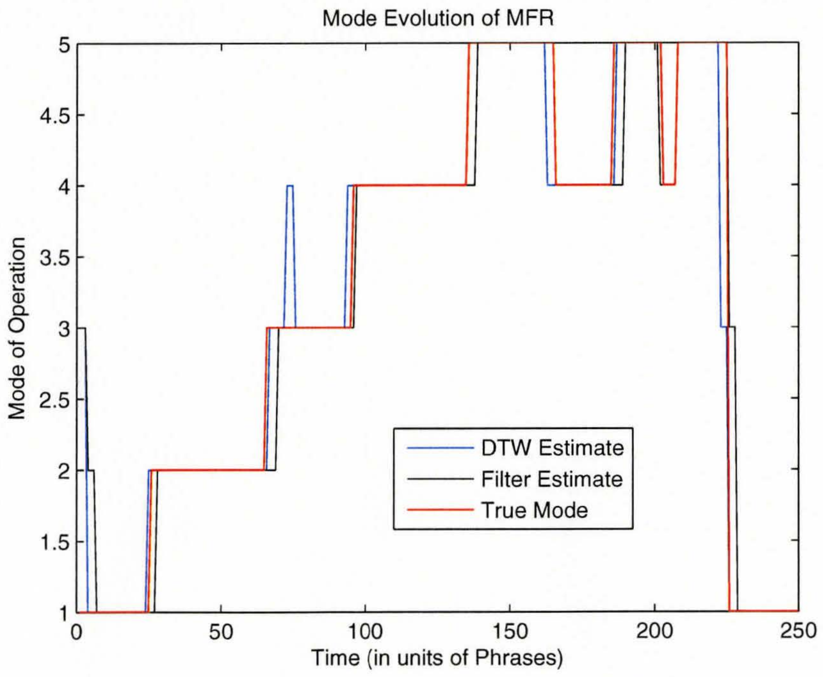

Figure 6.4: ESN based Ensemble Average MD & MAP Estimates for 10% corruption level $\forall s$ True Mode Evolution

RMLP based mode estimation for 10% corruption level

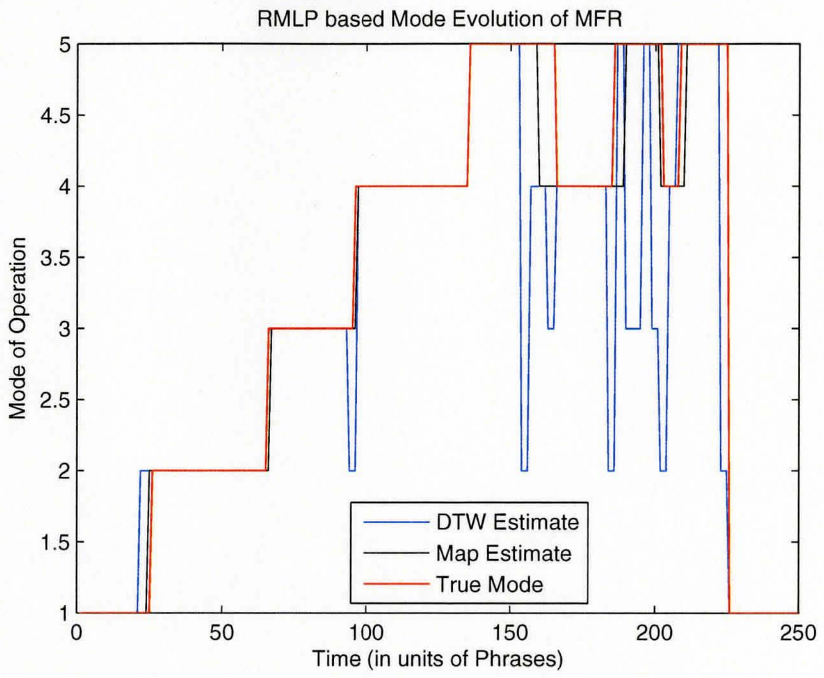

Figure 6.5: RMLP based Instantaneous MD & MAP Estimates for 10% corruption level ∀*s* True Mode Evolution

