MULTIPLE CLASSIFIER SYSTEMS FOR A GENERIC MISSILE WARNER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KUBİLAY BAŞIBÜYÜK

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2009

Approval of the thesis:

MULTIPLE CLASSIFIER SYSTEMS FOR A GENERIC MISSILE WARNER

submitted by KUBİLAY BAŞIBÜYÜK in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Mustafa Kuzuoğlu _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Mübeccel Demirekler _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Mustafa Kuzuoğlu _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Tolga Çiloğlu _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Çağatay Candan _____
Electrical and Electronics Engineering Dept., METU

Selçuk Avan (M. Sc.) _____
ASELSAN

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Kubilay Başıbüyük

Signature :

ABSTRACT

MULTIPLE CLASSIFIER SYSTEMS FOR A GENERIC MISSLE WARNER

Başıbüyük, Kubilay

M. Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Mustafa Kuzuoğlu

June 2009, 111 Pages

A generic missile warner decision algorithm for airborne platforms with an emphasis on multiple classifier systems is proposed within the scope of this thesis.

For developing the algorithm, simulation data are utilized. The simulation data are created in order to cover a wide range of real-life scenarios and for this purpose a scenario creation methodology is proposed. The scenarios are simulated by a generic missile warner simulator and tracked object data for each scenario are produced.

Various feature extraction techniques are applied to the output data of the scenarios and feature sets are generated. Feature sets are examined by using various statistical methods. The performance of selected multiple classifier systems are evaluated for all feature sets and experimental results are presented.

Keywords: Multiple Classifier Systems, Feature Extraction, Pattern Recognition

ÖZ

JENERİK BİR FÜZE İKAZ SİSTEMİ İÇİN ÇOKLU SINIFLANDIRICI SİSTEMLERİ

Başıbüyük, Kubilay

Yüksek Lisans, Elektrik Elektronik Mühendisligi Bölümü
Tez Yöneticisi: Prof. Dr. Mustafa Kuzuoğlu

Haziran 2009, 111 sayfa

Bu tez kapsamında hava platformları için kullanılabilecek jenerik bir füze ikaz sistemi algoritması önerilmektedir.

Algoritmayı geliştirmek için sentetik benzetim senaryoları hazırlanmıştır. Sentetik senaryolar hazırlanırken gerçek hayatta karşılaşılabilecek senaryoları en iyi şekilde temsil edebilme yeteneğine sahip bir veri kümesinin hazırlanmasına gayret edilmiş ve bu amaçla kullanılmak üzere bir yöntem önerilmiştir. Senaryolar jenerik füze ikaz simülatörü tarafından işlendikten sonra, senaryo içindeki takip edilmiş nesne verilerini içeren çıktılar oluşturulmuştur.

Çeşitli öznitelik çıkarım yöntemleri senaryo çıktı verilerine uygulanmış ve öznitelik kümeleri oluşturulmuştur. Bu öznitelik kümeleri, çeşitli istatistiksel yöntemler kullanılarak incelenmiştir. Çoklu sınıflandırıcı sistemlerinin performansları, oluşturulan tüm öznitelik kümeleri için değerlendirilmiş ve deneysel çalışmaların sonuçları özet bir şekilde sunulmuştur.

Anahtar Kelimeler: Çoklu Sınıflandırıcı Sistemleri, Örüntü Tanıma, Öznitelik Çıkarma

To My Parents,

My Brother

and

Tuna

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Prof. Dr. Mustafa Kuzuoğlu for his encouragements, guidance, advice, criticism and insight throughout the research.

I would like to thank ASELSAN Inc. for facilities provided for the completion of this thesis and TUBITAK-BIDEB for the financial support they provided during my graduate study.

I would like to forward my appreciation to all my friends and colleagues who contributed to my thesis with their continuous encouragement and friendships.

I am very grateful to Tuna for her love, patience and understanding.

Finally and mostly, I would like to thank to my parents and my brother, Çağatay. Without their love and support I would not have been able to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GMW	: Generic Missile Warner
LF	: Linear Fit
FA	: False Alarm
FT	: Fourier Transform
MCS	: Multiple Classifier System
MPOD	: Mean Probability of Declaration
MRT	: Mean Reaction Time
MW	: Missile Warner
ROC	: Receiver Operating Characteristics
TNFA	: Total Number of False Alarms

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Missile Warner (MW) can be defined as a system capable of detecting and declaring the missile threats against the platform on which it is installed. MW systems play a vital role in helping the survivability of the platforms in hostile environments. If early warning about the threat may be reported to a counter measure system by the MW, appropriate countermeasure methods can be applied and the probability of survival of the platform can be increased dramatically. The decision making algorithm is one of the key components of the system to increase the effectiveness of the MW. The main contribution of this thesis is the construction of a decision making algorithm based on multiple classifier systems for the use of a Generic Missile Warner (GMW) for airborne platforms.

Contemporary MW systems mainly operate in infrared and ultraviolet spectrum and they can be considered as electro-optical systems. The infrared spectrum has the disadvantage of having a highly dense background clutter whereas in the ultraviolet spectrum the background is very low compared to the infrared spectrum. Within the context of this thesis, it is assumed that the MW system makes use of the ultraviolet spectrum.

The MW should detect the radiation emitted from the threat, track the source of the radiation and classify the tracked entity by applying a decision making algorithm. Although the background clutter in the ultraviolet domain is relatively low, threats are not the only signals which are detected by the MW. In the band of the

frequencies occupied by the ultraviolet spectrum, many other sources which emit radiation exist, like fires, city lights and industrial facilities. These sources will be named as false alarms throughout the thesis and the main focus of this thesis is on proposing a feature extraction and classification system to discriminate the false alarms and threats.

Designing a classification system for a GMW involves the joint optimization of three performance measures; namely probability of declaration, reaction time and number of false alarms. Probability of declaration indicates how successfully the GMW declares the threats. Reaction time is defined as the interval beginning from the detection of source to the declaration of the object as a threat. False alarms can be defined as the non-threatening sources which the GMW classifies as threats.

It is worth to mention that, these measures are not equally important. Probability of declaration is the most vital performance measure. This measure should not be sacrificed for low false alarm rates or smaller reaction times. The second most important measure is the reaction time, which is desired to be as low as possible. False alarm is the measure that has the least significance among all. However false alarm rate should be kept at minimum in order to increase the reliability of the system.

In the following section the methodology for creating and testing a GMW algorithm is explained.

1.2 Methodology

In order to develop an algorithm for a Generic Missile Warner (GMW) system, firstly a data set should be created. The data set is composed of entities called scenarios.

A *scenario* can be defined as a setup consisting of a threat or a false alarm source, an atmosphere model and a platform carrying the GMW. In each scenario we have only one signal source that can be detected by the GMW which is labeled as a false alarm or a threat. In the threat scenarios, the signal source moves by simulating the

kinematic model of the missile and in the false alarm scenarios the signal source is stationary. The intensity characteristics of the signal source is time varying for both false alarm scenarios and threat scenarios. The details of the scenario generation process are explained in Section 2.1. The scenarios are simulated by using the simulation environment of ASELSAN Inc.

After scenario generation step, *feature extraction* methods are applied to the output data of the scenario simulator. In this thesis, two feature extraction methods are used, namely linear model fitting and Fourier Transform. The feature extraction process is repeated for each generated scenario output and a second data set consisting of feature sets are generated.

By using the features extracted, the performance of the Multiple Classifier Systems including feature grouping, bagging and boosting are examined. Two performance evaluation methods are used. Firstly, feature-based evaluation is used and secondly a scenario-based evaluation is used. The scenario-based evaluation is only applied for boosting.

The general overview of the methodology is depicted in Figure 1-1.

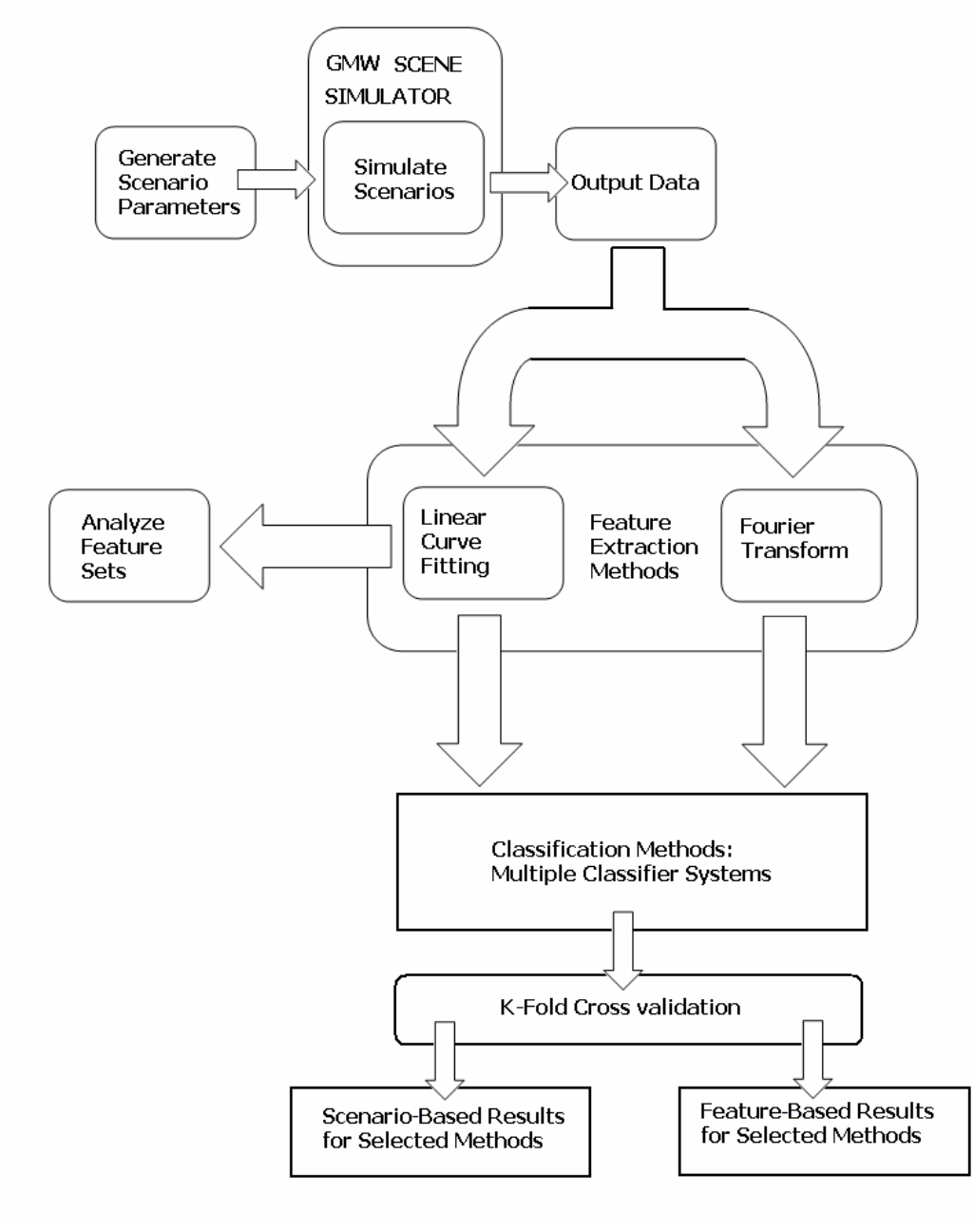

Figure 1-1: Overview of methodology

1.3 Organization of the Thesis

The thesis is organized as follows:

In **Chapter 2**, scenario generation methodology is explained, basic information about the GMW simulator is provided and a method for creating multiple scenarios is proposed.

In **Chapter 3**, feature extraction methods are explained and a method for extracting information from the GMW simulator's output is proposed. The feature extraction methods used within the content of this thesis are linear fit and Fourier transform.

In **Chapter 4**, statistical properties of the data sets are examined. Self organizing maps are used to visualize the data sets.

In **Chapter 5**, theoretical background about Multiple Classifier Systems (MCS) is provided. The design criteria for MCS are explained and various existing MCS methods are examined.

In **Chapter 6**, experimental results are provided for selected MCS methods including bagging, boosting and feature grouping.

In **Chapter 7**, a summary about the thesis and its results are given and possible future works are suggested.

In **Appendix – A**, scenario parameters are listed.

In **Appendix – B**, mean and standard deviation values for the data sets are given.

In **Appendix – C**, histogram plots regarding the feature sets can be found.

In **Appendix – D**, correlation matrices of the feature sets are given.

CHAPTER 2

DATA SET GENERATION

2.1 General Information

In many classification problems, the characteristic of the data is one of the most important aspects that should be considered when designing the pattern recognition system. The data set plays the leading role in determining the feature extraction and classification methods. The system designer should collect adequate data to represent the characteristics of the environment and choose the best methods of feature extraction and classification to maximize the overall system performance.

For a GMW (Generic Missile Warner), collecting real world data is essential for both designing and evaluating the accuracy of the system. However, due to the fact that this task involves missile firings in a controlled environment and excessive hours of false alarm recordings, generation of real world data is difficult and expensive.

In this study, synthetic data are created for training and test purposes of the GMW algorithm. The simulation environment of ASELSAN Inc. is utilized in order to generate the data set that will be used throughout the study.

The sections given below briefly discuss the process of creating scenarios, the content of the scenarios and the analysis of the created data.

2.2 The Process of Creating Scenarios

The scenarios are created using GMW simulator software. GMW software takes the scenario parameters which are defined in Section 2.2.1 as inputs and outputs the features of the object for the given scenario.

Since the main concern of this work is not on modeling of GMW systems, only basic information about the simulator is provided.

The GMW simulator software is mainly composed of three parts:

1. A camera model which maps the scenario information into digital objects
2. Kinematic models for the moving objects
3. Atmosphere modeling

The output of the GMW software is a list of features belonging to the objects detected and tracked by the GMW on the platform. Details of the output data are given in Section 2.2.3.

2.2.1 Scenario Parameters

The classification problem investigated in this thesis involves deciding between two alternative classes, which will be called as “threats”, and “false-alarms”. Threat scenarios will be abbreviated as “**t-scenarios**” and false alarm scenarios will be abbreviated as “**f-scenarios**”.

“**Threat**” scenarios are mainly missile engagement scenarios in which the platform carrying the GMW encounters an incoming missile.

“False alarm” scenarios are the scenarios, which the system should not classify the sources which are detected as a threat.

For the purpose of covering a wide range of the real world engagement scenarios, the simulations are performed in a parameterized way. These parameters allow simulations to be fit on a grid of sampled possible real world scenarios.

A suitably-defined rectangular coordinate system is used in defining all t and f scenarios. In Figure 2-1 a generic scenario setup is presented. The parameters of the scenario are indicated on the figure.

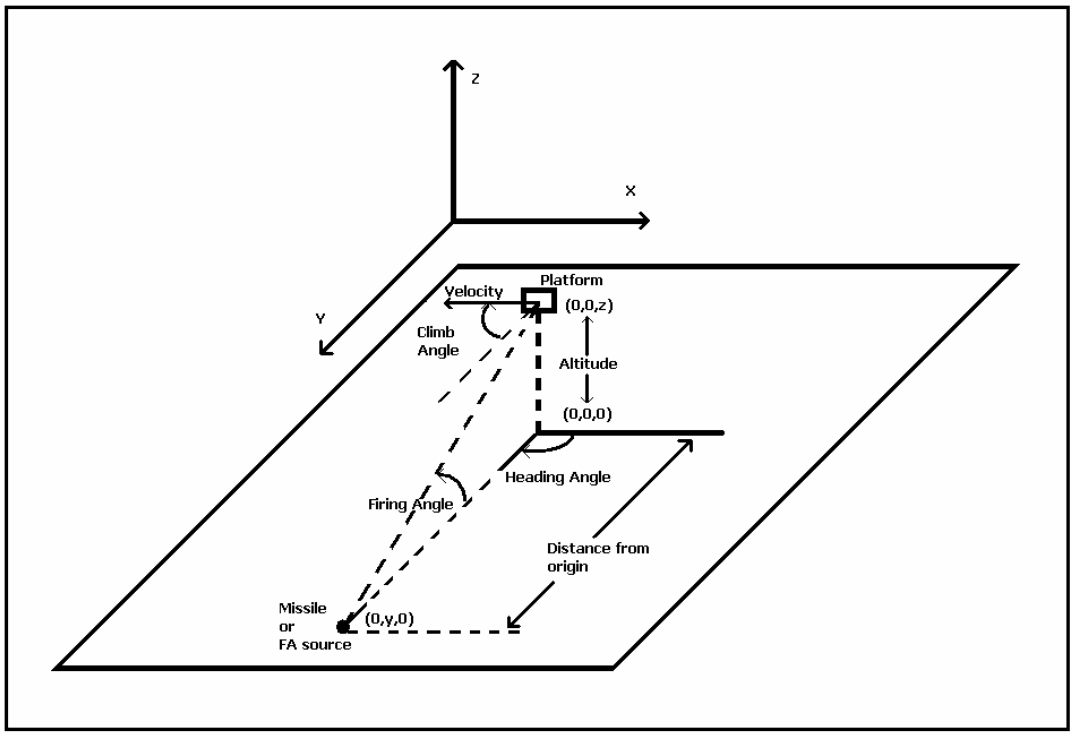

Figure 2-1: Scenario Setup

As it can be seen from the figure, the platform is located at the reference point of the x-y plane with an altitude of z. The variables seen on the figure with the addition of atmospheric parameters and the source specific parameters are considered as the main parameters that define a scenario. Firing angle is automatically calculated by using altitude and distance from origin parameters.

The parameters of the t-scenarios and f-scenarios are divided into three main categories, namely aircraft, threat and atmosphere specific parameters. In Table 2-1 the complete set of the scenario parameters are listed.

Table 2-1: Scenario parameters

AIRCRAFT (FOR ALL SCENARIOS)		
Parameter Name	***Definition***	***Units***
Altitude	Altitude of the platform	m
Velocity	Velocity of the platform	m / s
Heading Angle	Heading of the platform	degrees
Climb Angle	Climb angle of the platform	degrees
MISSILE (FOR T-SCENARIOS ONLY)		
Parameter Name	***Definition***	***Units***
Missile Type	Type of the missile (I-II-III)	-
Position of the missile on the x-y plane	Position of the missile relative to platform	m
Firing Angle[1]	Firing angle of the missile	degrees
FALSE ALARM (FOR F-SCENARIOS ONLY)		
Parameter Name	***Definition***	***Units***
FA Source Type	Type of the FA (I-II-III)	-
Position of the FA source on the x-y plane	Position of the FA source relative to the platform	m
ATMOSPHERE (FOR ALL SCENARIOS)		
Parameter Name	***Definition***	***Units***
Atmospheric Attenuation	Parameter characterizing the atmospheric loss	km^{-1}

2.2.2 Processing Steps of a Scenario

A scenario starts with an initial condition for the missile or the false alarm source and the platform. In all scenarios, the platform starts at the origin of the x-y plane of the coordinate system with a scenario specific altitude and the missile is aimed at

[1] This is not a user defined parameter, it is calculated automatically for a scenario

the platform. The position and type of the missile or the false alarm is determined by the scenario parameters.

A *platform* can be defined as a moving body with an initial speed, heading angle and climb angle. The kinematic model of the platform is very simple. The platform moves with a constant velocity and moves in a linear fashion in the direction of the climb and heading angles.

The *missiles* are characterized by their radiation emission levels over time and their kinematics. The GMW simulator software can simulate these behaviors of the missile models. For this study, three different kinds of generic missiles are used, each of which has a unique kinematic and emission model.

The *false alarm sources* are characterized by their radiation emission levels over time. Since these entities are stationary, no kinematic model is used.

The *atmosphere* is modeled to reflect the absorption characteristics of the environment. The radiation emitted from the missile propagates through the atmosphere and considerably attenuates before it reaches the platform. In order to simplify the scenario generation process, atmospheric attenuation is modeled with a single parameter (Table 2-1). Larger values of the atmospheric attenuation coefficient indicate strong attenuation of the signal. This parameter must be non-negative.

The irradiance coming from the missile or the false alarm source is detected by the camera model. The camera model tracks the source of the incoming irradiation and outputs the values mentioned in Table 2-2. The camera model is assumed to provide a full spherical coverage for the platform.

The GMW simulator works in discrete-time with fixed time steps. At each time step the following operations are performed:

1. Calculate the position of the platform by taking into account the position of the platform in the previous time step.

2. Calculate the position of the missile by taking into account the kinematic model of the missile and the position of the missile in the previous time step.

3. Calculate the irradiance level of the missile by using the radiation emission profile of the missile.

4. Irradiance emitted by the missile is propagated through the atmosphere to find the irradiance at the platform location.

5. Construct the object seen by the camera model based on the irradiance level incident on the platform

6. Calculate the output data of the camera model.

These steps are repeated until the scenario ends and an output data set consisting of the tracked properties of the object is formed. The workflow of the GMW simulator software for one time step is depicted in Figure 2-2.

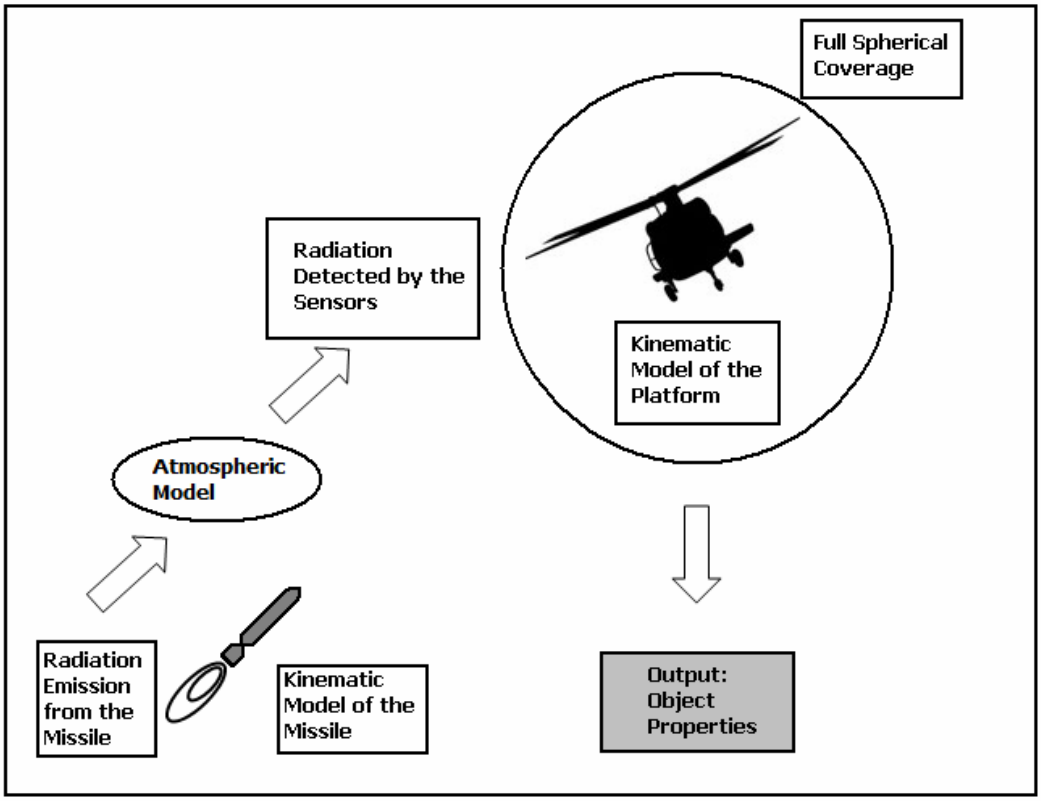

Figure 2-2: The workflow of the GMW simulator software for one time step

2.2.3 Output Data

An entity detected by the GMW simulator is called an object. Detection time for a scenario is determined by the camera model of the GMW, the atmospheric attenuation and the emission level of the missile. If an object is detected by the GMW, it is tracked and temporal characteristics of the object are presented as the output, until the object is out of sight or the scenario is finished. The output data generated by the GMW simulator has three main components:

1. **Position:** The angular position of the object with respect to the body of the platform in terms of azimuth and elevation.

2. **Area:** The number of pixels illuminated on the digital image of the object

3. **Digital level:** The average digital level of the object that is projected on the GMW.

Owing to the fact that the GMW simulator samples the scenario at defined time-steps, the output data is a list of the above-mentioned components. To put it in another way, the GMW simulator outputs the temporal data of the tracked object, which are angular position, area and digital level of the object. In Table 2-2 the output data of the GMW simulator software is listed.

Table 2-2: Sample output data of the GMW simulator software

Frame Number	Azimuth	Elevation	Area	Digital Level
1	142.49	4	46	88
2	142.49	4	51	90
3	142.48	3.2	56	88
4	142.48	3.2	57	98
5	142.48	3.2	66	104
6	143.25	3.19	81	96
7	142.48	3.2	105	96
8	142.48	3.2	142	82
9	143.25	3.19	194	91
10	143.25	3.19	312	82

The feature extraction process will be carried out on these data. The temporal correlation of the fields Azimuth, Elevation, Area and Digital Level will be exploited to get the information needed to classify the tracks. The details of the feature extraction process are presented in Section 3.3

2.3 Content of the Created Scenarios

In this section, a method to generate multiple scenarios is proposed. Scenarios are generated by using the combinations of the parameters described in Section 2.2.1. The scenario generation methodology is explained in Section 2.3.1.

2.3.1 Scenario Generation Methodology

As mentioned before, a scenario is composed of the parameters listed in Table 2-1. Technically speaking, any combination of these parameters will result in a scenario. However, the parameters must have values between specific ranges in order to claim that the scenarios are realistic. To exemplify this, consider a scenario having an atmospheric attenuation value equal to one hundred. This would be unreal because such an extinction coefficient cannot occur in real life.

For a better simulation of reality, the parameter values should be chosen within realistic limits and scenarios should cover a wide range of real world cases. In order to achieve this goal the following methodology is applied when creating scenarios:

1. Sample each parameter from a uniform distribution. The parameter type determines the limits of the distribution. The parameter values and their limits are listed below (Table 2-3):

Table 2-3: Limit of the parameter values

	Minimum Value	Maximum Value	Units
Altitude	200	1000	m
Velocity	0	80	m / s
Heading Angle	-180	180	degrees
Climb Angle	-20	20	degrees
Position of the missile/false alarm on the x-axis	-3000	3000	m
Position of the missile/false alarm on the y-axis	-3000	3000	m
Atmospheric Attenuation	0.7	1.5	km^{-1}

2. Run the scenario for each missile type and false alarm source type. Since the simulation studies are carried out on three types of missiles and three types of false alarms, each physical scenario setup is executed six times.

3. Repeat this step until the desired number of scenarios are created.

If we consider the parameter space as a whole, this method allows us to sample from the parameter space uniformly within the given limits owing to the fact that the individual parameters are assumed to be independent of each other.

With the aim of preventing unrealistic scenarios, firing angle is limited between 10 and 45 degrees. The firing angle (θ_f) is calculated by using parameters z (altitude), m_x (position of the missile on the x-axis) and m_y (the position of the missile on the y-axis) (Equation 2-1):

$$\theta_f = \arctan\left(\frac{z}{\sqrt{m_x^2 + m_y^2}}\right) \tag{2-1}$$

2.3.2 Distribution of Scenario Parameters

Since the parameters of the scenarios are sampled randomly, it is worth examining their statistical behavior. The histograms of the parameters altitude, velocity, heading angle, climb angle and atmospheric attenuation are shown in Figure 2-3. The missile positions are shown as a scatter plot on the x-y plane. Although firing angle is not a scenario parameter, it is also shown in because it summarizes the geometrical structure of the scenario.

The altitude parameter does not show a uniform behavior. This stems from the fact that the firing angle is limited between 10 and 45 degrees. By imposing a constraint on the firing angle, the altitude and the position of the missile on the x-y plane become dependent parameters. This dependency shows itself not only in the distribution of the altitude but also in the scatter plot indicating the position. The neighborhood of the origin of the scatter plot indicating the position is not crowded.

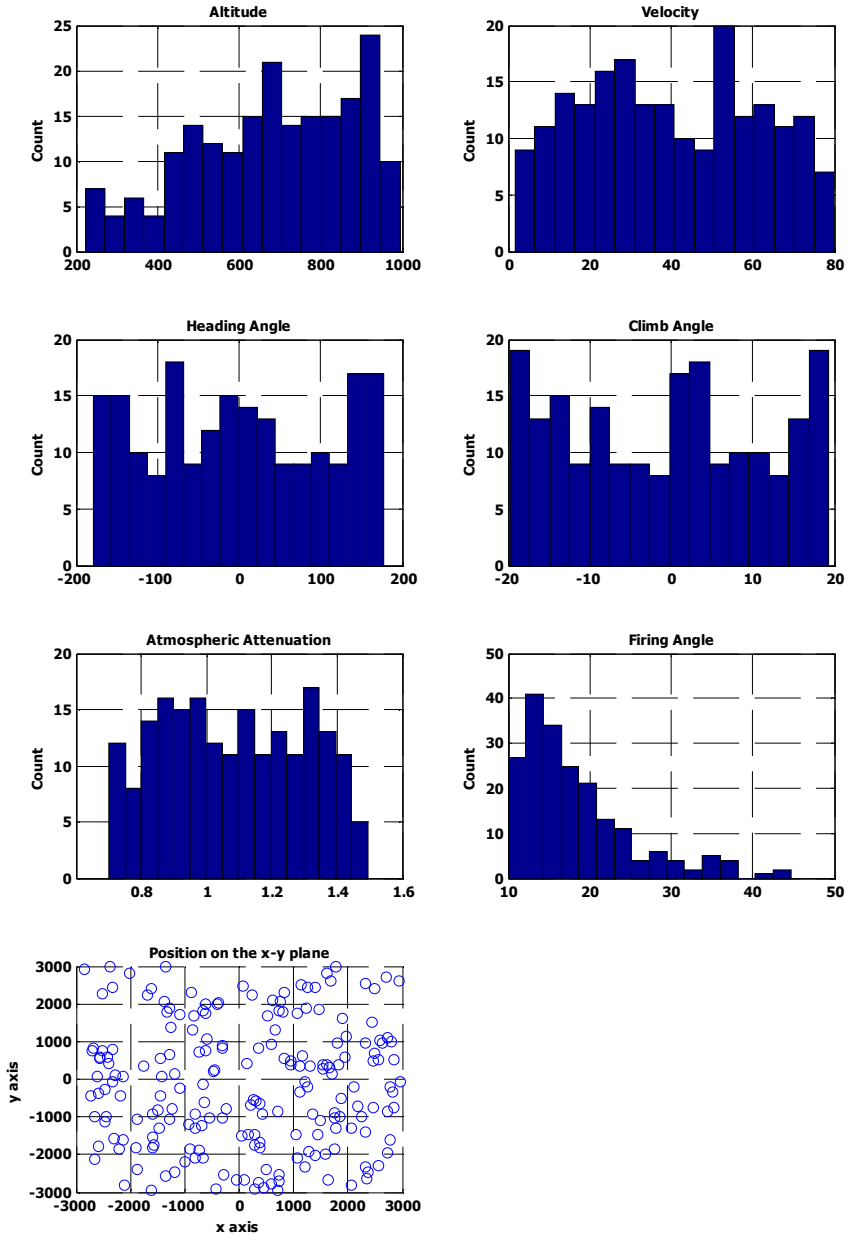

Figure 2-3: Distribution of scenario parameters

2.3.3 Quantitative Information about Scenarios

In this work a total of two hundred physical scenario setups are created using GMW simulator. Each scenario setup is used with three types of threats and three types of false alarms. By this way, a data set of six hundred threats and six hundred false alarms is produced.

For each element of the data set there exists an output data set whose format is given in Section 2.2.3.

The detailed list of the scenario setups can be found in Appendix - A.

CHAPTER 3

FEATURE EXTRACTION

3.1 General Information

The main aim of a pattern recognition system is to assign labels to the samples described by a set of features. In order to achieve this task, the system should have at least two essential capabilities:

1. Feature Extraction

2. Classification

Feature extraction is the process of producing valuable information from data for the use of the classifier. Generally, designing the feature extraction part of a pattern recognition system requires the accumulation of knowledge about the domain [1]. Alternatively, there exist various methods to extract features from the data set by using statistical tools without using any explicit knowledge about the domain of the problem [2].

In this thesis, the challenge is to correctly classify a threat oncoming to a platform and to keep the false alarm rate at a minimum level. For the feature extraction part of this problem a modeling approach is utilized. The output data mentioned in Section 2.2.3 is processed to reflect the time characteristic of the data. The modeling approach includes the use of linear data fit and the Fourier transform.

3.2 Definitions

Before going into the details of the feature extraction process, some definitions are given. The problem we are dealing with might be considered as a classification problem represented by a time series. A scenario is defined as a number of consecutive observations. In our case we have two kinds of scenarios, namely T and F. T represents threat scenarios, and F represents false-alarm scenarios.

$$T = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{bmatrix} \tag{3-1}$$

$$F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix} \tag{3-2}$$

Each observation at specific time instant in a T or F scenario is a k-dimensional vector:

$$T_t = [x_{t1}\, x_{t2} \ldots x_{tk}] \tag{3-3}$$

$$F_t = [x_{t1}\, x_{t2} \ldots x_{tk}] \qquad 1 \le t \le N \tag{3-4}$$

If each observation within a scenario is represented by a row vector, we get a matrix of size *N x k,* where *N* is the duration of the scenario and *k* is the number of time series. In other words, we can represent each T-scenario or F-scenario by *k* distinct time-series. In our case there exists four time series for a scenario. We can express a scenario in the following format:

$$T = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & x_{N4} \end{bmatrix} \tag{3-5}$$

$$F = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N1} & x_{N1} & x_{N1} \end{bmatrix} \tag{3-6}$$

It must be noticed that each scenario may have a different duration, however all scenarios contain the same time-series, namely azimuth, elevation, area and digital level. The duration of a scenario is mainly determined by the total observation time during which the object is tracked by the camera model.

Each of these time series gives information about a particular characteristic of the object tracked by the simulator. For convenience these characteristics are listed in Table 3-1.

Table 3-1: Time series in a scenario

Parameter	Definition
Azimuth	Position of the object in azimuth
Elevation	Position of the object in elevation
Area	Area of the object
Digital level	Integer number between 0 and 255

3.3 Feature Extraction Methods

In this section methods that will be used in the process of feature extraction are discussed.

Most of the proposed time-series classification methods in the literature depend on high level representations, which include Fourier Transforms, Wavelet Transforms, Piecewise Linear Representations, Regression Tree Representations, and Symbolic Representations [3].

Two methods of feature extraction will be examined within this thesis.

The first method mainly depends on parameterization of the behavior of the time series by using the linear data fit approach. The other method makes use of the Fourier transform as a feature extraction approach.

3.3.1 Modeling - Linear Fit

For a GMW, it is essential to make the decisions for the threat and false alarm classes quickly and correctly. Also, the system must have the capability of dynamic decision making so that new information about the object could be used in the classification process.

In a dynamic classification process, the information about the objects will be coming one after the other at a rate depending on the sampling rate of the GMW. Taking into account the fact that the information of the tracker in a single frame could not be used solely, a windowing approach should be applied to the data and this window should be moved with each frame.

Assume that we have a window of size W. To start the feature extraction process, a simple linear polynomial fit is applied to each time series within this window. For the sake of simplicity "*azimuth, elevation, area and digital level*" are abbreviated as *az, el, ar* and *dl* respectively in the equations below.

$$f_{az\_fit}(t) = a_{az\_fit}t + b_{az\_fit} \quad (3\text{-}7)$$

$$f_{el\_fit}(t) = a_{el\_fit}t + b_{el\_fit} \quad (3\text{-}8)$$

$$f_{ar\_fit}(t) = a_{ar\_fit}t + b_{ar\_fit} \quad (3\text{-}9)$$

$$f_{dl\_fit}(t) = a_{dl\_fit}t + b_{dl\_fit} \quad (3\text{-}10)$$

The following features are extracted for each function by using the fit parameters and statistical values about the fitting process [4].

Table 3-2: Features extracted from the linear fit

Number	Feature	Definition	Equation
1	*a*	1st coefficient of the line	3-17
2	*b*	2nd coefficient of the line	3-18
3	*se(a)*	Standard error for "*a*"	3-20
4	*se(b)*	Standard error for "*b*"	3-21
5	*f(t)*	Fit value of the line at time t[2]	3-7..3-10[3]
6	*rmse*	Root mean square error of the fit	3-23
7	r^2	Square of multiple correlation coefficient	3-22

Since there are four different time series, the linear fit operation will be applied to each of them and a total of twenty eight features will be extracted from a time window *W*.

[2] "t" indicates the time at the end of the sliding window.
[3] For each time series the corresponding equation is used.

In order to give the mathematical definitions of the features, assume that we have a set of N data points in the two-dimensional plane, $\{y_i, t_i\}$. We are trying to fit a line to these data points in the least square sense:

$$f(a,b,t) = at + b \tag{3-11}$$

In order to calculate a and b, we define the following parameters:

$$\bar{t} = \frac{1}{N}\sum_{i=1}^{N}(t_i) \tag{3-12}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N}(y_i) \tag{3-13}$$

$$S_{tt} = \sum_{i=1}^{N}(t_i - \bar{t})^2 \tag{3-14}$$

$$S_{yy} = \sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{3-15}$$

$$S_{ty} = \sum_{i=1}^{N}(t_i - \bar{t})(y_i - \bar{y}) \tag{3-16}$$

Then the parameters a and b can be calculated as [5]:

$$a = \frac{S_{ty}}{S_{tt}} \tag{3-17}$$

$$b = \bar{y} - a\bar{t} \tag{3-18}$$

In order to calculate the standard errors for parameters a and b, we will use the following definition:

$$S = \sqrt{\frac{S_{yy} - \frac{S^2_{ty}}{S_{tt}}}{N-2}} \tag{3-19}$$

Using 3-14, 3-15, 3-16 and 3-19 [5], the standard error for *a* and *b* is calculated as :

$$se(a) = \frac{S}{\sqrt{S_{tt}}} \tag{3-20}$$

$$se(b) = S\sqrt{\frac{1}{N} + \frac{(\bar{t})^2}{S_{tt}}} \tag{3-21}$$

The multiple correlation coefficient and root mean square error is calculated as follows [5]:

$$r^2 = \frac{S_{ty}^{\ 2}}{S_{tt}S_{yy}} \tag{3-22}$$

$$rmse = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - (at_i + b))^2} \tag{3-23}$$

A sliding window approach will be used to extract the linear fit coefficients. With each newly coming frame the window *W* is shifted to include the new information and exclude the oldest one. Throughout this thesis, *W* is chosen to be 20. The feature extraction process terminates when the end of the time series is reached. The methodology is explained in detail in Section 3.3.2.

The logic behind using the parameters of a line fit is to represent the changing behavior of the time series. For example the first feature in Table 3-2 gives information about the general trend of the time series, a negative slope indicating a decrease in the values of the time series and a positive slope indicating an increase.

As another example, consider the standard errors of the first two parameters. These values give information about how reliable the first two features are.

In Figure 3-1 an example related to the line fitting procedure is presented. Notice that the time series corresponding to the digital level is presented in the figure and the extracted features for the current window are also shown.