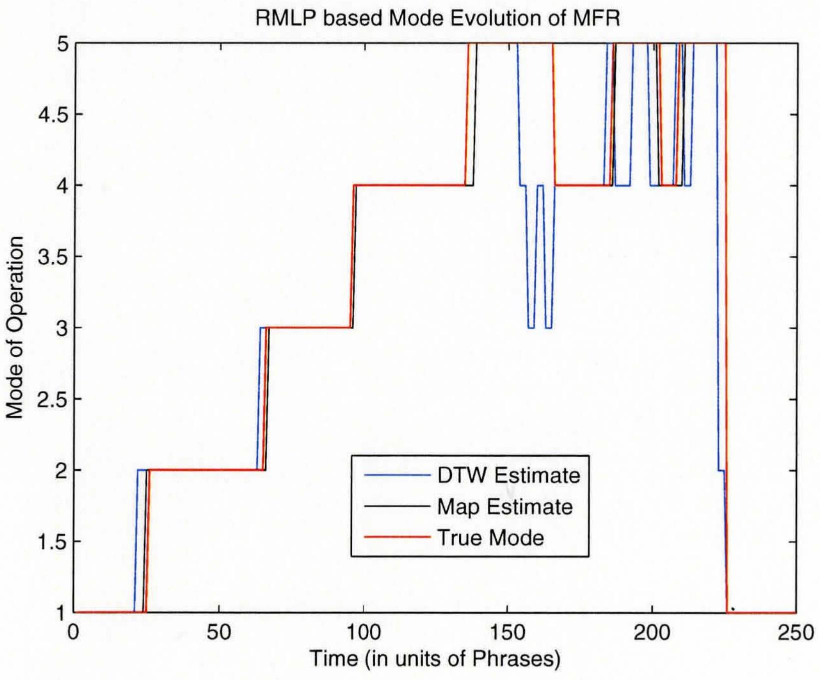

Figure 6.6: RMLP based Ensemble Average MD & MAP Estimates for 10% corruption level ∀*s* True Mode Evolution

ESN based mode estimation for 12% corruption level

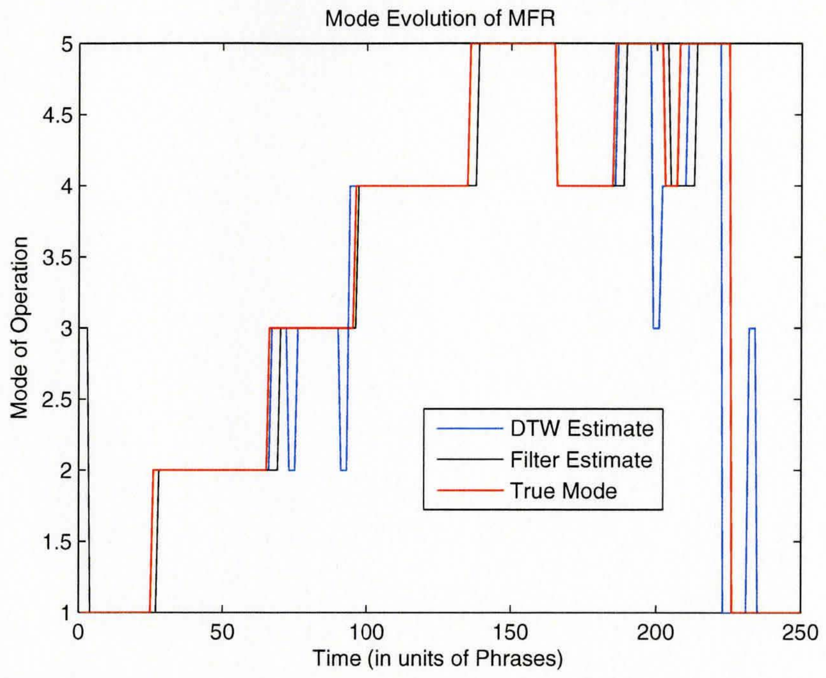

Figure 6.7: ESN based Instantaneous MD & MAP Estimates for 12% corruption level ∀*s* True Mode Evolution

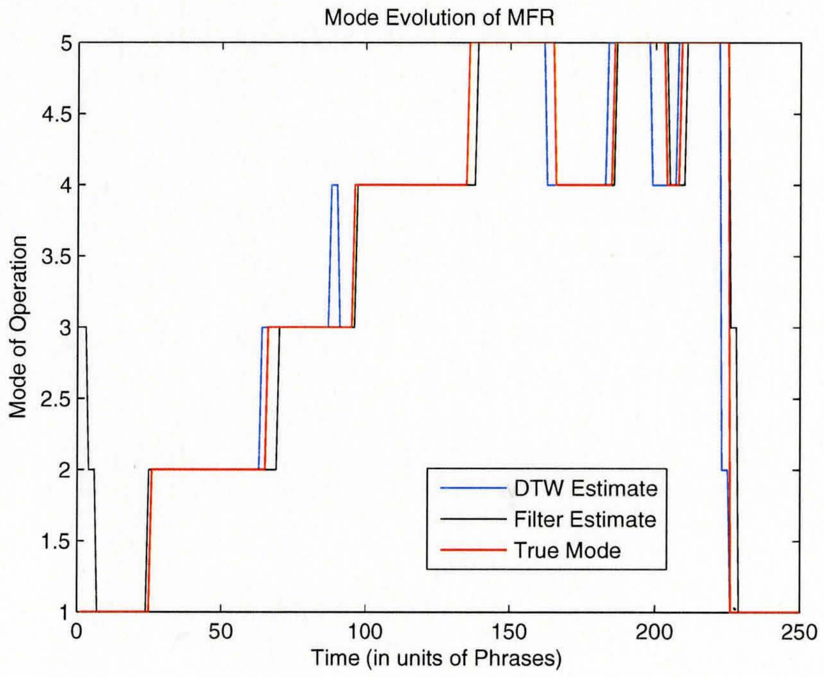

Figure 6.8: ESN based Ensemble Average MD & MAP Estimates for 12% corruption level ∀*s* True Mode Evolution

RMLP based mode estimation for 12% corruption level

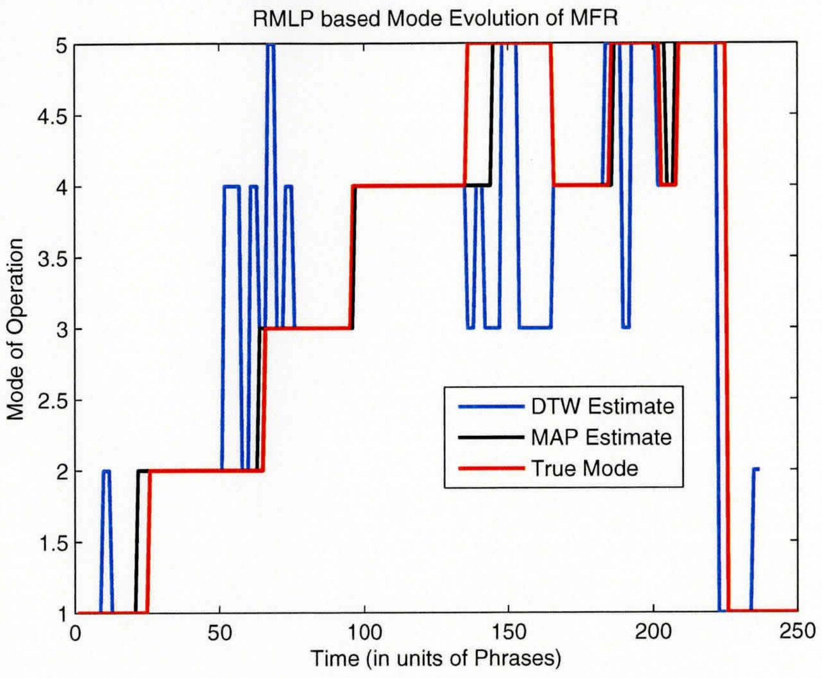

Figure 6.9: RMLP based Instantaneous MD & MAP Estimates for 12% corruption level $\forall s$ True Mode Evolution

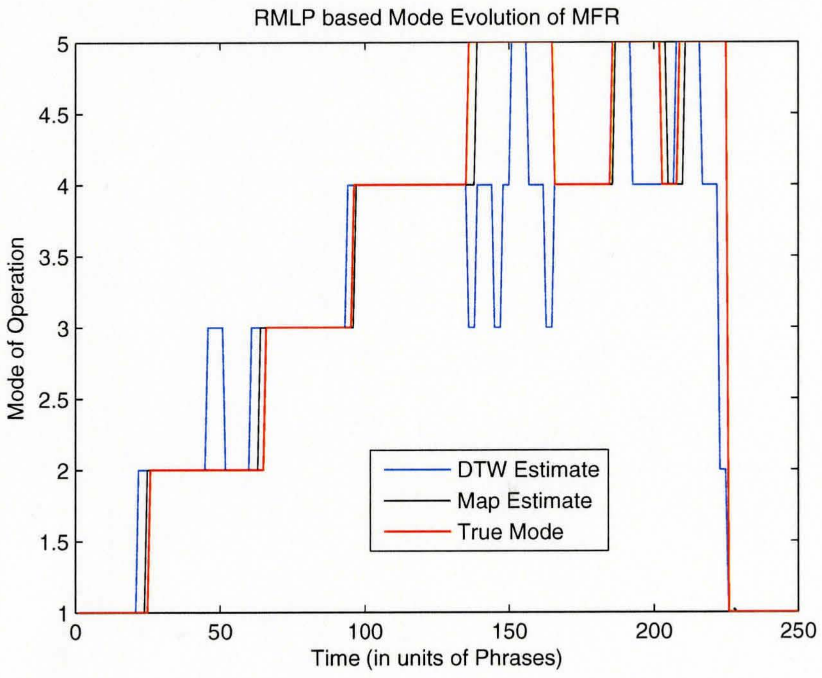

Figure 6.10: RMLP based Ensemble Average MD & MAP Estimates for 12% corruption level $\forall s$ True Mode Evolution