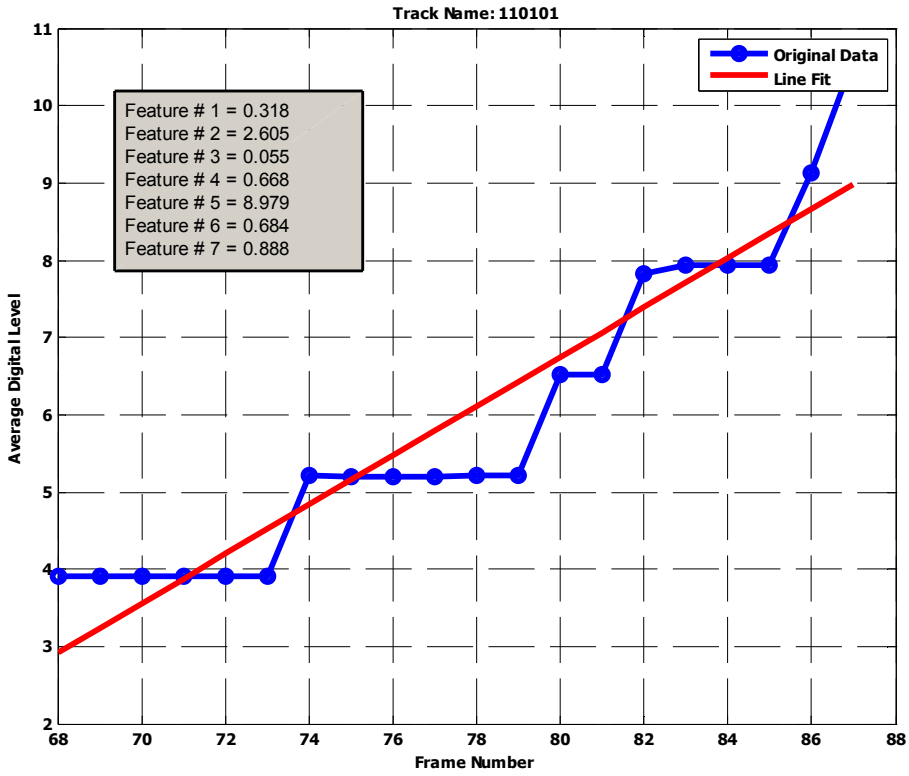

Figure 3-1: Line fitting example

3.3.2 Modeling - Fourier Transform

Fourier transform can be used to generate a suitable approximation of a time series as a weighted sum of some basis functions. The basis functions of the Fourier transform are cosine and sine functions.

Assume that the time series that we are dealing with consists of W discrete data points. Then the discrete time Fourier transform for this data can be defined as below:

$$f(n) = a_0 + \sum_{k=1}^{W/2}[a_k \cos(\frac{2\pi kn}{W}) + b_k \sin(\frac{2\pi kn}{W})] \tag{3-24}$$

The coefficients a_k and b_k can be found by using the orthogonality principle. Basically, the orthogonality principle is used to take the inner product of the basis vectors with the data and normalizing the result in order to find the corresponding coefficients. The Fourier coefficients are found as:

$$a_k = \frac{2}{W}\sum_{n=1}^{W}[f(n)\cos(\frac{2\pi kn}{W})],\ k = 1...W/2 - 1 \tag{3-25}$$

$$a_0 = \frac{1}{W}\sum_{n=1}^{W}[f(n)] \tag{3-26}$$

$$a_{W/2} = \frac{1}{W}\sum_{n=1}^{W}[f(n)\cos(\pi n)] \tag{3-27}$$

$$b_k = \frac{2}{W}\sum_{n=1}^{W}[f(n)\sin(\frac{2\pi kn}{W})],\ k = 1...W/2 - 1 \tag{3-28}$$

The basis functions for the Fourier transform are shown in

Figure 3-2 and Figure 3-3 for the frequencies corresponding to k=1, 2 and 3. When we are working in discrete-time, these functions will be samples taken at the defined time steps.

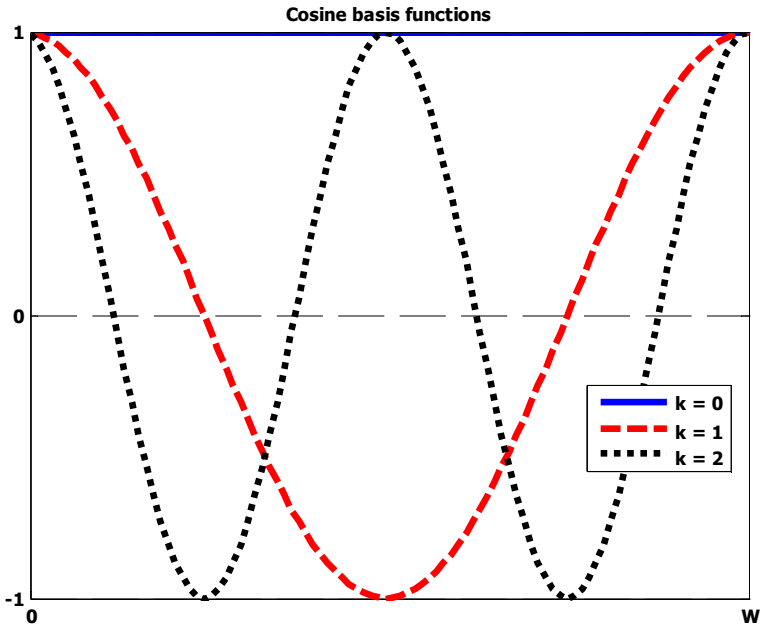

Figure 3-2: Cosine basis functions for k=1, 2, 3

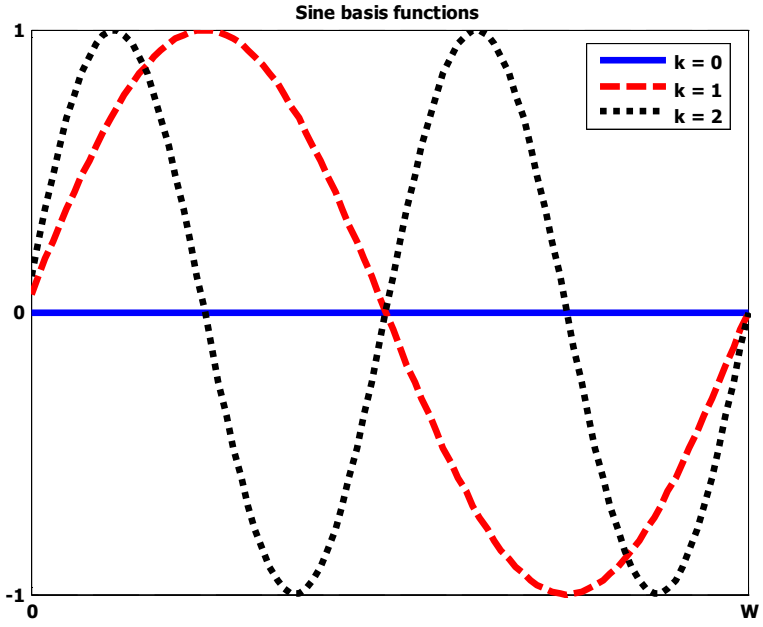

Figure 3-3: Sine basis functions for k=1, 2, 3

If we use a window of length W, we can find W coefficients for this window length. The Fourier coefficients found for this window can be used to construct a feature vector for this time series. Forming an augmented feature vector which consists of Fourier coefficients will result in a vector having the form:

$$v = [a_0\ b_0\ a_1\ b_1 \ldots a_n\ b_n] \tag{3-29}$$

The Fourier coefficients can be visualized as an indicator of the frequency content of the data. The representation containing the first 3 basis functions will be used to construct the feature vectors for the data set. Since b_0 is zero for all time series, our feature vector for a window of length W will be:

$$v = [a_0\ a_1\ b_1\ \ a_2\ b_2] \tag{3-30}$$

A sliding window approach will be used to extract the Fourier coefficients of the data. With each newly coming frame, this window will be shifted to include the new information and exclude the oldest one.

The Fourier coefficients will be calculated for each temporal data listed in Table 2-2. Since we have 4 primary temporal data, namely azimuth, elevation, digital level and area, the feature extraction process will yield a vector of dimension 20. The features for one temporal data are shown in Table 3-3.

Table 3-3: Features extracted from the Fourier transform

Number	Feature	Definition	Equation
1	a_0	The coefficient a_0 (DC component)	3-26
2	a_1	The coefficient a_1	3-25
3	b_1	The coefficient b_1	3-28
4	a_2	The coefficient a_2	3-25
5	b_2	The coefficient b_2	3-28

In order to get a better understanding of the feature extraction methods, it is worth listing the operation steps of the feature extraction algorithm:

1. Wait for the data to accumulate in a *W* frame-length window. In this work *W* is chosen to be 20.

2. Apply Fourier transformation for each temporal data within this window and construct the following vector:

$$v = \left[v^{az}\ v^{el}\ v^{ar}\ v^{dl} \right] \tag{3-31}$$

where

$$v^{az} = \left[a_0^{az}\ a_1^{az}\ b_1^{az}\ a_2^{az}\ b_2^{az} \right] \tag{3-32}$$

$$v^{el} = \left[a_0^{el}\ a_1^{el}\ b_1^{el}\ a_2^{el}\ b_2^{el} \right] \tag{3-33}$$

$$v^{ar} = \left[a_0^{ar}\ a_1^{ar}\ b_1^{ar}\ a_2^{ar}\ b_2^{ar} \right] \tag{3-34}$$

$$v^{dl} = \left[a_0^{dl}\ a_1^{dl}\ b_1^{dl}\ a_2^{dl}\ b_2^{dl} \right] \tag{3-35}$$

3. Shift the sliding window by 1 and repeat the step 2 until the end of the data is reached.

In the figures below (Figure 3-4 and Figure 3-5) the process mentioned above is visualized for a *t-scenario*. The feature extraction method is applied to digital level field of the temporal data. Two consecutive windows are used. First window is between frames 68-88 and the second window is between frames 69-89. The reconstructed waveform by using cosine and sine basis functions with k = 0, 1 and 2 is also shown in the figures.

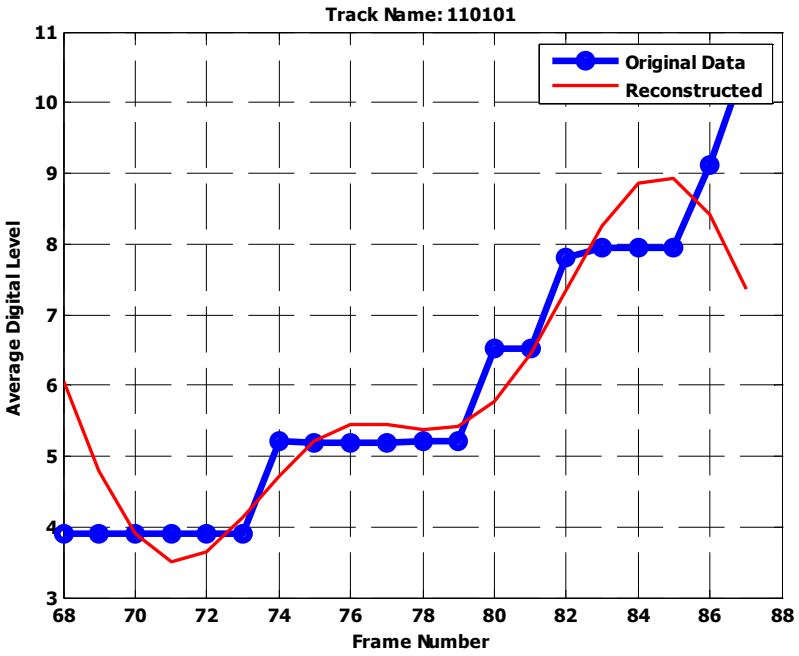

Figure 3-4: Windowing of track 110181 between 68-88 frames

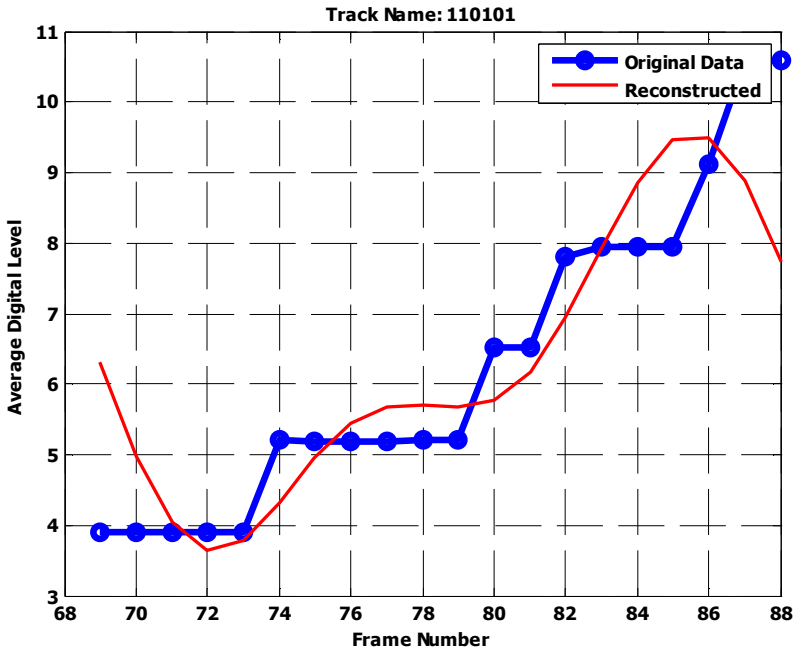

Figure 3-5: Windowing of track 110181 between 69-89 frames

For this track the evolution of the Fourier coefficients is shown in Fig. 2-8 and Fig. 2-9. In the graphs each point on the x-axis represents a sliding window and the lines correspond to Fourier coefficients.

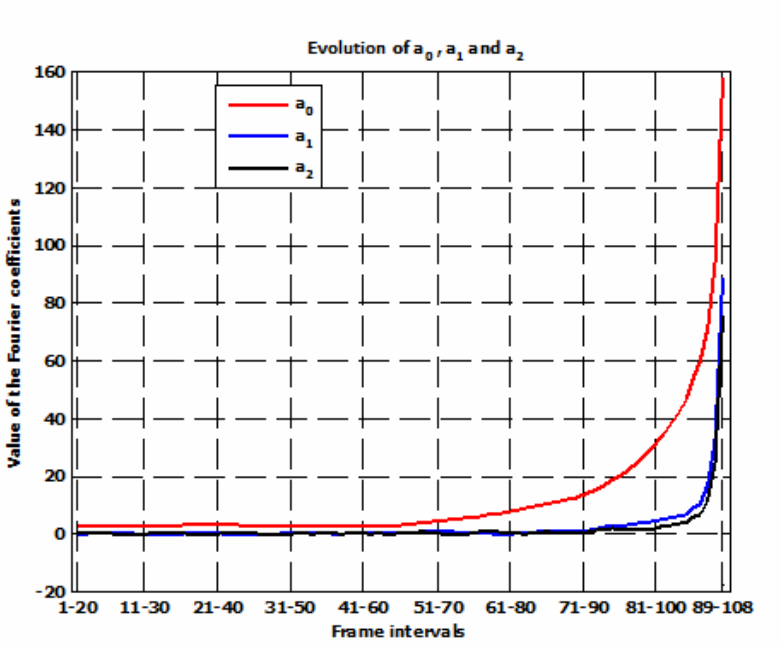

Figure 3-6: Evolution of Fourier coefficients a_0, a_1, a_2

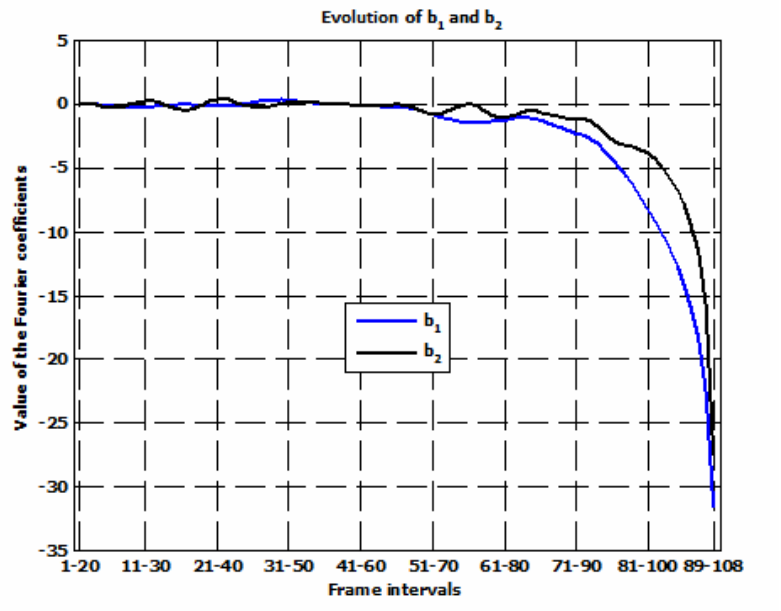

Figure 3-7: Evolution of Fourier coefficients b1, b2

CHAPTER 4

ANALYSIS OF THE FEATURE SETS

4.1 Introduction

In statistical pattern recognition we treat the feature vectors to be classified as random vectors. So it is of great importance to examine and investigate the statistical properties of these random vectors.

We have two different data sets for two different feature extraction methods. The first data set is the result of modeling the time series within a predefined window size with a line and extracting the parameters mentioned in Table 3-2. The second data set is the result of applying the Fourier transformation within a predefined window size and extracting the Fourier coefficients. The set consisting of features extracted by using the line fit parameters will be abbreviated as *LF-Set,* and the set consisting of features extracted by using Fourier transform will be abbreviated as *FT-Set*.

Both data sets share the property that the feature extraction operation is applied to all of the time series mentioned in Table 2-2. In other words, in each data set we have 4 sub-feature sets, extracted from the primary time series azimuth, elevation, area and digital level.

In the following sections; the histograms of the feature sets and basic statistical parameters are examined Then, self organizing maps are used to visualize the data projected to two-dimensions.

4.2 Notation

The feature sets used in this work are based on extracting information from four time series, namely azimuth, elevation, area and digital level. The abbreviations for each feature and their meaning are summarized in the tables below.