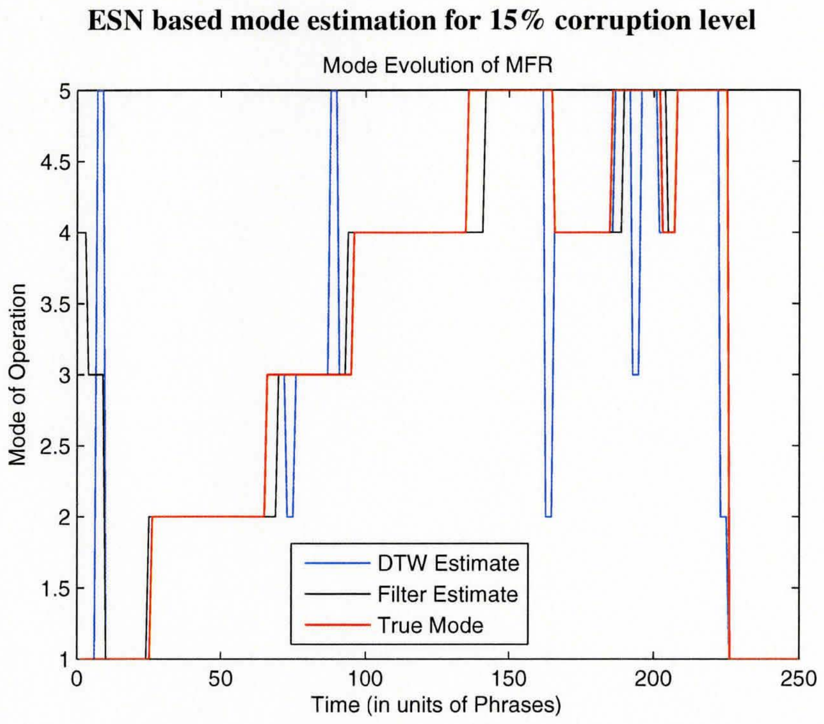

Figure 6.11: ESN based Instantaneous MD & MAP Estimates for 15% corruption level $\forall s$ True Mode Evolution

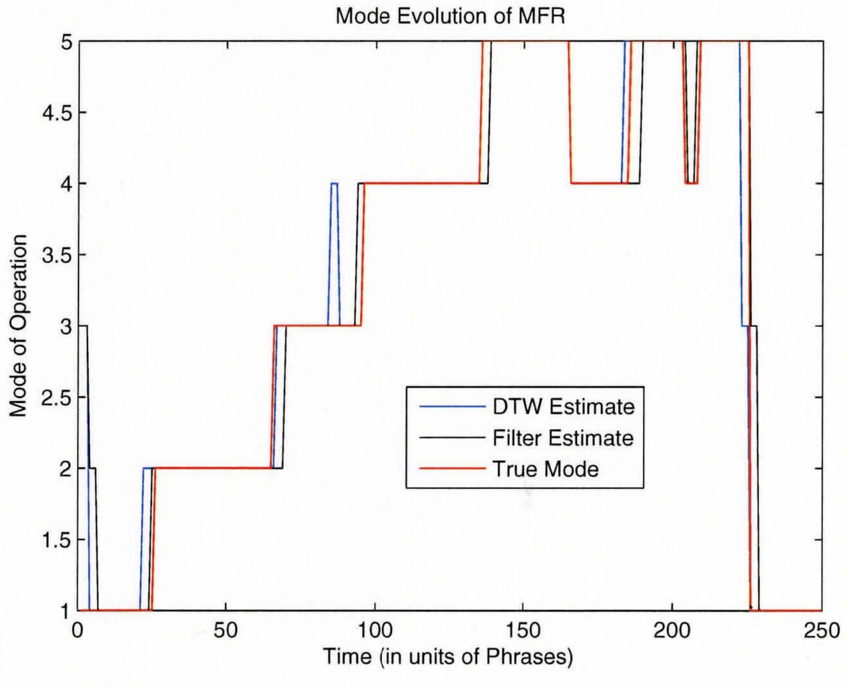

Figure 6.12: ESN based Ensemble Average MD & MAP Estimates for 15% corruption level $\forall s$ True Mode Evolution