Table 4-1: Summary of features extracted by using Fourier Transform

Features Extracted by Using Fourier Transform			
Feature Number	**Abbreviation**	**Source**	**Definition**
1	Az/a0	Azimuth	The coefficient a0 of Fourier Transform
2	Az/a1	Azimuth	The coefficient a1 of Fourier Transform
3	Az/b1	Azimuth	The coefficient b1 of Fourier Transform
4	Az/a2	Azimuth	The coefficient a2 of Fourier Transform
5	Az/b2	Azimuth	The coefficient b2 of Fourier Transform
6	El/a0	Elevation	The coefficient a0 of Fourier Transform
7	El/a1	Elevation	The coefficient a1 of Fourier Transform
8	El/b1	Elevation	The coefficient b1 of Fourier Transform
9	El/a2	Elevation	The coefficient a2 of Fourier Transform
10	El/b2	Elevation	The coefficient b2 of Fourier Transform
11	Ar/a0	Area	The coefficient a0 of Fourier Transform
12	Ar /a1	Area	The coefficient a1 of Fourier Transform
13	Ar/b1	Area	The coefficient b1 of Fourier Transform
14	Ar/a2	Area	The coefficient a2 of Fourier Transform
15	Ar/b2	Area	The coefficient b2 of Fourier Transform
16	Dl/a0	Digital Level	The coefficient a0 of Fourier Transform
17	Dl /a1	Digital Level	The coefficient a1 of Fourier Transform
18	Dl/b1	Digital Level	The coefficient b1 of Fourier Transform
19	Dl/a2	Digital Level	The coefficient a2 of Fourier Transform
20	Dl/b2	Digital Level	The coefficient b2 of Fourier Transform

Table 4-2: Summary of features extracted by using Linear Fitting

Features Extracted by Using Fourier Transform			
Feature Number	**Abbreviation**	**Source**	**Definition**
1	Az/a	Azimuth	1st coefficient of the line
2	Az/b	Azimuth	2nd coefficient of the line
3	Az/se(a)	Azimuth	Standard error for "a"
4	Az/se(b)	Azimuth	Standard error for "b"
5	Az/f(t)	Azimuth	Fit value of the line at time t
6	Az/rmse	Azimuth	Root mean square error of the fit
7	Az/r^2	Azimuth	Square of multiple correlation coefficient
8	El/a	Elevation	1st coefficient of the line
9	El/b	Elevation	2nd coefficient of the line
10	El/se(a)	Elevation	Standard error for "a"
11	El/se(b)	Elevation	Standard error for "b"
12	El/f(t)	Elevation	Fit value of the line at time t
13	El/rmse	Elevation	Root mean square error of the fit
14	El/r^2	Elevation	Square of multiple correlation coefficient
15	Ar/a	Area	1st coefficient of the line
16	Ar/b	Area	2nd coefficient of the line
17	Ar/se(a)	Area	Standard error for "a"
18	Ar/se(b)	Area	Standard error for "b"
19	Ar/f(t)	Area	Fit value of the line at time t
20	Ar/rmse	Area	Root mean square error of the fit
21	Ar/r^2	Area	Square of multiple correlation coefficient
22	Dl/a	Digital Level	1st coefficient of the line
23	Dl/b	Digital Level	2nd coefficient of the line
24	Dl/se(a)	Digital Level	Standard error for "a"
25	Dl/se(b)	Digital Level	Standard error for "b"
26	Dl/f(t)	Digital Level	Fit value of the line at time t
27	Dl/rmse	Digital Level	Root mean square error of the fit
28	Dl/r^2	Digital Level	Square of multiple correlation coefficient

4.3 Basic Statistical Parameters

The mean and standard deviation can be considered as the most basic statistical parameters for a data set. It is essential to examine the behavior of these parameters since they will be used as normalization parameters for data set visualization and classification. Furthermore, these parameters may give information about the class separability. Although in higher dimensions joint-statistics of all features should be considered to calculate the class separability, it is worth to investigate the individual features. The mean values can be thought as a measure indicating how far away the class clusters are from each other and the standard deviations indicate the overlap between the class boundaries.

In Appendix - B, the mean and standard deviation values for t and f-scenarios for each feature set are presented. In order to get a visual feeling about the mean and standard deviation values, error bar charts are shown in Figure 4-1 and Figure 4-2. In Figure 4-3 and Figure 4-4 normalized values for t and f-scenarios are shown. Normalization is based on the mean and standard deviation of the t- scenarios. Assume that the mean and the standard deviation of the t-scenarios of the first data set which consists of the features extracted by using linear fitting are abbreviated as $\mu_{t,lf}$ and $\sigma_{t,lf}$, respectively. And let the mean and the standard deviation of the t-scenarios of the second data set which consists of the features extracted by using Fourier transform are abbreviated as $\mu_{t,ft}$ and $\sigma_{t,ft}$, respectively. Then for a vector x the normalization is done in the following way:

$$\tilde{x}_{ft} = \frac{x_{ft} - \mu_{t,ft}}{\sigma_{t,ft}} \tag{4-1}$$

$$\tilde{x}_{lf} = \frac{x_{t,lf} - \mu_{t,lf}}{\sigma_{t,lf}} \tag{4-2}$$

where $\tilde{x}$ is the normalized version of the vector x.

Inspecting Figure 4-3 and Figure 4-4 reveals that for some features the mean and the standard deviation values of t and f-scenario classes are not very different from

each other. This may stem from the fact that these features are unable to discriminate the classes as compared to the others. However before reaching such a decision, some feature selection algorithms must be applied and the selected features should be tested with appropriate classifiers. The mean and standard deviation values can only give an insight about the feature selection process.

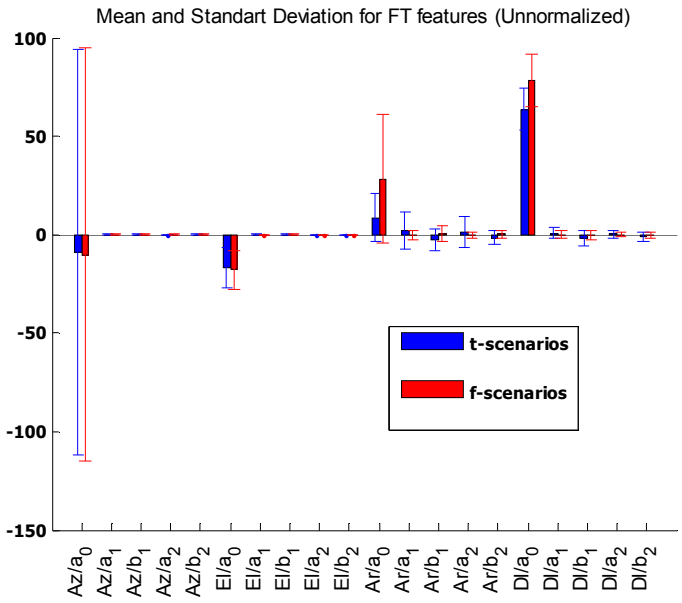

Figure 4-1: Error bar chart for FT features

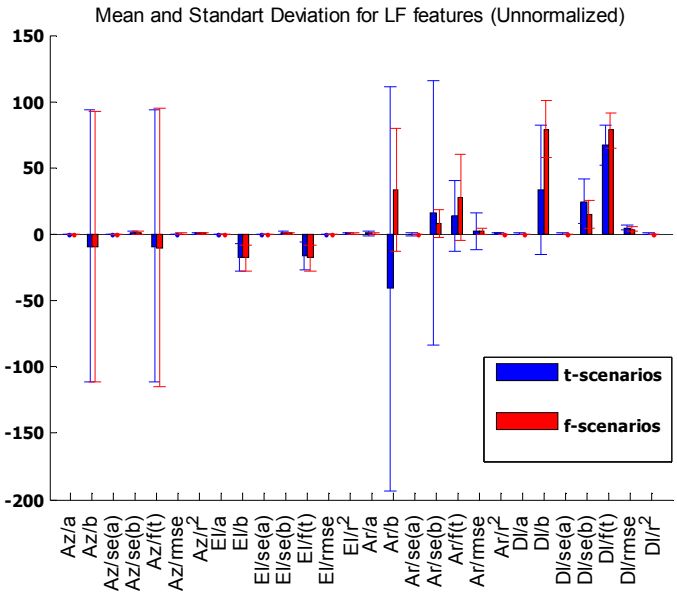

Figure 4-2: Error bar chart for LF features

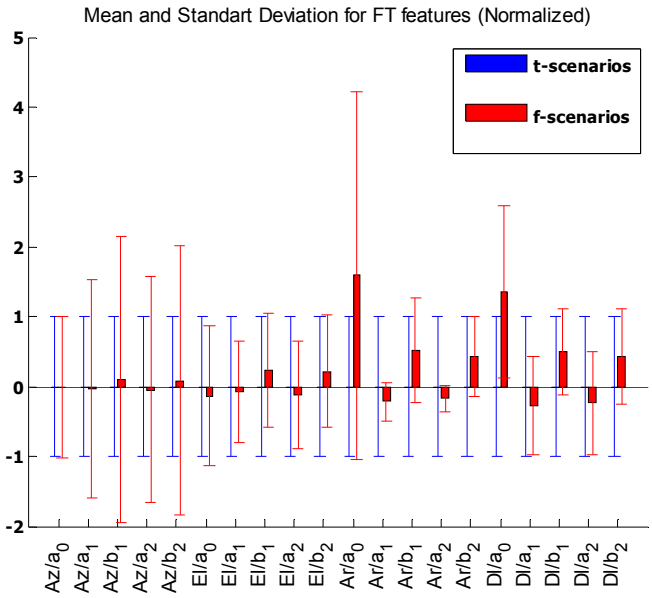

Figure 4-3: Error bar chart for FT features (Normalized)

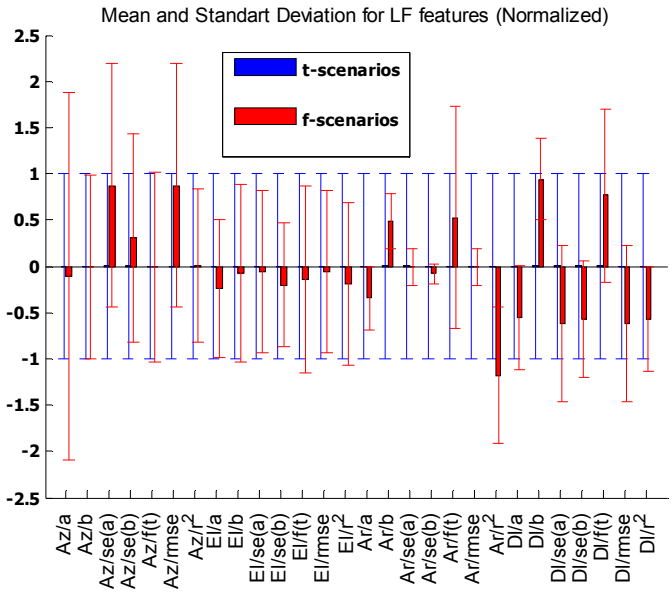

Figure 4-4: Error bar chart for LF features (Normalized)

4.4 Histograms

Histograms can be used to approximate the distribution of a data set. In this part the histograms of the individual features are examined. The features are normalized as explained in Section 4.3. In each histogram plot the distribution of both t and f-scenarios are shown. The histogram plots are given in Appendix C.

4.5 Correlation Analysis

In this part the correlation between the features of the data set is examined. The correlation matrix is used to visualize the pair wise correlations among the features. The correlation matrices are calculated for t and f-scenarios of each data set. The correlation matrices are given in Appendix D.

4.6 Self Organizing Maps

4.6.1 Brief Information about Self Organizing Maps

Self Organizing Maps can be used to project higher dimensional data into two dimensions. A self organizing map consists of neurons on a two dimensional grid. Each neuron on the grid is associated with a *k-dimensional* prototype. Here k represents the number of features for a data set and each entry of the prototype vector is named as a *weight*. The neurons are connected to each other by a neighborhood relationship. Training phase of a self organizing map can be summarized as follows [21]:

1. The prototype vectors associated with each neuron are randomly initialized.

2. An input pattern from the data set is fed to the network. Let's assume that the input pattern is x.

$$x = [x_1, x_2, ..., x_k] \tag{4-3}$$

3. The nearest prototype vector (in terms of Euclidian distance) is found:

$$\|x - w_c\| = \min_i \{\|x - w_i\|\} \tag{4-4}$$

where w_c indicates the nearest prototype vector in the map.

4. The weights of the nearest prototype and its neighbors are increased. A proper neighborhood function can be defined as:

$$h_{ci}(t) = h_0 \exp(-\|r_i - r_c\|^2 / \sigma(t)^2) \tag{4-5}$$

where r denotes the coordinate of the prototype vectors on the two dimensional grid, h_0 is a constant and σ(t) is the variance. σ(t) should be chosen as a

decreasing function of time so that at each iteration of the algorithm the neighbourhood relations are confined to a smaller area.

5. The update equation for the weights is defined as:

$$w_i(t+1) = w_i(t) + \alpha(t)h_{ci}(t)\left[x - w_i(t)\right] \quad (4\text{-}6)$$

where $\alpha(t)$ is a decreasing function of time and represents the learning rate.

After the training process the neurons that have similar weights become clustered in the two dimensional grid. This property enables one to visualize higher dimensional data in two dimensions.

4.6.2 Analysis

The analyses are carried out by using self organizing map toolbox [22]. For each feature set, self organizing maps of 30x15 neurons are trained. Before the training, no feature selection algorithm is applied and all of the features from each data set are used.

In the figures below, distance matrices which can be used to visualize the distance between neighboring neurons are shown. Each value on the map is the median distance of the neuron to each of its neighboring neurons. High values in Figure 4-5 indicate the possible cluster borders and uniform low areas can be thought as clusters. We can see more than two clusters in each d-matrix in Figure 4-5 which are mainly due to three types of false alarms and three types of threats used in the scenario generation process. However in the scope of this thesis the main aim is to discriminate between two classes; namely, false alarms and threats and the work is concentrated on analyzing these two classes.

Another way to visualize the data set on the self organizing map is achieved by using a simple voting procedure. First best matching units for each neuron in the map are calculated by using the complete data set, and then the label of the class which has more hits on a specific neuron is assigned. The best matching unit stands

for the nearest prototype in the self organizing map to a vector in a data set in terms of Euclidian distance. The resulting distribution is depicted in Figure 4-7 where T stands for threats and F stands for false alarm label. Here it can be seen that although the boundaries between classes are not smooth, t and f-scenarios are clustered and can be visually discriminated from each other.

Hit histograms are another way to visualize the data set on the self organizing maps. They indicate how the best matching units of a data set are distributed over the self organizing map. The size of the shapes is directly proportional to the number of data hit on a neuron. The hit histograms are shown in

Figure 4-8 and Figure 4-9. Examination of the hit histograms reveals that although the patterns from t and f-scenarios seem to be grouped together, there exist some ambiguous regions for each class. The patterns in these ambiguous regions are generally hard to classify and cannot be easily distinguished from each other.

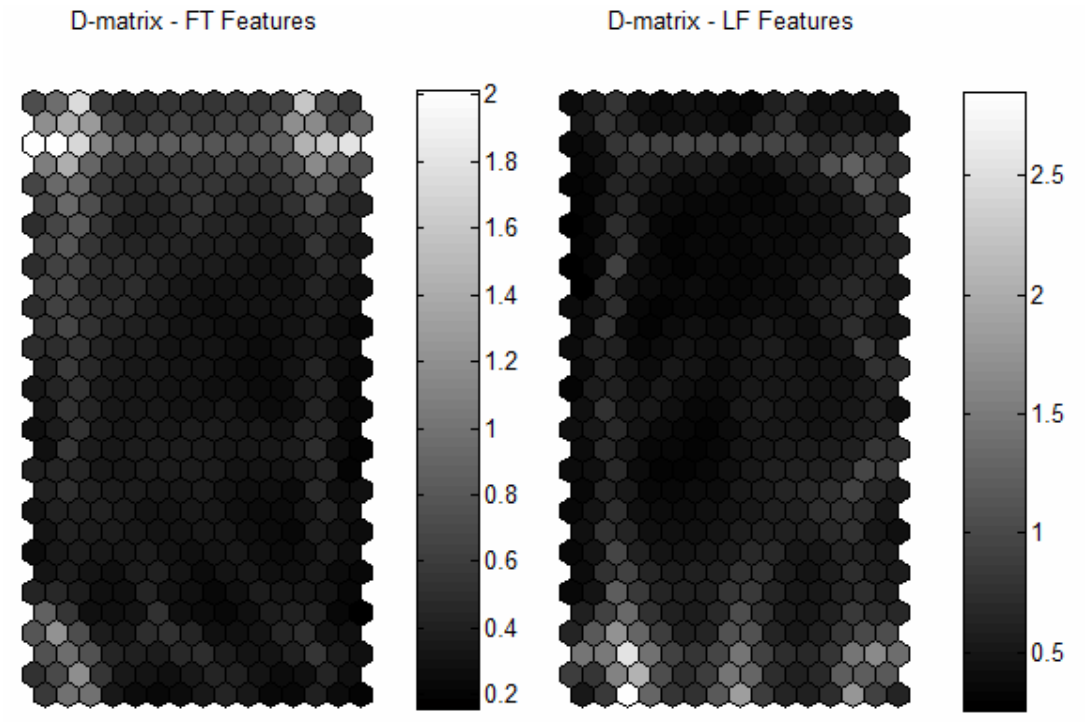

Figure 4-5: Distance matrices for both feature sets

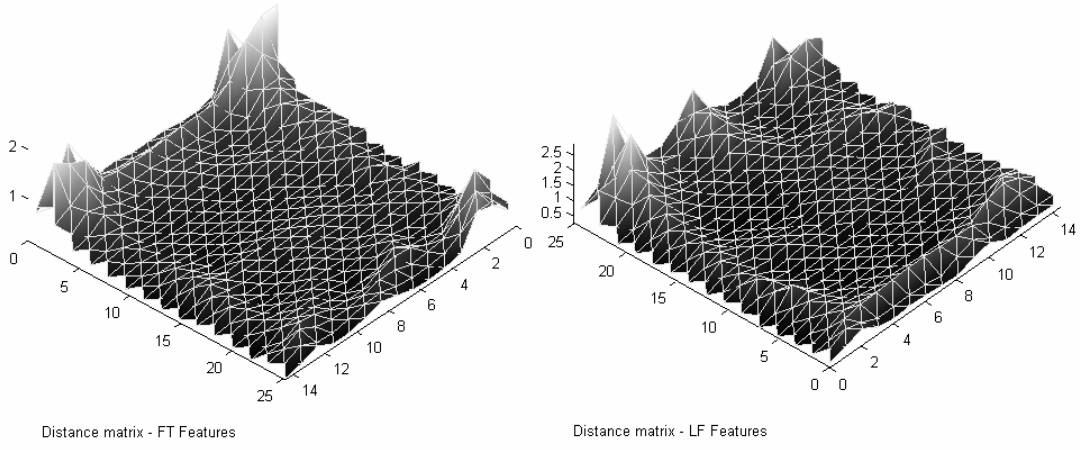

Figure 4-6: Distance matrices for both feature sets (3-D View)