RMLP based mode estimation for 15% corruption level

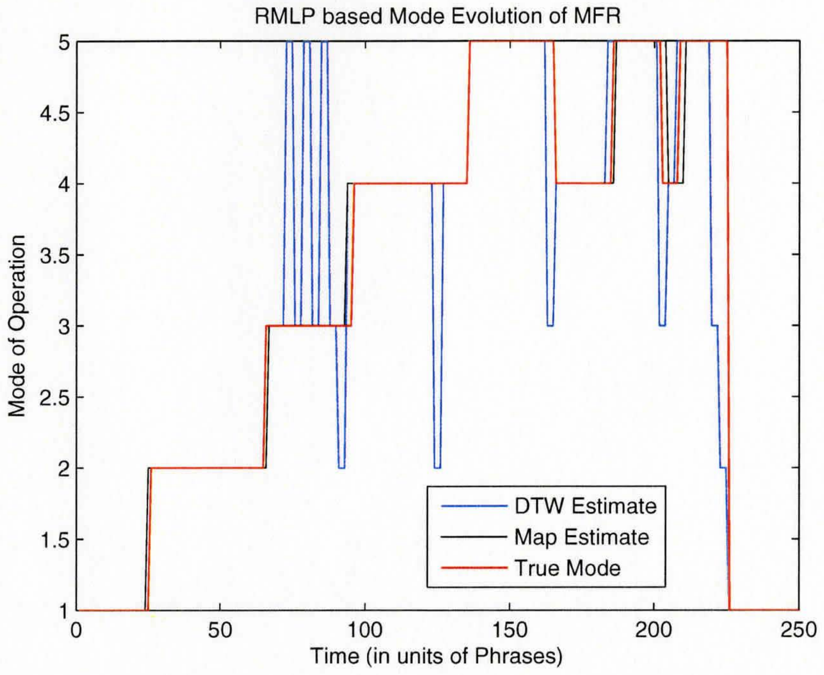

Figure 6.13: RMLP based Instantaneous MD & MAP Estimates for 15% corruption level $\forall s$ True Mode Evolution

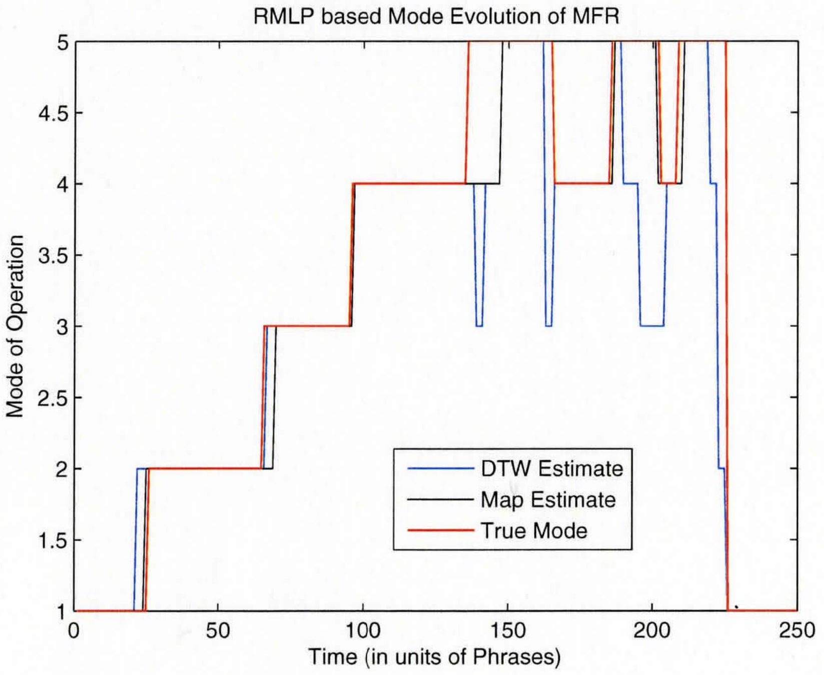

Figure 6.14: RMLP based Ensemble Average MD & MAP Estimates for 15% corruption level $\forall s$ True Mode Evolution

ESN based mode estimation for 18% corruption level

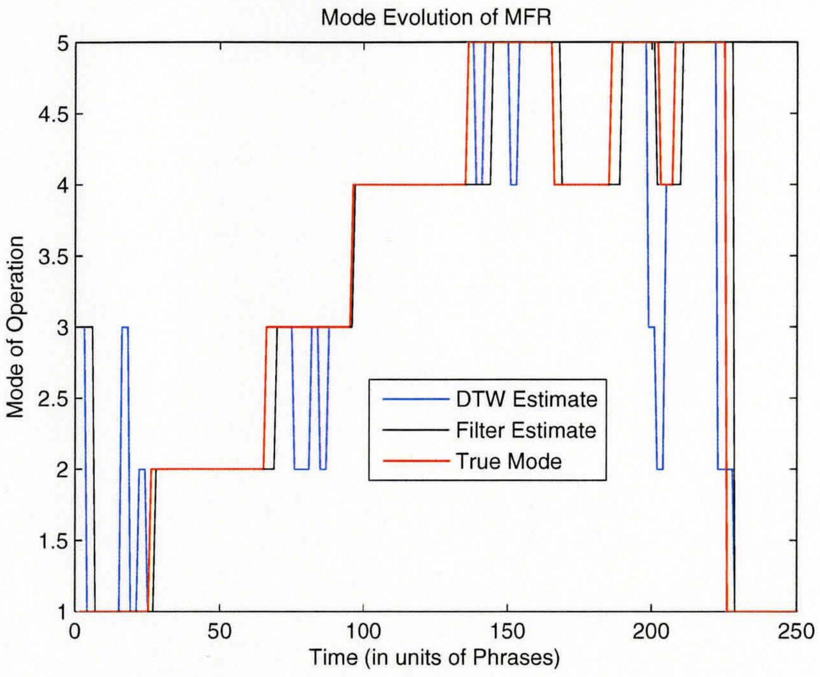

Figure 6.15: ESN based Instantaneous MD & MAP Estimates for 18% corruption level $\forall s$ True Mode Evolution