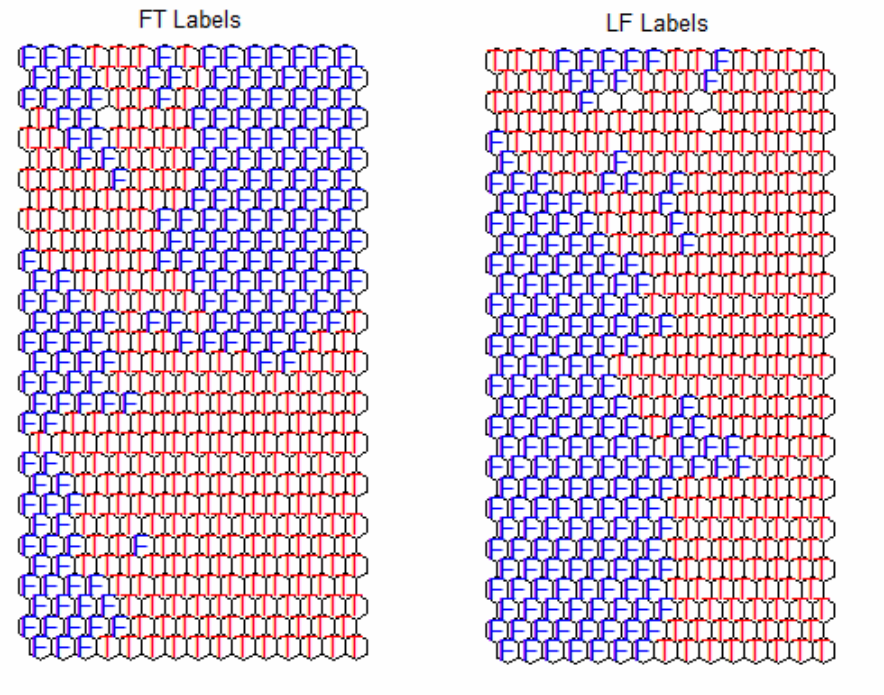

Figure 4-7: Data on self organizing map – Voting Procedure

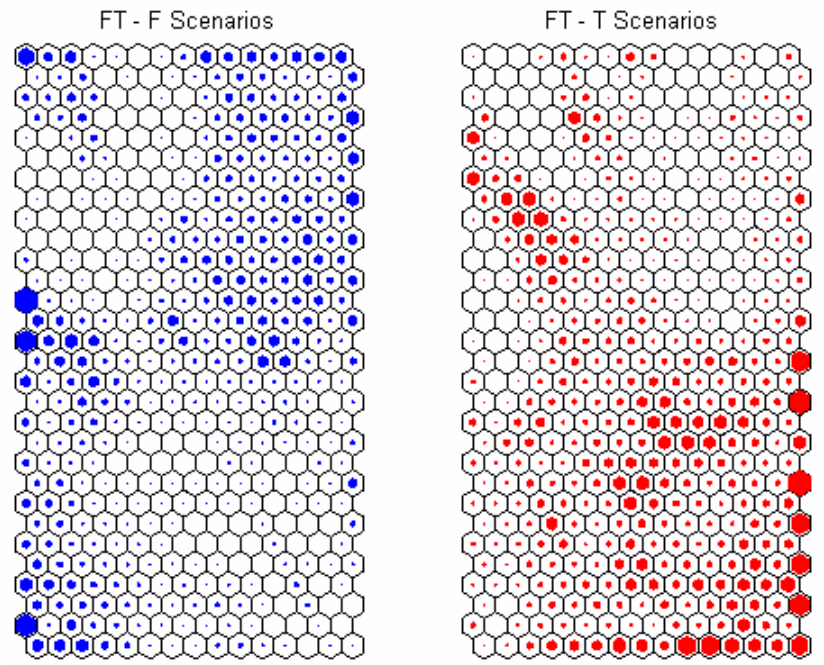

Figure 4-8: Hit Histograms for features extracted by using Fourier transform

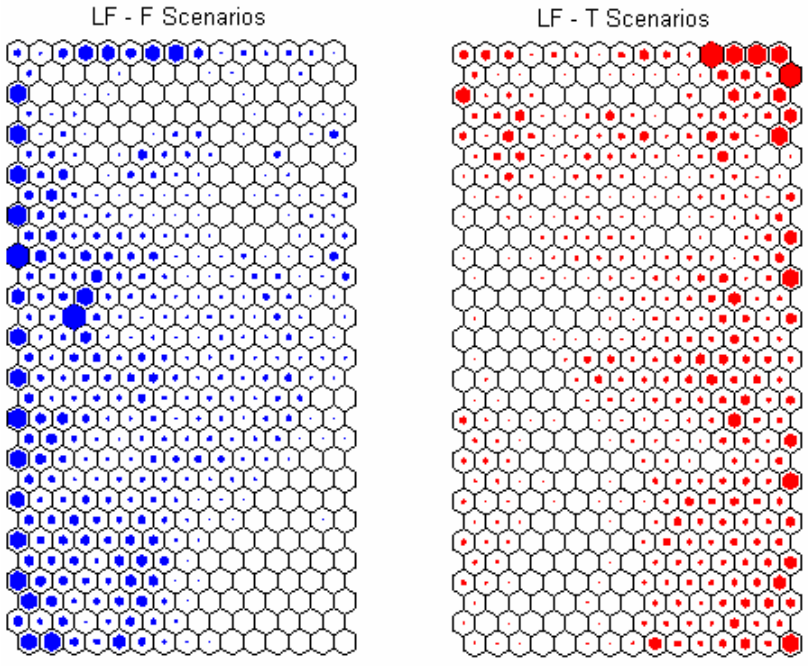

Figure 4-9: Hit Histograms for features extracted by using linear fitting

CHAPTER 5

CLASSIFICATION

5.1 Introduction

If we look at the problem of developing a GMW algorithm from the pattern recognition viewpoint, building a successful classifier for the decision making algorithm is very important to increase the probability of declaration, to achieve faster reaction times and to obtain lower false alarm rates. An approach developed in the recent years in the pattern recognition domain is to combine the decisions of multiple classifiers instead of using only one classifier. By means of such an approach, better and more reliable classifiers can be built.

In this section classification methods that use multiple classifiers will be examined and the design criteria for multiple classifier systems are explained.

5.2 Multiple Classifier Systems

The main motivation behind using multiple classifiers instead of a single one is the intuitive idea that the opinions of multiple decision makers will yield a more reliable conclusion than the opinion of a single decision maker.

Traditional pattern recognition approaches aim to select the best classifier for a problem by comparing their error rates via validation data. However, in real world validation data may be collected in limited amounts preventing it to reflect the true characteristics of the data. In the worst case, this may result in constructing an

unsuccessful classifier for a problem. Using multiple classifiers avoids this risk by combining different classifiers' decisions.

MCS (Multiple Classifier Systems) are reported to be more successful in terms of classification performance than the individual members of the set [7]. The necessary conditions for the MCS to become more efficient than the individual members of the ensemble can be stated as follows: 1- The classifiers should achieve *a classification rate better than the random guess*, in other words they must have a probability of error less than 1/2. 2 – The classifiers should be *diverse.*

Diversity implies that the individual classifiers should perform differently on the same data set. The increased diversity among the members of the classifier ensemble leads to a reduction of the final classification error of the ensemble [6].

Hansen & Salamon (1990) [8] proved that if the classifiers in an ensemble are independent and their error rates are equal, then the error rate of the ensemble decreases with the increasing number of classifiers. Assume that an ensemble is composed of classifiers, each of which can correctly classify a pattern with a probability of *1-p* and the errors they make are independent. Then the probability for a pattern to be misclassified *k* times by *N* classifiers is:

$$P_1 = \binom{N}{k} p^k (1-p)^{N-k} \tag{5-1}$$

Now, assume that we want to combine the decision of the individual classifiers by using the majority voting rule. (The details of the majority voting rule are explained in 5.3.2.1.) Than the probability that this pattern is misclassified becomes:

$$P_2 = \sum_{k>N/2}^{N} \binom{N}{k} p^k (1-p)^{N-k} \tag{5-2}$$

It can be shown that the above equation is a monotonically decreasing function of N, provided that p is smaller than 0.5 and N is odd. Hansen & Salamon also propose a model for random error rates; however they report that this model predicts a far

superior performance to what they experimentally observe, owing to the fact that classifiers are not independent.

Another important aspect in creating a MCS is the combination function of the classifier decisions. Although in the literature it is claimed that the role of the combination function is not as important as the diversity itself, Kuncheva [9] has shown its importance in the performance of MCSs.

5.3 Design criteria of MCS Systems

The discussion in the previous section points out the main factors that affect the design of a MCS. These factors determine the general structure of the MCS and its decision fusion mechanism.

To be more precise, the issues that are needed to be considered when constructing the architecture of a MCS, can be listed as follows:

1. ***Diversity:*** The classifiers should be as diverse as possible. The training phases, features used, the parameters of the classifiers or the classifiers themselves should be adopted to achieve the desired diversity among the ensemble. The individual classifiers must have an error rate better than the random guess.

2. ***Combination Rule:*** The decision of the individual classifiers should be fused in a suitable way to minimize the probability of error.

3. ***Topology:*** Determines how the operations and the interactions of the classifiers occur within the ensemble. Topologies that are generally used are serial, parallel and hybrid topologies.

5.3.1 Diversity generation

Diversity among the members of the ensemble can be achieved by using various methods. Most famous methods to achieve diversity are bagging, boosting and using different feature subsets.

5.3.1.1 Bagging

Bagging is the acronym for "bootstrap aggregation". Bootstrap sampling is a method for approximating the distribution underlying a data set [11]. In the method of bagging, diversity among the members of the ensemble is achieved by a certain manipulation of the training samples.

The main idea behind bagging is to generate a subset of a training set by bootstrap sampling. The generated subset has the same size as the original set, however it contains replicated patterns in it. Breiman [10] proved that if the sample size for a training set is sufficiently large (i.e. as the training set size approaches infinity) probability of an instance to be replicated is about 0.632. The proof can be sketched as follows:

Assume that we have a set of size N, then the probability for a pattern to be replicated as the sample size goes to infinity becomes:

$$P = \lim_{N\to\infty}\left[1-\left(1-\frac{1}{N}\right)^{N}\right] = 1-\frac{1}{e} \cong 0.632 \qquad (5\text{-}3)$$

This suggests that about 63.2 percent of the training set is contained in the re-sampled set created by using bagging.

Subsequently these subsets are used to train individual classifiers and the decisions of the individual classifiers are combined by using a simple scheme such as majority voting or simple averaging. In order to get a better understanding of the operation, bagging method is illustrated in Figure 5-1.

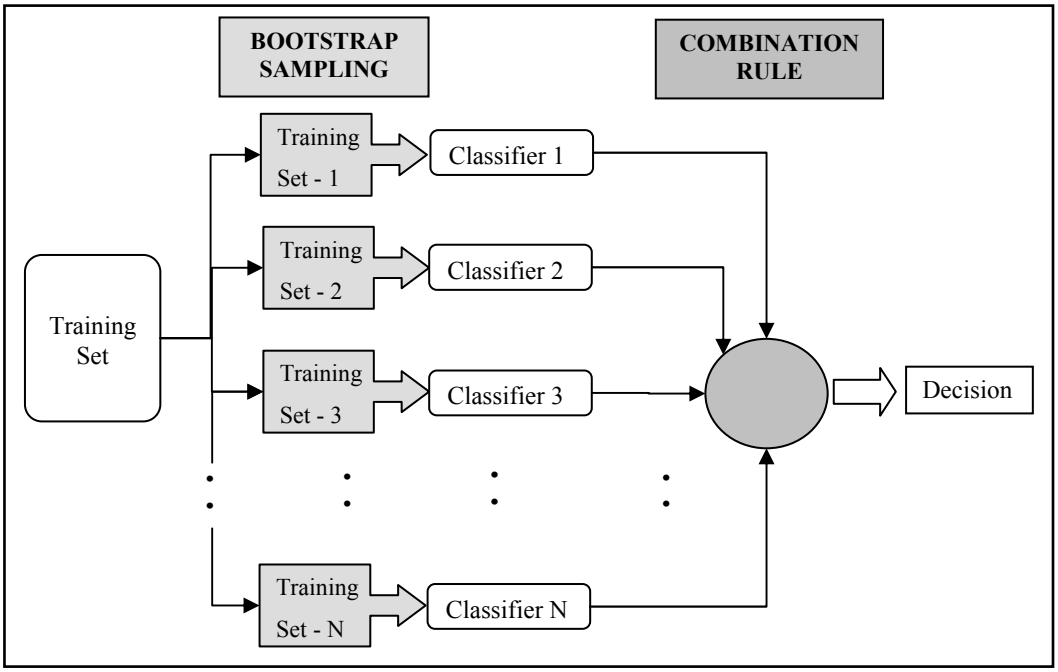

Figure 5-1: Bagging Method

5.3.1.2 Boosting

This algorithm achieves the diversity among the members of the ensemble by changing the sampling distribution adaptively and it also proposes a combination method.

The concept of boosting was proposed by Schapire in 1990 [12]. In this work, it was proven that any classifier with an error rate better than random guess can be used to build a better classifier. In other words, any weak leaner can be boosted into a strong learner.

Later in 1997 Schapire and Freund [13] introduced the AdaBoost algorithm. This algorithm is similar to the bagging in approach, because successive classifiers are trained by sampling the training data. However, the major superiority of the algorithm is that the distribution from which the samples are drawn is iteratively updated to give more importance to the misclassified samples. In this way, successive classifiers are trained to compensate the errors of the previous ones. At the end of the training phase, weighted majority voting scheme is applied as a

combination rule for the classifiers. The weight for each classifier is determined by the error observed on the training sets.

The boosting algorithm proposed by Schapire and Freund is summarized in Table 5-1:

Table 5-1: AdaBoost Algorithm

Definitions:

1. Consider a training set, *E*. Assume that there are *n* training instances denoted by *x* with their class labels denoted by *y*.

   $E = \{x_i, y_i\}$, $i = 1,...,n$

2. Assume that the number of the classifiers in the ensemble is k_{max} and each classifier is denoted by S_k

3. Let the sampling distribution for the k^{th} classifier be W_k

   $W_k = \{W_k(1),....,W_k(n)\}$, $k = 1,..., k_{max}$

4. The error for the k^{th} classifier is H_k.

Algorithm:

$W_1 = 1/n$ (Initialize to uniform distribution)

FOR (k = 1:k_{max})

1. *Sample E by using W_k and create E_k*
2. *Train S_k by using E_k and calculate H_k.*
3. *Calculate*

$$\alpha(k) = 1/2 \ln[(1 - H_k)/H_k] \quad (5\text{-}4)$$

4. *Update* the distribution:

$$W_{k+1}(i) = \frac{W_k(i)}{\sum_{i=1}^{n} W_k(i)} \times \begin{cases} e^{-\alpha(k)} & \text{if } x_i \text{ is correctly classified} \\ e^{\alpha(k)} & \text{else} \end{cases} \quad (5\text{-}5)$$

END

5.3.1.3 Using Different Feature Subsets

Choosing different feature subsets may help the classifiers among the ensemble to achieve diversity. The subsets may be generated either by randomly picking some combination of the features from a larger feature set or by using different feature extraction techniques. The classifiers may be trained on these different subsets so that the ensemble may achieve a good performance.

Random subspace method [14] is one of the most popular ensemble generations methods that aim to achieve classifier diversity by training classifiers on randomly selected feature subspaces.

In [15] Cherkauer proposes a method in this direction in the context of image processing. For this purpose he introduces an ensemble of neural networks trained on different feature subsets. The input feature subsets are not selected randomly; however the domain knowledge of the designer is applied in creating subsets. By using different image processing techniques, different feature subsets are formed and the neural networks are trained by using these data. The performance of the method is claimed to be as high as the performance of a human expert.

5.3.2 Combination rules

In the previous section, methods for generating diverse classifiers are discussed. Although some methods, like AdaBoost, suggest a systematic algorithm to generate both diverse classifiers and a combination rule for their decisions, the ensemble decision mechanism must be handled properly for the successful performance of the multiple classifier systems.

Xu et al, [16] provides a general framework to group similar output combination methods based on the information supplied by the classifiers. According to their taxonomy, there exist 3 kinds of output information:

1. Abstract Level: A classifier only outputs a unique label

2. Rank Level: A classifier ranks the label of the candidate classes

3. Output Level: A classifier outputs confidence values for classes.

In addition to the output information, combination rules can also be categorized as trainable or non-trainable rules. Trainable rules require the parameters of the combination method to be estimated by using training data. For example, these parameters can be the weights of the classifiers for a weighted voting scheme. Furthermore, the output of the classifiers can also be treated as random vectors and provide input for a meta-classifier whose job is to output the decision of the ensemble. Non-trainable rules do not require training data, their operation is static rather than dynamic, and these rules do not have parameters that are needed to be adjusted. Majority voting is the most common example of non-trainable rules.

The output information, structure of the voting rule (trainable vs. non-trainable) and diversity generation method affect the design of combination rules. In the following sections, the combination methods that are mainly used are examined.

5.3.2.1 Majority Voting

This method assumes that we have an ensemble of classifiers each of which outputs the information in abstract level. Final decision is made by counting the votes of each classifier. The class that achieves the highest vote among all is considered as the winner. The idea can be formulated mathematically as below:

Assume that we have N classifiers and M classes, denoted by C and J respectively and the pattern to be classified is denoted by x. The label of the pattern x is assigned by the operator L as $L(x)$.

$C = \{C_1, \ldots, C_N\}$

$J = \{ J_1, \ldots, J_M \}$

Let $O_{n,m}$ denote the decision of the n^{th} classifier for the m^{th} class:

$$O_{n,m} = \begin{cases} 1 \text{ if } C_n \text{ decides in favor of the m}^{\text{th}} \text{ class} \\ 0 \text{ otherwise} \end{cases} \quad (5\text{-}6)$$

Then, the class label is determined according to the formula below:

$$L(x) = \arg\max_{1 \le m \le M} \sum_{n=1}^{N} O_{n,m} \quad (5\text{-}7)$$

5.3.2.2 Weighted Majority Voting

Weighted Majority voting is an extension of the procedure mentioned above. In this case the output of each classifier is multiplied by a factor and the resulting decision values are summed. By using similar definitions mentioned in Section 5.3.2.1 and assuming that the weight of each classifier is W_n, then the weighted majority voting rule can be formulated as:

$$L(x) = \arg\max_{1 \le m \le M} \sum_{n=1}^{N} W_n O_{n,m} \quad (5\text{-}8)$$

Weighted majority voting is used in the AdaBoost algorithm. The weights of the classifiers in AdaBoost are calculated in the training phase of the algorithm being inversely proportional to the error they make on the training set. In fact, this can be considered as the most intuitive way of assigning weights to an ensemble of classifiers and can be applied to any other ensemble generation algorithm.

5.3.2.3 Borda Count Method

Borda Count method is an example of rank-level based combination methods. This method can be used to order the classes with ranks indicating the strength of the decision of the classifier.

Assume that we have N classifiers and M classes. The rank assigned by the classifier n to class m is indicated by $R_{n,m}$. Then the Borda Count label assignment procedure can be formulated as:

$$L(x) = \underset{1 \le m \le M}{\arg\max} \sum_{n=1}^{N} R_{n,m} \tag{5-9}$$

5.3.2.4 Other Algebraic Combination Rules

There are other algebraic combination rules that can be used for decision combination purposes. They are listed below:

1. Maximum rule: Assuming that the classifiers provide continuous output levels, the decision of the classifier which yields the highest output score is considered as the final decision. The idea behind this rule is to trust the most confident classifier. However it is sensitive to badly trained classifiers.

   Assume that we have N classifiers, then label of the input pattern x becomes:

$$L(x) = \underset{1 \le n \le N}{\arg\max} C_n(x) \tag{5-10}$$

2. Minimum rule: In this case, depending on how $C_n(x)$ is defined, the decision of the classifier which outputs the minimum score is chosen.

$$L(x) = \underset{1 \le n \le N}{\arg\min} C_n(x) \tag{5-11}$$

3. Median rule: The median value of the classifiers' output score is found and the corresponding label is assigned to the input pattern.

$$L(x) = \underset{1 \le n \le N}{median} C_n(x) \tag{5-12}$$

4. Product Rule: The output score values of the classifiers are multiplied. The result is used to classify the input pattern.

$$L(x) = \arg\max_{1 \le m \le M} \prod_{n=1}^{N} C_n(x) \tag{5-13}$$

The final decision obtained by the product rule is highly affected by the least confident classifiers. In the worst case, this may lead to false decisions if a badly trained classifier exists in the ensemble. [17]

5. Sum rule: This rule is similar to the majority voting rule. The main difference is that the output levels of the ensemble members are considered. The sum of the continuous outputs is used to reach the final decision.

$$L(x) = \arg\max_{1 \le m \le M} \sum_{n=1}^{N} R_{n,m} \tag{5-14}$$

Kittler et al. [17] have reported that the sum rule outperforms the rules such as maximum, minimum, median, majority voting and product rule under most restrictive assumptions. This has also been found to be most robust method among all.

5.3.2.5 Behavior Knowledge Space

The approach of Behavior Knowledge Space for combining multiple classifiers is proposed by Huang and Suen [18]. In this method there exists two steps: 1- Finding all possible combinations of the classifiers' decisions. 2- Taking into account the real labels of the classes in the training data, assign a decision for each combination of the classifiers' decision. In order to exemplify the idea, assume that we have 3 classifiers $C = \{C_1, C_2, C_3\}$ and 2 classes $J = \{J_1, J_2\}$. In Table 5-2 below all possible combinations of the classifiers are listed and next to them the number of classes that belong to these combinations are listed.

Table 5-2: BKS Example

Assignment of Classifiers			True Class	
C_1	C_2	C_3	J_1	J_2
J_1	J_1	J_1	23	84
J_1	J_1	J_2	71	37
J_1	J_2	J_1	22	49
J_1	J_2	J_2	45	69
J_2	J_1	J_1	13	26
J_2	J_1	J_2	48	10
J_2	J_2	J_1	73	21
J_2	J_2	J_2	67	45

For example we have an input pattern x that is classified by the ensemble as $\{J_1, J_2, J_1\}$. If we look at the Table 5-2 it is seen that in the training set this combination belongs to class J_1 for 22 times and belongs to class J_2 for 49 times. So the final decision of the ensemble for this pattern becomes J_2.

It should be noted BKS method tries to estimate the posterior class probabilities for each combination. If we have M classifiers and N classes we need to estimate N^M posterior probabilities which may be formidably large. This may ultimately lead to serious problems if the number of training samples is low compared to the number of posterior probabilities to be estimated [19].

5.3.2.6 Stacked Generalization

Stacked generalization method aims to combine the decisions of individual classifiers by the help of a high-level classifier. The high-level classifier uses the outputs of the ensemble as its feature space and predicts the label of the input pattern [20].

The training phase of the Stacked Generalization method for a 2-level architecture can be explained as follows:

Assume that we have 3 level-0 classifiers $C^0 = \{C_1^0, C_2^0, C_3^0\}$ and 1 level-1 classifier C^1. Our training set is Z with 4 disjoint subsets Z_1, Z_2, Z_3, and Z_4.

Train each classifier using one of the disjoint sets. An example training set combination is shown in the following table:

Table 5-3: Example training set setup

Training Set for C_1^0	*Training Set for C_2^0*	*Training Set for C_3^0*
Z_1	Z_2	Z_3
Z_2	Z_3	Z_4
Z_3	Z_4	Z_1

Level-1 classifier is trained on the data set obtained by using the vectors formed from the outputs of the level-0 classifiers and the correct class labels. The input patterns to the level-0 classifiers are chosen from the unused disjoint set during the training.

The major drawback of this approach is the fact that it requires a large amount of training data. Furthermore the output space of the low level classifiers may not be appropriate to be used as an input feature space.

5.3.3 Topology

Topology determines the way how the classifiers are trained and how the data is processed within the ensemble. Commonly used topologies are [23]:

1. **Serial:** In this topology the classifiers are applied one after another producing a reduced set of possible classes for each pattern. By this way a difficult classification problem is gradually reduced to a simpler one.

2. **Parallel:** Each classifier in the ensemble operates in parallel. The input pattern to the ensemble is evaluated by each classifier and their decisions are combined to reach a final decision.

3. **Hybrid:** This structure tries to exploit the fact that some classifiers perform better on particular patterns. A classifier selection rule is applied to input patterns and best classifier for the input pattern is used.

4. **Conditional:** If the input pattern fed to the ensemble cannot be easily classified or appears to be ambiguous for the first classifier, then a secondary classifier is applied to resolve the conflict. By this way, a more complex classifier is used for difficult patterns and computational efficiency can be achieved. This structure becomes complicated if the number of classifiers is more than two.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Methodology

In this chapter, experimental results for selected multiple classifier systems are presented. The performance of the classifier ensembles are evaluated for both of the feature sets presented in Section 3.3. The set consisting of features extracted by using the line fit parameters will be abbreviated as *LF-Set,* and the set consisting of features extracted by using Fourier transform will be abbreviated as *FT-Set*.

In order to evaluate the classification performance the approaches given below are followed.

6.1.1 Feature-based evaluation

In this evaluation method, all features extracted from the scenarios are pooled into a single data set. The features are labeled by the type of the scenario class that they are coming from, namely t-scenarios or f-scenarios.

The performances of the classifiers are evaluated on this data set by using the *K-Fold Cross Validation* procedure. In this method the data set is divided into *K* subsets of size *N/K*, where *N* is the size of the data set. Then one of the subsets is used as a test data set and the remaining *K-1* subsets are merged to form the training set. This process is repeated for each of the *K* subsets and the classification accuracies are averaged.

6.1.2 Scenario-based evaluation

As mentioned in Section 3.3.1 and 3.3.2, the features are extracted from the scenarios by using a sliding window approach. So each scenario can be visualized as a series of feature vectors.

In order to get a better understanding about this method, the training and testing steps should be examined separately.

Training: Firstly, scenarios that will be used in the training phase are selected. Then the features extracted from these scenarios are used as the training set for the classifier. The rest of the scenarios are used as test cases.

Testing: For each scenario in the testing set, the feature vectors are evaluated by the classifier. By this way the series of feature vectors of a scenario is converted to a series of score values given by the classifier. The score value indicates the confidence level of the classifier for an input pattern.

At this point, the score values are compared with a threshold and a decision about each feature vector of a scenario is made. To be more precise, the feature vectors of a scenario are replaced with score values first, and then, with labels of t-scenario and f-scenario.

The main aim of the scenario-based evaluation method is to label the whole scenario, not only the features of the scenario, as threat or false alarm by using a simple decision rule. According to this decision rule, a scenario is considered as **threat** if ***P*** consecutive feature vectors for a scenario are labeled as t-class, otherwise the scenario is labeled as false alarm.

After a scenario is labeled by the classifier, the following measures are extracted:

- If the scenario is a t-scenario and the classifier has labeled it correctly **the reaction time** for this scenario is calculated as the number of frames from the beginning of the scenario to the frame before which P consecutive frames are labeled as threats.

- If the scenario is a t-scenario and the classifier has labeled the scenario as an f-scenario, then the scenario is considered as a **non-declared scenario**.

- If the scenario is an f-scenario and the classifier has labeled the scenario as a t-scenario, then the scenario is considered as **a false alarm**.

When the measures mentioned above are found for all the scenarios in the testing set, the following measures are calculated:

Mean Probability of Declaration (MPOD):

MPOD = (Total Number of Scenarios – Non-Declared Scenarios) / Total Number of Scenarios

Mean Reaction Time (MRT):

MRT = Mean Value for Reaction Time[4]

Total Number of False Alarms (TNFA):

TNFA = Total Number of False Alarm Scenarios

The scenario based evaluation is depicted in Figure 6-1. The y-axis represents the output of the classifier and x-axis represents the number of the feature vector in the scenario. For each feature vector, the confidence output of the classifier is calculated and plotted. The vertical red line is the score threshold which is -4.5 in this case. If the score values are below this threshold for three consecutive frames, alarm is declared. The vertical red line indicates the number of the feature vector where the alarm is declared.

[4] Only declared scenarios are used

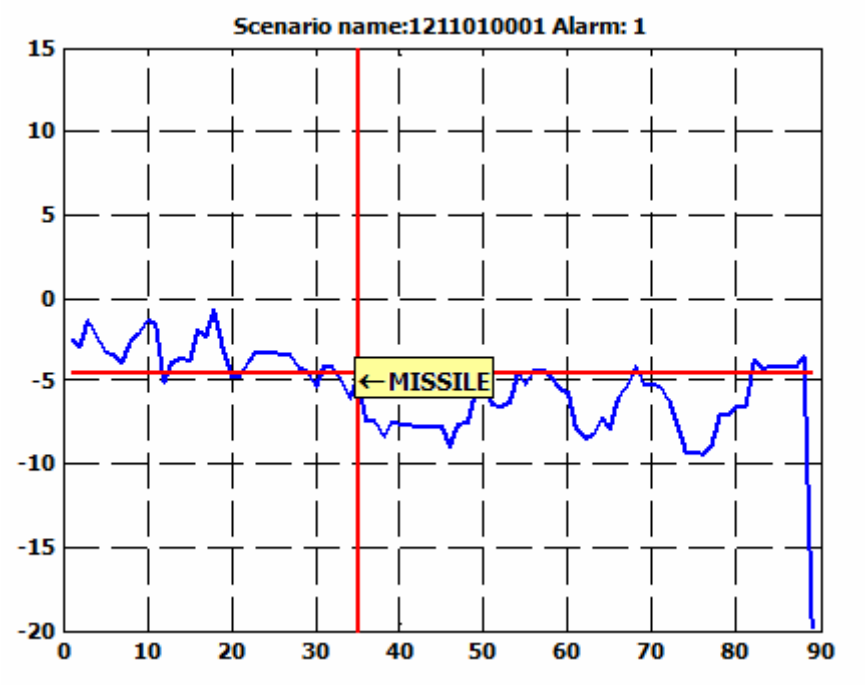

Figure 6-1: Example scenario based evaluation

6.2 Base Classifiers

Each classifier used in a MCS is called a base classifier. In this thesis, decision trees and stumps will be used as base classifiers.

6.2.1 Decision Trees

Decision trees can be used to classify patterns by asking a series of questions. In a typical tree structure, the starting node where the first question about a pattern is asked is called the root node. The root node is connected to other nodes by branches. The terminal nodes from where no further branches appear are called leaf nodes. At leaf nodes no more decisions are made and the label of the pattern is assigned. In Figure 6-2, a typical binary decision tree is shown.

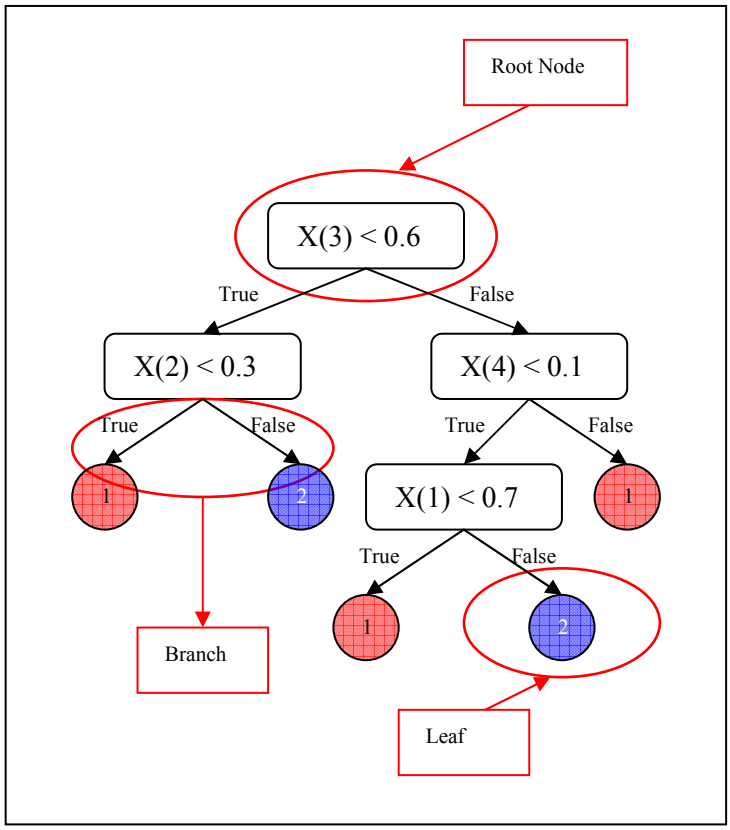

Figure 6-2: Binary decision tree example

Decision trees can be used as base learners for building multiple classifier systems due to their *unstable* nature [26]. Instability means that small variations in the training data might result in a very different tree structure. The instability can be used to achieve the desired diversity among the members of the ensemble.

The core ideas behind building a decision tree can be summarized as follows [1]:

1. **Number of splits**

   Each split in a tree corresponds to a decision and divides the training data for that node into two parts. By applying consecutive splits to the training data, the tree is constructed and threshold values for each node are found. The maximum number of splits can be limited to increase the speed of the training phase of the classifier. The results of applying this idea to a multiple classifier system can be found in Section 6.4 under the heading Boosting Results – 3.

2. Measuring the node impurity and selecting the best feature for a node

This part may be considered as the most important part of tree growing. The main aim is to determine the appropriate feature and threshold value for a node. Several impurity measures are proposed in the literature like entropy impurity, variance impurity, gini impurity and misclassification impurity. By using one of these measures, the best feature and the threshold value for a node is determined.

3. Stopping the split of the tree

The tree building process may be stopped by using one of the following criteria [26].

The splitting process can be stopped

- when the error in the validation set starts to increase,
- when the impurity gain in the child nodes is below a certain threshold,
- when the number of patterns to train a node is below a certain threshold,
- when the complexity of tree is above a certain measure,
- when further splitting is not beneficial.

4. Pruning the tree

Instead of applying stopping criteria, the tree may first be grown to the full size and then can be pruned to a smaller size to prevent the horizon effect. The horizon effect can be described as the lack of getting the further beneficial tree splits if the training process is stopped early.

5. Assignment of labels to leaf nodes

The label of each leaf is determined in the training process by the number of majority patterns (“t” or “f”) assigned to it.

In the experimental work the tree implementation of C4.5 [25] and its slightly modified java version J4.8 are used.

C4.5 algorithm measures impurity by using the entropy concept. After the complete tree is built the tree is pruned. Pruning removes the branches with little or no gain. The pseudo code for C4.5 is given below:

Node creation: Search all features in the training data and find the one d_{min} that separates the data with minimum entropy and find the corresponding threshold T. The node which has the condition is created

$$x_{d_{\min}} < T \tag{6-1}$$

The training data is split into two by using the condition in (6-1)

Algorithm:

Root node is formed.

Continue until the errors in the leaves are zero

1. Choose the leaf connected to the parent node that has the biggest error.
2. Construct a node by using the training data on this leaf.
3. Replace the leaf in 1 with the node created in 2

End

Prune the tree after the operation.

In the following sections related to the numerical results, decision trees and decision stumps are used as base classifiers. A decision stump can be described as a tree having only one split. In other words, there exists only a root node and two leaf nodes for a decision stump.

The output of a decision tree can be given as a confidence level. In Section 5.3.2, this is referred as output level information. The confidence level is calculated by using the error of the classifier on a leaf. When the decision tree is used in a MCS, the corresponding confidence level is multiplied by the weight of the decision tree assigned by the MCS. The weight assigning process for AdaBoost is explained in Section 5.3.1.2.

6.3 Bagging

Bagging method described in Section 5.3.1.1 is used in the experiments. The features are normalized with respect to the t-scenarios as explained in Section 4.3.

Bagging is reported to be more effective when used with unstable classifiers. The term unstable here refers to the fact that small changes in the training data lead to significant changes at the output of the classifier [24]. Decision trees are used as the base learner in experiments since they exhibit an unstable behavior. To demonstrate the effect of the unstable learner, bagging results for stumps, which are not unstable learners, are also analyzed. 4-Fold Cross Validation is used as training and testing procedure.

The results are presented in the following order:

1. Feature Based Evaluation for Bagging – Base Classifier: Decision Stumps
2. Feature Based Evaluation for Bagging – Base Classifier: Decision Trees

6.3.1 Bagging Result – 1

Stumps are not categorized as unstable classifiers. In fact, since they are decision trees with only one split (i.e. with two leaf nodes), training stumps on bootstrap replicates is expected to have no effect on the overall performance of the ensemble.

In Figure 6-3 and Figure 6-4 error curves for FT-Set and LF-Set for each fold can be found. In Figure 6-5 the average error curves for both sets are compared.

Boosting decision stumps has no significant effect on the performance of the ensemble, since their structure cannot be changed by sampling the data with replacement. The average error curves indicate that the LF-Set seems to perform better than the FT-Set in this experiment.

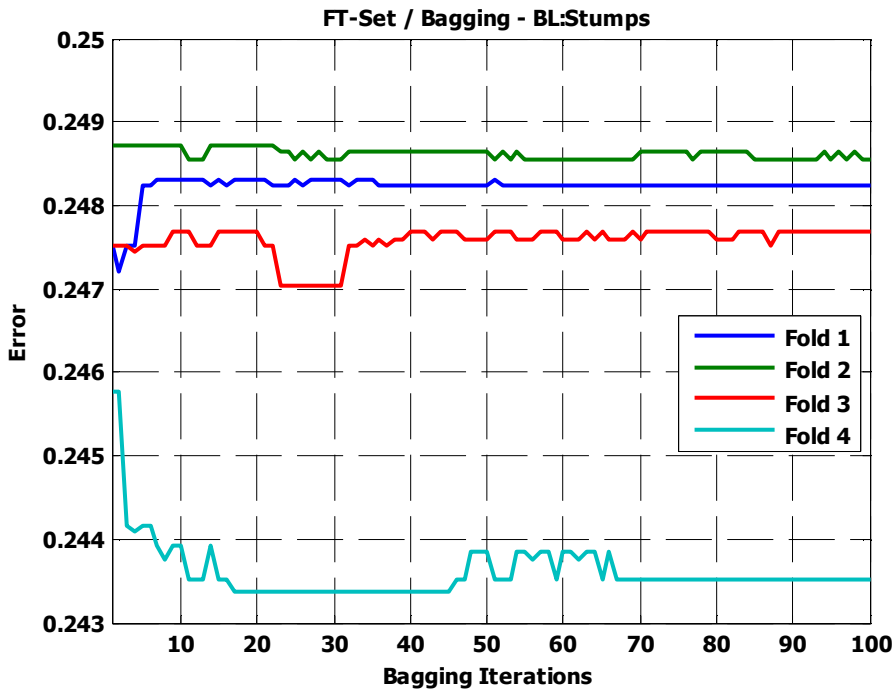

Figure 6-3: Iterations versus Error / (FT-Set Bagging, Base Learner: Stumps)

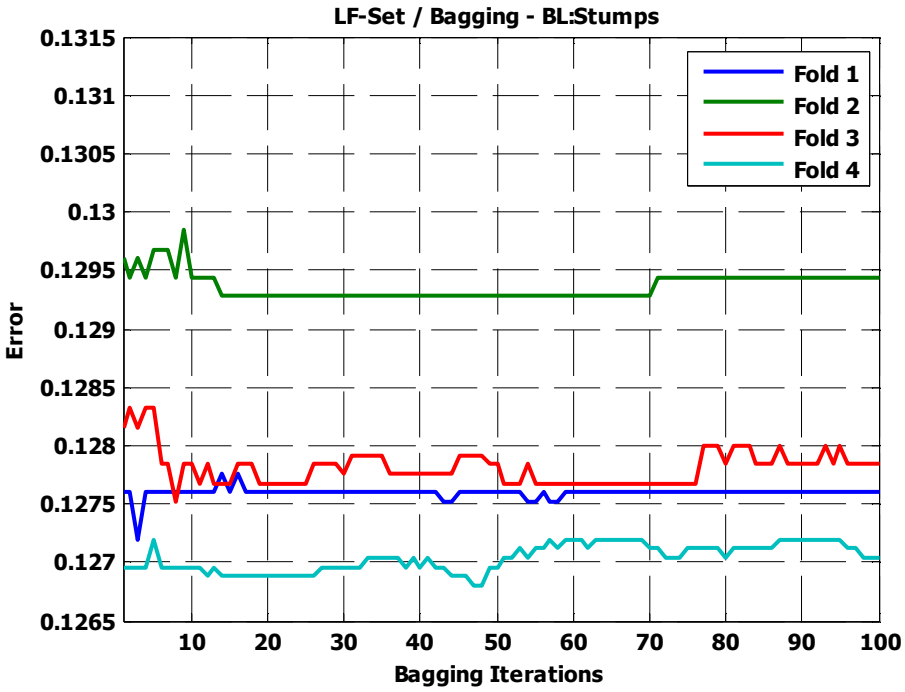

Figure 6-4: Iterations versus Error / (LF-Set Bagging, Base Learner: Stumps)

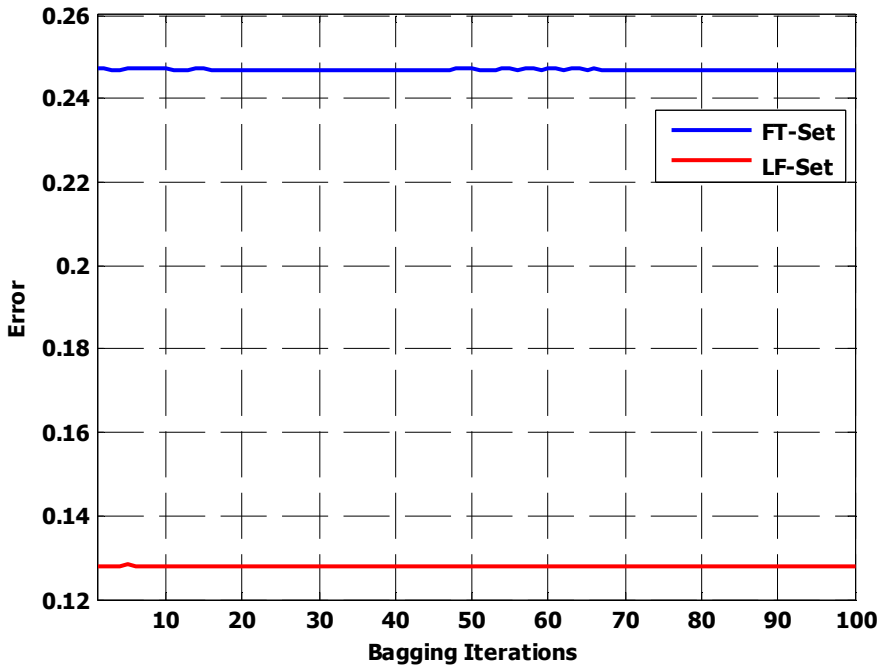

Figure 6-5: LF-Set FT-Set Comparison / (Bagging, Base Learner: Stumps)

6.3.2 Bagging Result – 2

The base classifier used in this experiment is Decision Trees. The decision tree implementation of J4.8 is used [25].