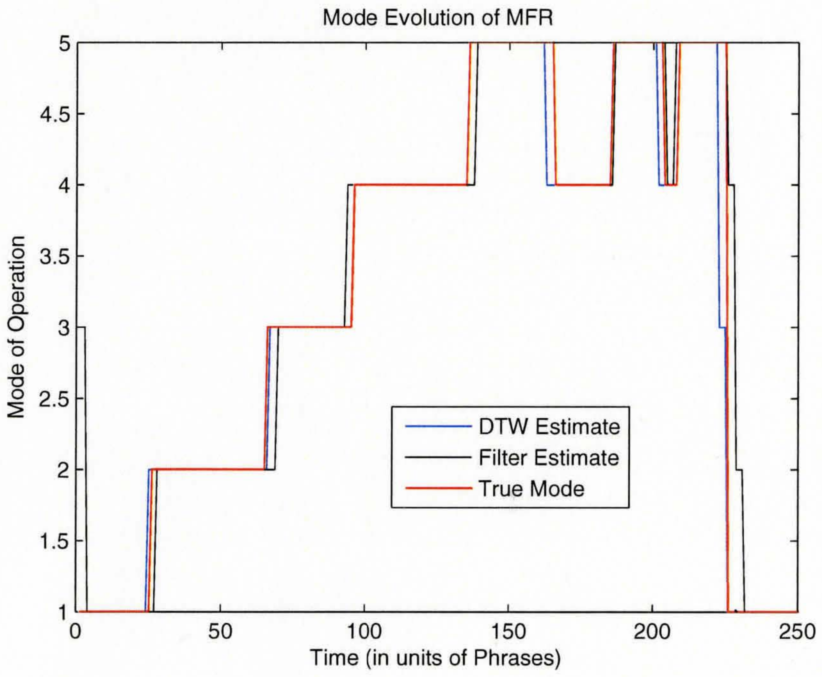

Figure 6.16: ESN based Ensemble Average MD & MAP Estimates for 18% corruption level $\forall s$ True Mode Evolution

RMLP based mode estimation for 18% corruption level

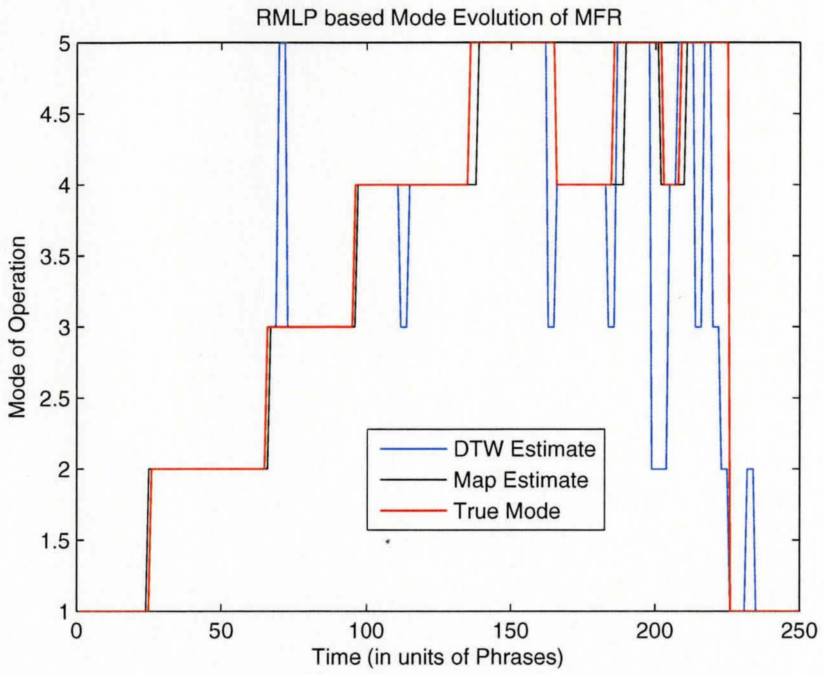

Figure 6.17: RMLP based Instantaneous MD & MAP Estimates for 18% corruption level $\forall s$ True Mode Evolution

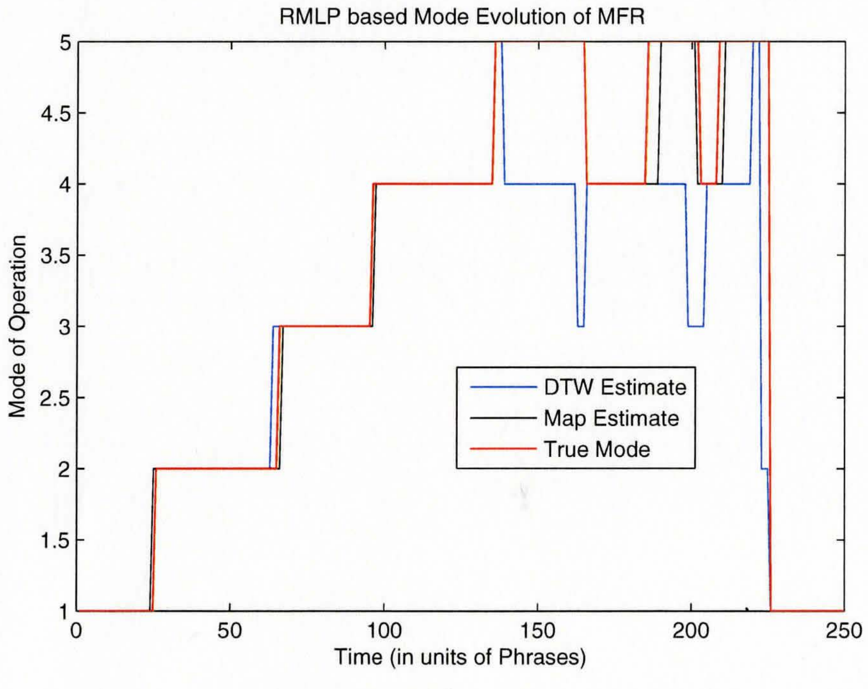

Figure 6.18: RMLP based Ensemble Average MD & MAP Estimates for 18% corruption level $\forall s$ True Mode Evolution

Chapter 7

Discussion

The mode-estimation problem of MFR has been studied in detail in this thesis. This chapter presents conclusions by summarizing the most important contributions and results of the investigation.

7.1 Major Contributions

In this section, we aim to summarize the important contributions of this research effort.

- **RNN for Modeling** — RNNs are a type of artificial neural networks that are characterized by the use of feedback. They are dynamically driven networks capable of mimicking dynamic systems with arbitrary precision. The presence of feedback helps the RNN to learn more efficiently than other learning models. The use of RNN for mode estimation has offered specific benefits over previous modeling techniques in terms of accuracy, latency, dwell time detection and word corruption tolerance. Supervised training of the ESN and training of the RMLP by the EKF algorithm, which is also supervised, are efficient in capturing the word generation mechanism of the radar. This subsequently helps in estimating the operational mode of MFR accurately.

- **Single Model Approach** — A single model approach is proposed to track the operational mode of MFR. The mode estimation of MFR is achieved by building a single RNN model to learn the underlying word generation mechanism of the MFR through supervised training. A single model approach overcomes the burden of building individual models for each operational mode of MFR, which would prove to be a cumbersome task if the number of operational modes is large. Thus a single model approach is more practical to solve the MFR problem.

- **Word Level Estimation** — Since the MFR employs complex pulse structure with the capability to perform multiple tasks, mode estimation becomes very difficult at the pulse level. To overcome this difficulty and to keep the signal processing complexity manageable, a hierarchial signal structure is proposed, in which individual words extracted from the MFR pulses are used for mode estimation. Mode estimation carried out in the word level, which is subsequent to the pulse level, is less computationally intensive and erroneous than pulse level processing.

- **Improved Performance** — The RNN-DTW hybrid presented in this investigation has demonstrated improved performance over previous approaches in the mode estimation of MFR. The significant performance improvements include higher degree of reliability and accuracy, reduced latency rates, improved dwell time detection and most importantly higher word corruption tolerance. The word corruption tolerance has been increased to 18% in RNN-DTW hybrid from below 10% supported by previous techniques. Computer simulations show that the RNN-DTW hybrid has not only overcome the limitations associated with the previous techniques, but has also achieved it with comparable computational intensity.

7.2 Conclusion

This thesis presents a novel method, the RNN-DTW hybrid to estimate and track the mode of operation of MFR accurately. Since the electromagnetic emissions of the radar are very complex, a hierarchial signal architecture is proposed to keep the signal processing complexity manageable. The mode estimation is then performed at a higher level of the hierarchial signal architecture, known as the word level.

The proposed methodology was tested for two RNNs, namely, the ESN and RMLP. The RNNs are trained in a supervised manner to capture the underlying word generation mechanism of the MFR. DTW is used as the post processor for both RNN models. Minimum distance is the decision criterion used to estimate the mode of operation. The phenomenon of occasional mode jumps of the minimum distance estimate is mitigated by updating the probability of each operational mode in a recursive fashion. This is achieved by using the knowledge of the distance estimates and the prior mode transition probabilities in a grid filter framework. Finally, the jump free estimate of the current operational mode is obtained in the MAP sense by maximizing the mode probability at that particular time instant.

Computer simulations conducted for both instantaneous and Monte Carlo trials show that the RNN approach performs better than the previous mode estimation techniques with increased corruption tolerance, reduced latency rates, improved dwell time detection and comparable run times. Among the two RNNs, the results show that the performance of grid based ESN/DTW hybrid is much similar to its RMLP counterpart in all the above mentioned performance criterion. However, in terms of computation time the performance of the former is only about one fifth of the latter. Since computation time may be an important parameter in the threat analysis of MFR, it is considered to be the deciding factor between the two RNN models. Therefore, we conclude that grid based ESN/DTW hybrid is a suitable candidate for tracking the operational mode of the MFR, both in terms of performance and computational complexity.