In Figure 6-6, Figure 6-7 and Figure 6-8 the results are presented. Examining the figures clearly signifies that by using *bagged* decision trees we can increase the overall performance of the ensemble, which is an anticipated result. The comparison of two data sets reveals that LF-Set is superior to the FT-Set.

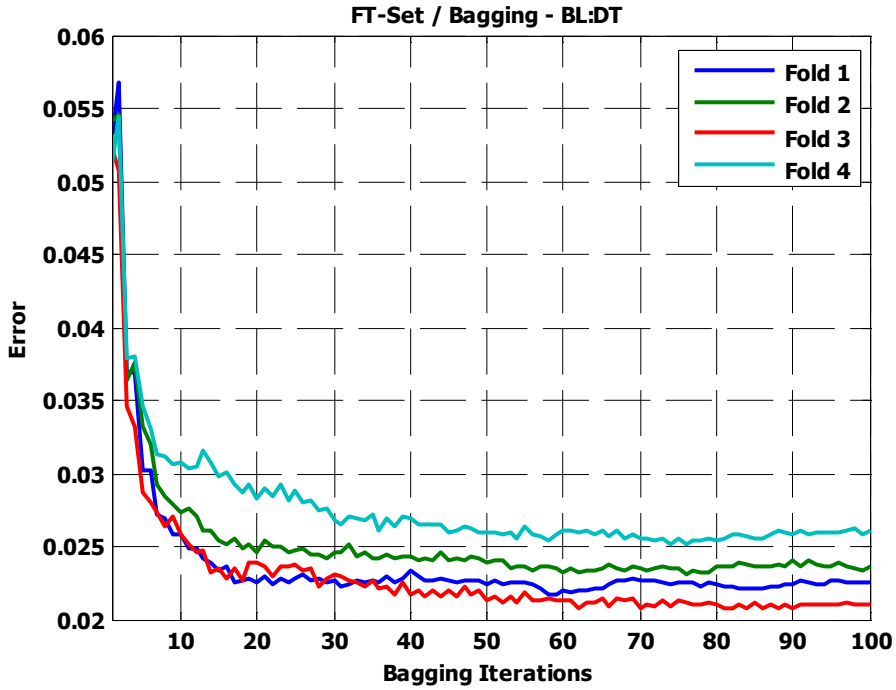

Figure 6-6: Iterations versus Error / (FT-Set Bagging, Base Learner: Decision Tree)

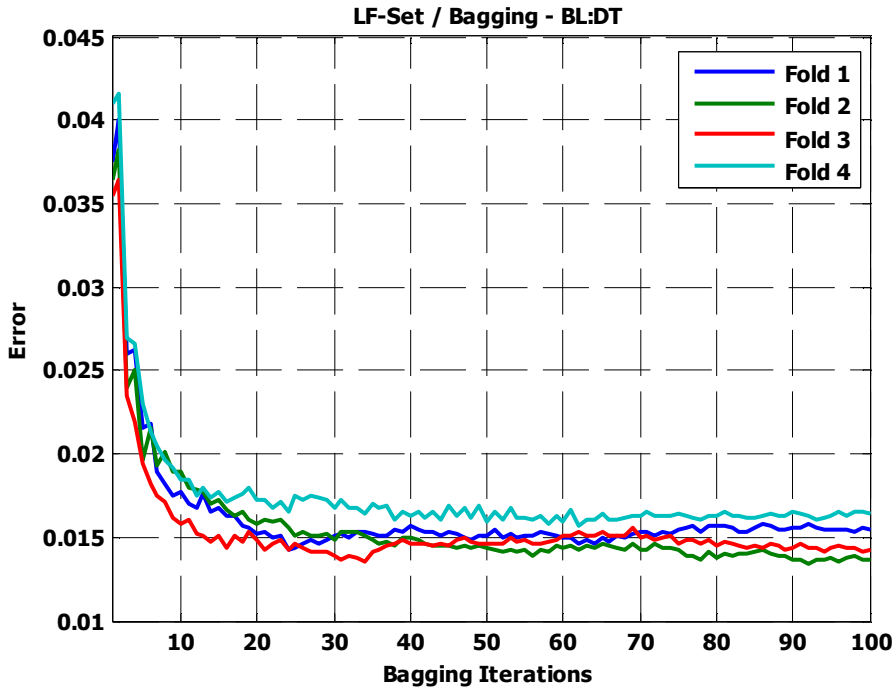

Figure 6-7: Iterations versus Error / (LF-Set Bagging, Base Learner: Decision Tree)

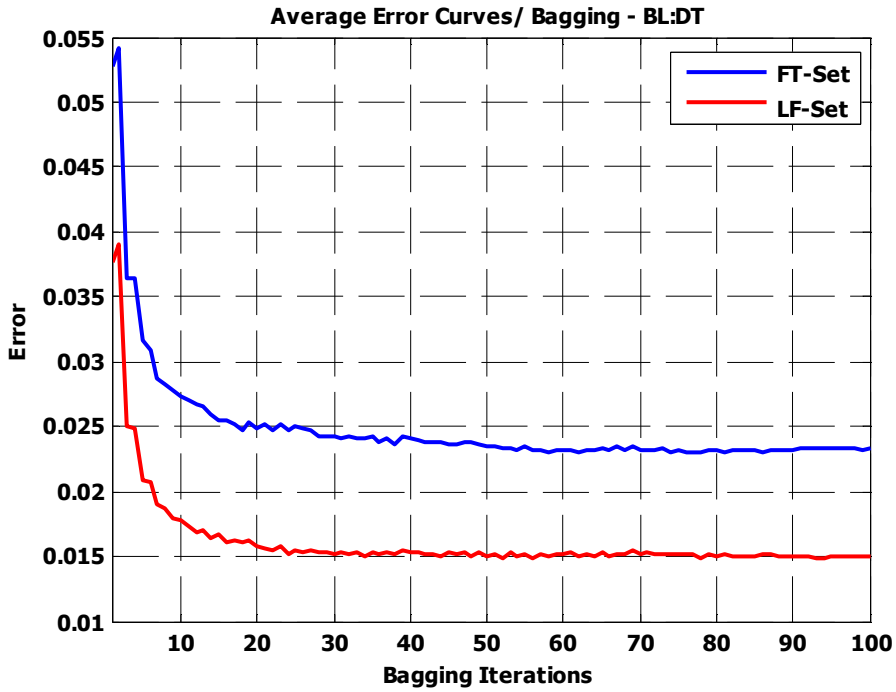

Figure 6-8: LF-Set FT-Set Comparison / (Bagging, Base Learner: Decision Tree)

6.4 AdaBoost

The AdaBoost algorithm used in this part is given in Table 5-1. Weak classifiers used with AdaBoost algorithm are stumps and decision trees. 4-Fold Cross Validation is used as testing and training procedure. The features are normalized with respect to the t-scenarios as explained in Section 4.3.

The experimental results are presented in the following order:

1. Feature Based Evaluation for AdaBoost – Weak Classifier: Stumps
2. Feature Based Evaluation for AdaBoost – Weak Classifier: Decision Trees
3. Scenario Based Evaluation for AdaBoost – Weak Classifier: Decision Trees

6.4.1 AdaBoost Result – 1

The weak classifier that has been used in the experiments is stumps. 4-fold cross validation is used to train and test the data set. In Figure 6-9 and Figure 6-10, the error versus boosting iteration curves for all folds are shown. The average values for both data sets can be seen in Figure 6-11.

Examining the results reveals that LF-Set performs better than the FT-Set in terms of error performance. Furthermore it is clear that increasing the number of boosting iterations improves the error rates for both feature sets.

It can be seen in Figure 6-11 that the error for LF-Set is 0.065 and FT-Set is 0.087 with 100 boosting iterations. The error curves seem to be asymptotically bounded around approximately 0.06 and 0.08.

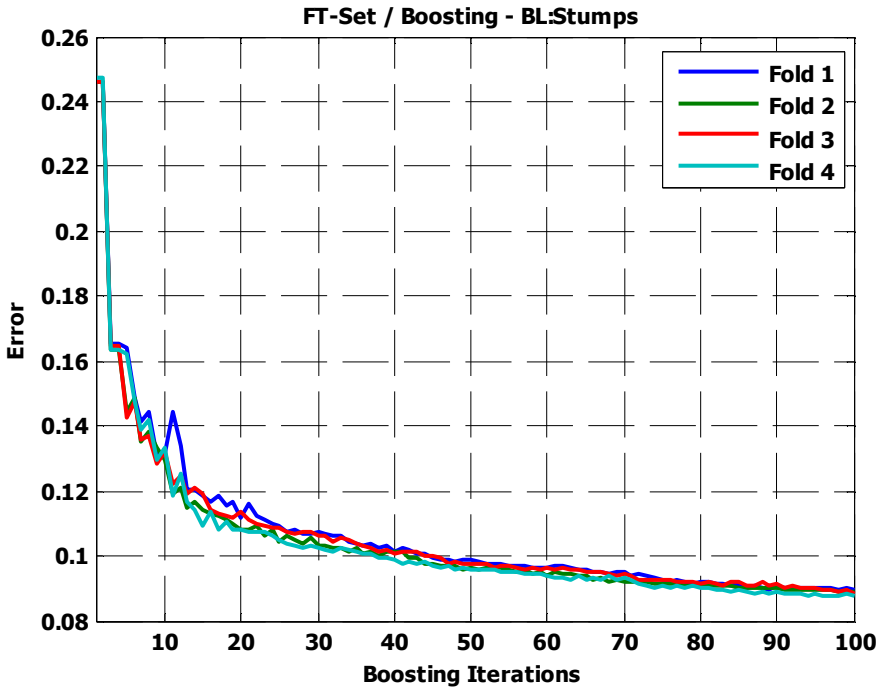

Figure 6-9: Iterations versus Error / (FT-Set AdaBoost, Base Learner: Stumps)

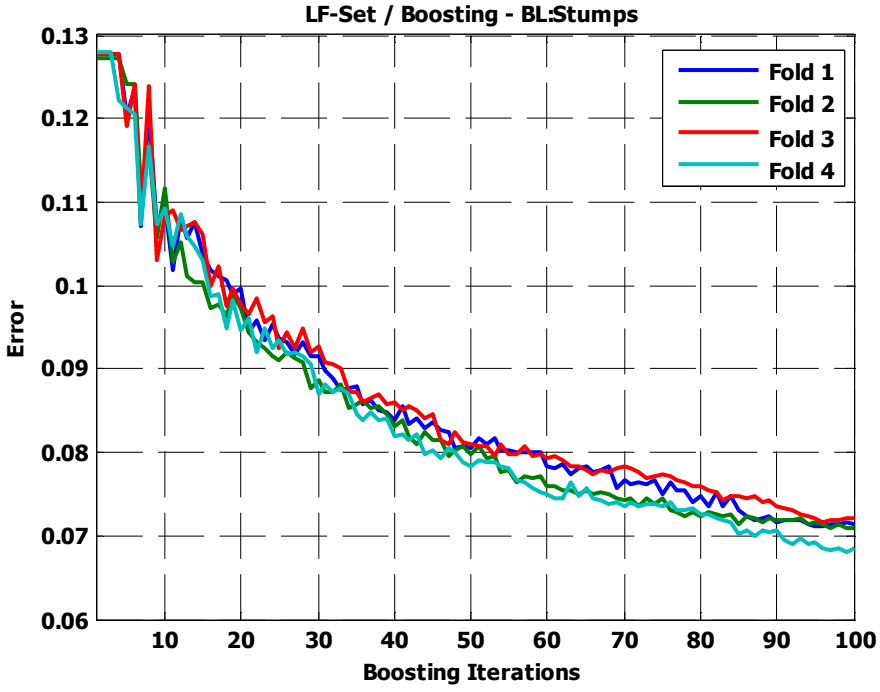

Figure 6-10: Iterations versus Error / (LF-Set AdaBoost, Base Learner: Stumps)

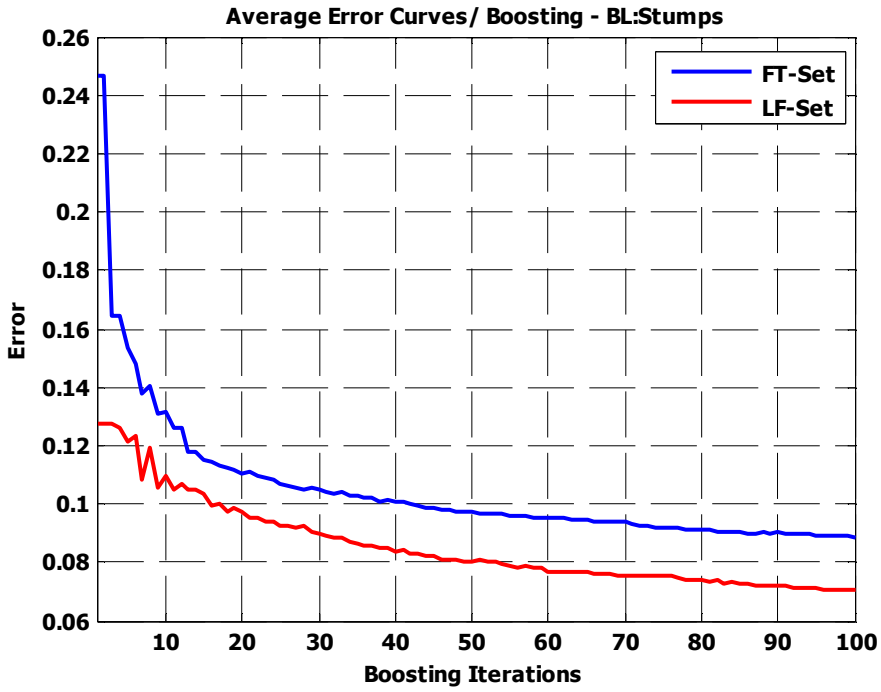

Figure 6-11: LF-Set FT-Set Comparison / (AdaBoost, Base Learner: Stumps)

6.4.2 AdaBoost Result – 2

In this part decision trees are used as weak classifier. The decision tree implementation is J4.8 [26].

The error is calculated by using 4-fold cross validation method. The results are presented in Figure 6-12, Figure 6-13 and Figure 6-14.

Boosting decision trees yields an error rate below 2% for FT-Set and below 1% for LF-Set. As the boosting iterations increase, the classifiers concentrate more on difficult samples and with each newly added classifier, the error rate decreases. Boosting above 50 iterations will seem to further increase the performance, however the trade off between memory requirements and performance increase should be considered before deciding to use more boosting iterations when designing a real system.

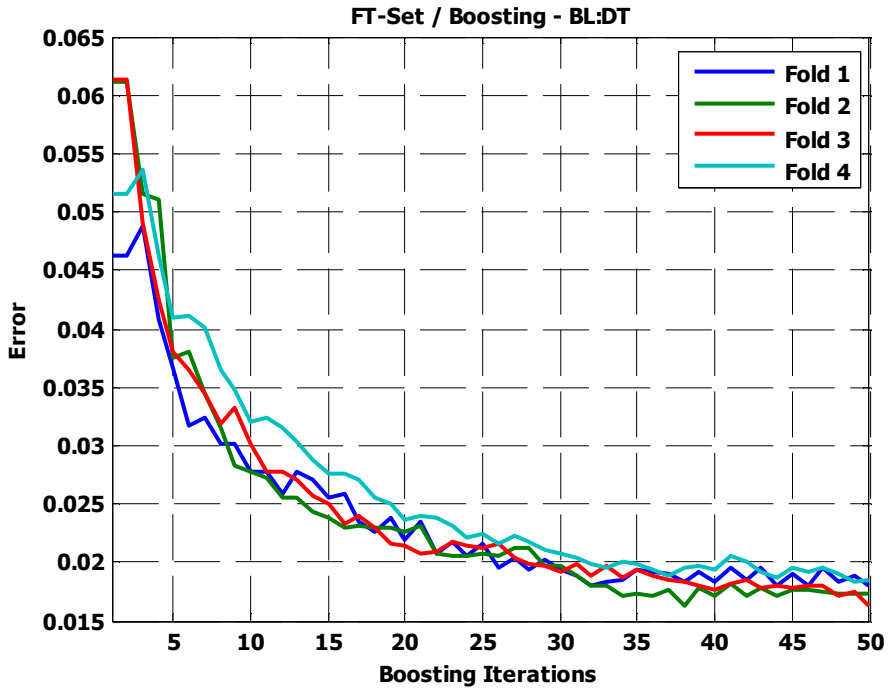

Figure 6-12: Iterations versus Error / (FT-Set AdaBoost, Base Learner: Decision Trees)

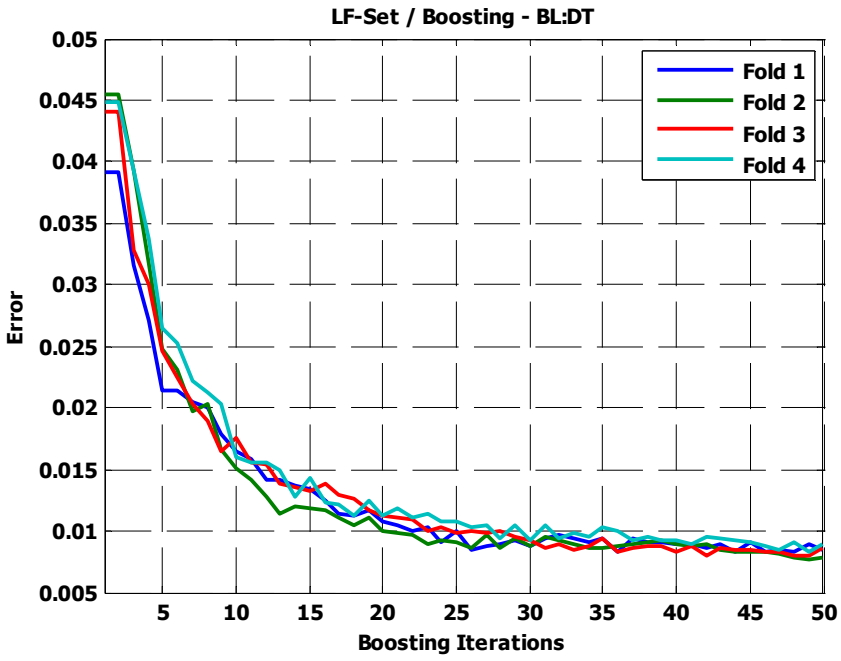

Figure 6-13: Iterations versus Error / (LF-Set AdaBoost, Base Learner: Decision Trees)

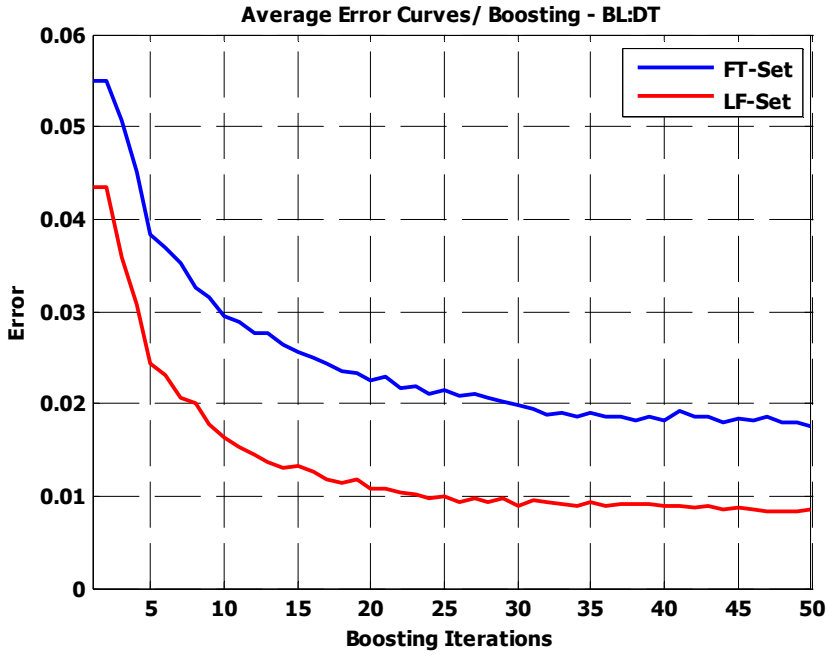

Figure 6-14: LF-Set FT-Set Comparison / (AdaBoost, Base Learner: Decision Tree)

6.4.3 AdaBoost Result – 3

In this part the Receiver Operating Characteristics (ROC) of the AdaBoost is examined. In order to decrease the computational burden, the AdaBoost algorithm is iterated 20 times and the trees are allowed to make 10 maximum splits.