Bibliography

[1] C. Schleher, *Introduction to Electronic Warfare*, Artech House, UK, 1986.

[2] M. I. Skolnik, *Introduction to Radar Systems*, McGraw-Hill, NY, 2002.

[3] I. Arasaratham, "Tracking the Mode of Operation of Multi-Function Radar," Master's Thesis, Dept. of Electrical and Computer Eng., McMaster University, Canada, 2006.

[4] R. G. Wiley, *Electronic Intelligence: The Analysis of Radar Signals*, Artech House, UK, 1993.

[5] R. G. Wiley, *Electronic Intelligence: Interception of Radar Signals*, Artech House, UK, 1985.

[6] N. Visnevski, "Syntactic Modeling of Multi Function Radar", PhD Dissertation, Dept. of Electrical and Computer Eng., McMaster University, Canada, 2005.

[7] N. Visnevski, S. Haykin, V. Krishnamurthy, F. Dilkes, and P. Lavoie, "Hidden Markov Model for Radar Pulse Train Analysis in Electronic

Warfare", *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, Philadelphia, PA, Mar. 2005.

[8] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, USA, 2003.

[9] S. Salvador and P. Chan, "Fast DTW: Towards Accurate Dynamic Time Warping in Linear Time and Space", *KDD Workshop on Mining Temporal and Sequential Data*, Dept. of Computer Sciences, Florida Institute of Technology, FL, 2004.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, NJ, 1999.

[11] H. Jaeger and H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication", Science, Apr. 2004, pp. 78-80.

[12] H. Jaeger,"A Tutorial on Training Recurrent Neural Networks, Covering BPTT, RTRL, EKF and the Echo State Network Approach", ***, 2004.

[13] H. Jaeger,"Adaptive Nonlinear System Identification with Echo State Networks" *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer (Eds), MIT Press, Cambridge, MA, 2003, pp. 593-600.

[14] H. Jaeger, "The Echo State Approach to Analysing and Training Recurrent Neural Networks", GMD Report, German National Research Center for Information Technology, 2001.

[15] J. Matuszewski and L. Paradowski,"The Knowledge-Based Approach for Emitter Identification", *International Conference on Microwaves and Radar*, MIKON, 1998, pp. 810-814.

[16] P. J. Werbos, "Backpropagation Through Time: What it does and how to do it", *Proceedings of the IEEE*, Vol. 78, Issue 10, Oct. 1990, pp. 1550-1560.

[17] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Applications*, Artech House, UK, 2004.

[18] G. V. Puskorius and L. A. Feldkamp, "Decoupled Extended Kalman Filter Training of Feedforward Layered Network", *International Joint Conference on Neural Networks*, Vol. 1, 1991, pp. 771-777.

[19] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks", *IEEE Transactions on Neural Networks*, Vol. 5, 1994, pp. 279-297.

[20] R. J. Williams and J. Peng, "An Efficient Gradient Based Algorithm for Online Training of Recurrent Network Trajectories", *Neural Computations*, Vol. 2, 1990, pp. 490-501.

[21] S. Singhal and L. Wu, "Training Feed-Forward Networks with Extended Kalman Filter", *IEEE International Confrence for Acoustics, Speech and Signal Processing*, Glasgow, Scotland, 1989, pp. 1187-1190.

[22] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", *Transactions of ASME, Journal of Basic Engineering*, Vol. 82, 1960, pp. 35-45.

[23] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, 77(2), Feb. 1989, pp. 257-286.

[24] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, MA, 2002.

[25] H. Jaeger, "Discrete-Time, Discrete-Valued Observable Operator Models: A Tutorial", Tech. Report, GMD, Sankt Augustin, 1998.

[26] H. Jaeger, "Observable Operator Models for Discrete Stochastic Time Series", *Neural Computation*, No. 6, 2000, pp. 1371-1398.

[27] H. Jaeger, M. Zhao, K. Krettzhmar, T. Obesrtein, D. Popovinci, and A. Kolling, "Learning Observable Operator Models via the ES Algorithm", *New Directions in Statistical Signal Processing: from Systems to Brains*, (eds. S. Haykin, J. C. Principe, T. Sejnowski, and J. McWhirter), MIT Press, Cambridge, MA, 2006.

[28] Y. Bengio, *Neural Network for Speech and Sequence Recognition*, International Thomson Publishing, 1995.

[29] Y. Xue, L. Yang, and S. Haykin, "Decoupled Echo State Network with Lateral Inhibition", *Elsevier Neural Networks*, 20(3), Apr. 2007, pp. 365-376.

[30] N. Chomsky, "On Certain Properties of Grammars, Information and Control", Vol. 2, Issue 2, Jun. 1959, pp. 137-167.