The AdaBoost algorithm with decision trees is modified to output confidence levels instead of class labels so that the effect of applying different score thresholds on the performance of the system can be examined.

The scenario based training and evaluation are explained in 6.1.2. According to this methodology, ROCs for probability of declaration, reaction time and number of false alarms are obtained for all folds. Then the mean of these folds are presented as the final result.

In Figure 6-15, Figure 6-16 and Figure 6-17 the ROC for FT set can be seen. The ROCs are obtained for all folds. In Figure 6-18 Figure 6-19 and Figure 6-20 the average values of these curves are presented to summarize the characteristics of FT-

Set. From the figures it is clearly seen that as the score value increases, the probability of declaration increases, warning time decreases and number of false alarms increases. The optimization for score threshold can be made by using one of the ROC curves in Figure 6-18 Figure 6-19 and Figure 6-20. For example if you allow 4 false alarms you will get 99.5 POD and 39.09 frames reaction time on the average. It is obvious that if higher POD and lower warning time are desired than a relatively small amount of false alarms should be tolerated.

The results for LF-Set are presented in Figure 6-21, Figure 6-22, Figure 6-23, Figure 6-24, Figure 6-25 and Figure 6-26 by using the same order explained for FT-Set. The same trade-off in the ROC curves are also applicable for LF-Set.

Comparing the average ROC curves for LF-Set and FT-Set reveals that LF-Set performs better than the FT-Set in terms of scenario based evaluation. Because the ROC for mean reaction time and false alarm of the LF-Set are below their ROC counterparts of the FT-Set, which in turn means that they result in better performance for the same score value. Furthermore the ROC for probability of declaration of LF-Set starts to decrease at lower score values compared to the other one.

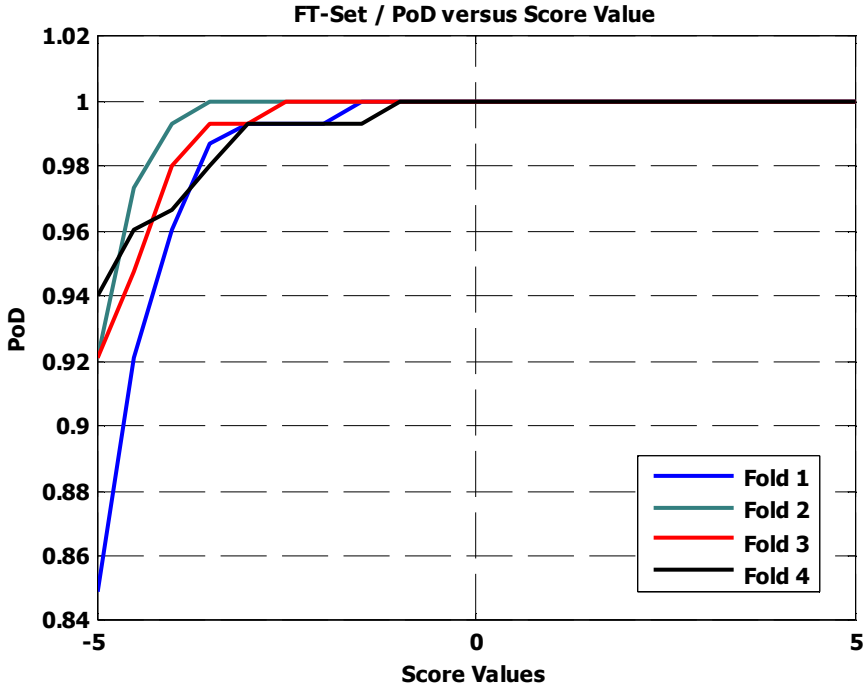

Figure 6-15: FT-Set, Probability of Declaration vs. Score Value (All Folds)

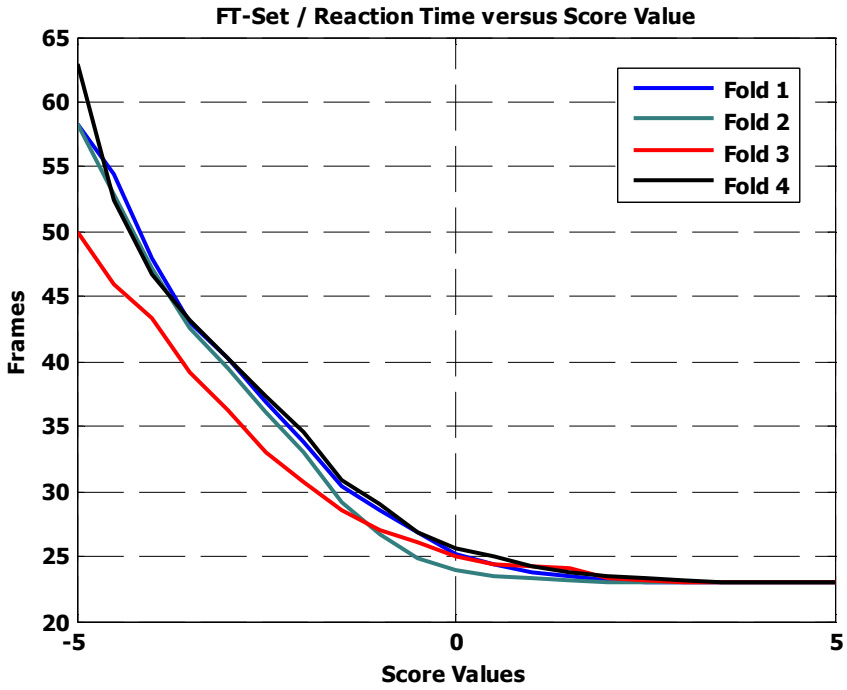

Figure 6-16: FT-Set, Reaction Time vs. Score Value (All Folds)

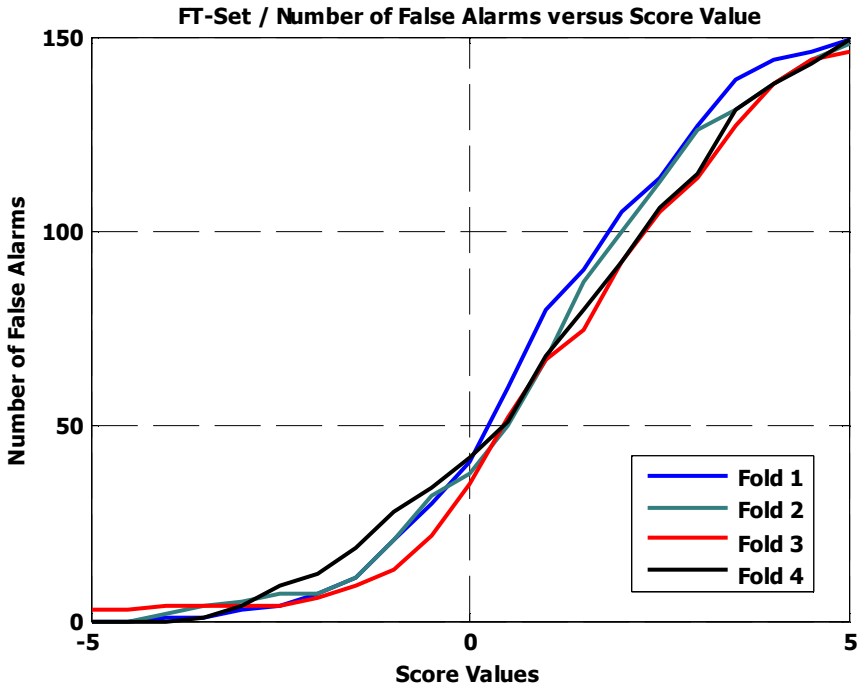

Figure 6-17: FT-Set, False Alarm vs. Score Value (All Folds)

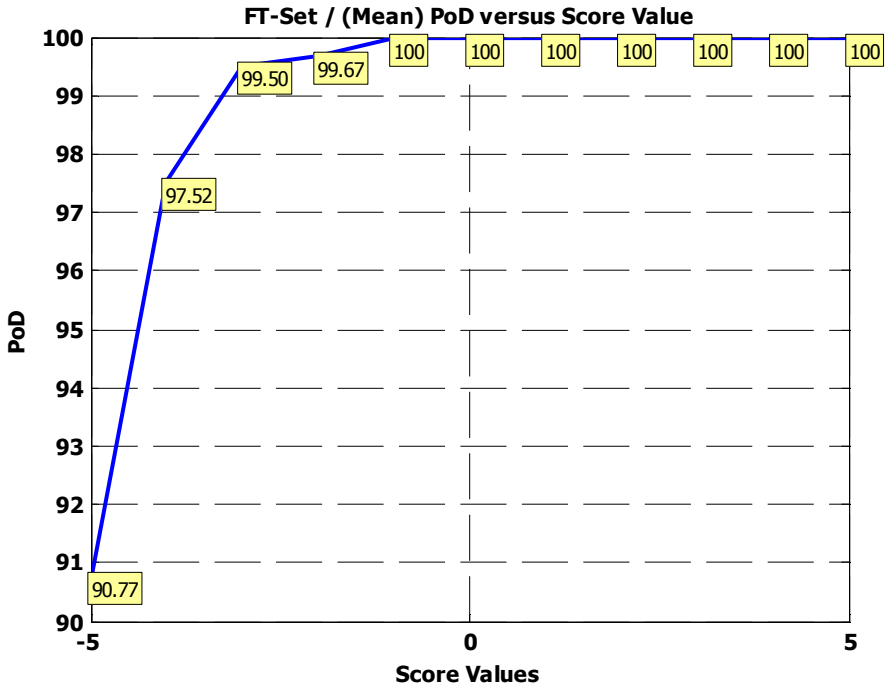

Figure 6-18: FT-Set, Probability of Declaration vs. Score Value (Mean Value)

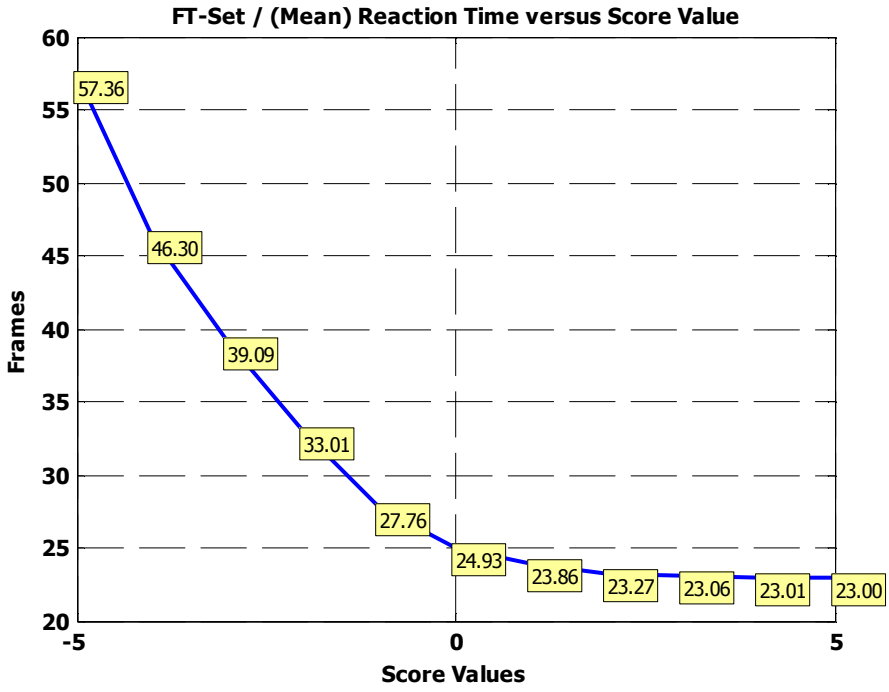

Figure 6-19: FT-Set, Reaction Time vs. Score Value (Mean Value)

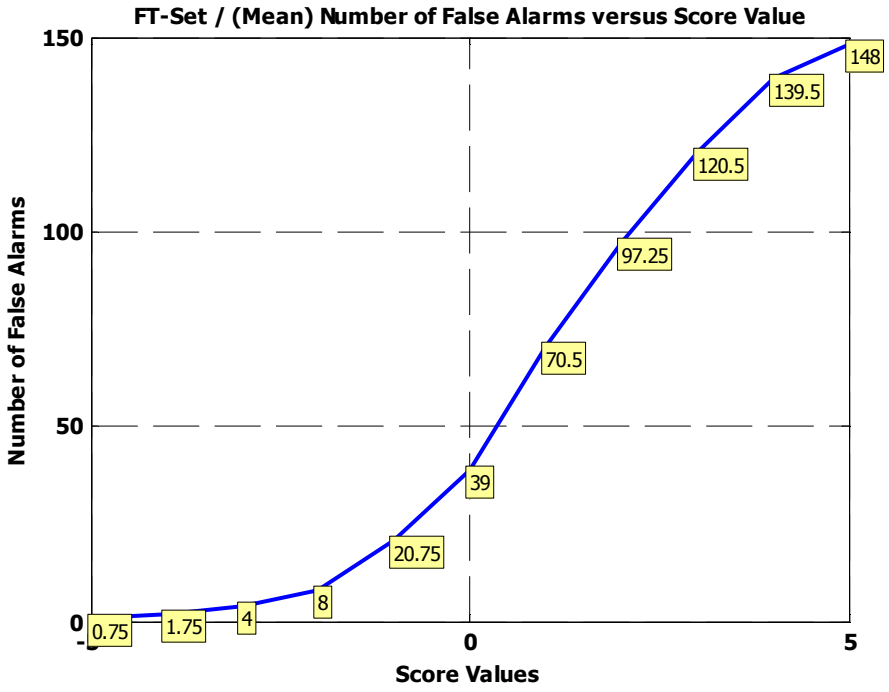

Figure 6-20: FT-Set, False vs. Score Value (Mean Value)

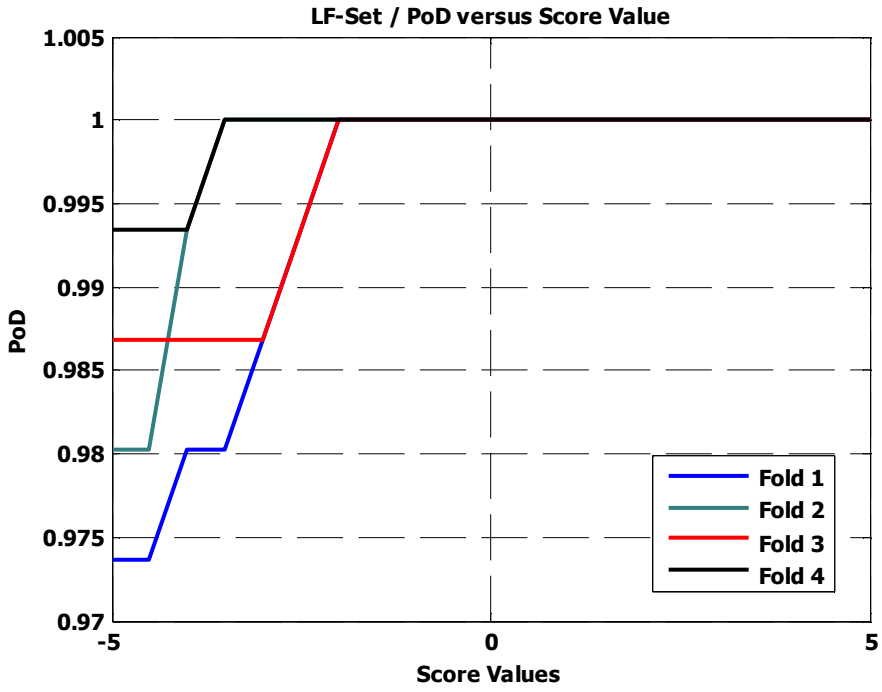

Figure 6-21: LF-Set, Probability of Declaration vs. Score Value (All Folds)

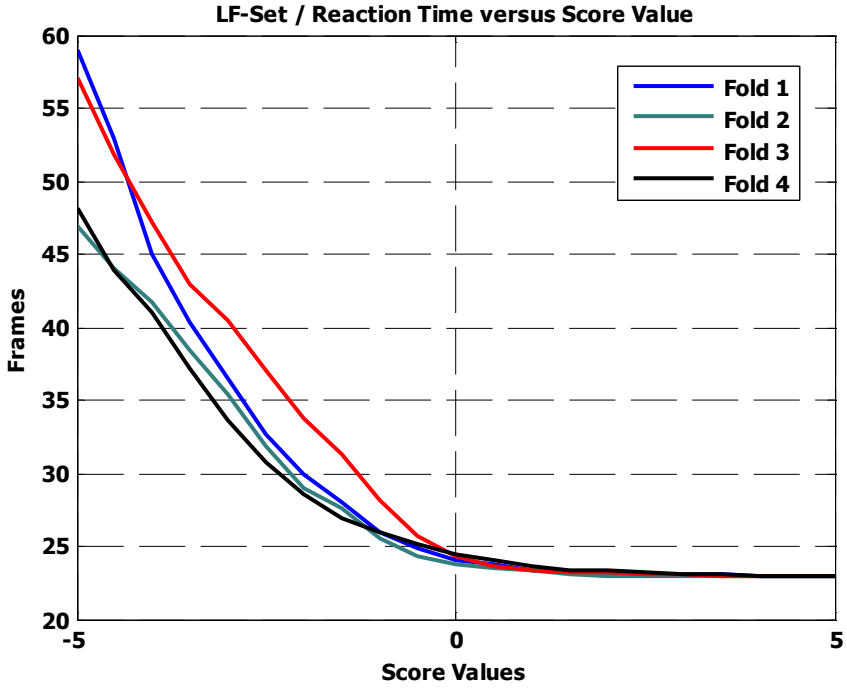

Figure 6-22: LF-Set, Reaction Time vs. Score Value (All Folds)

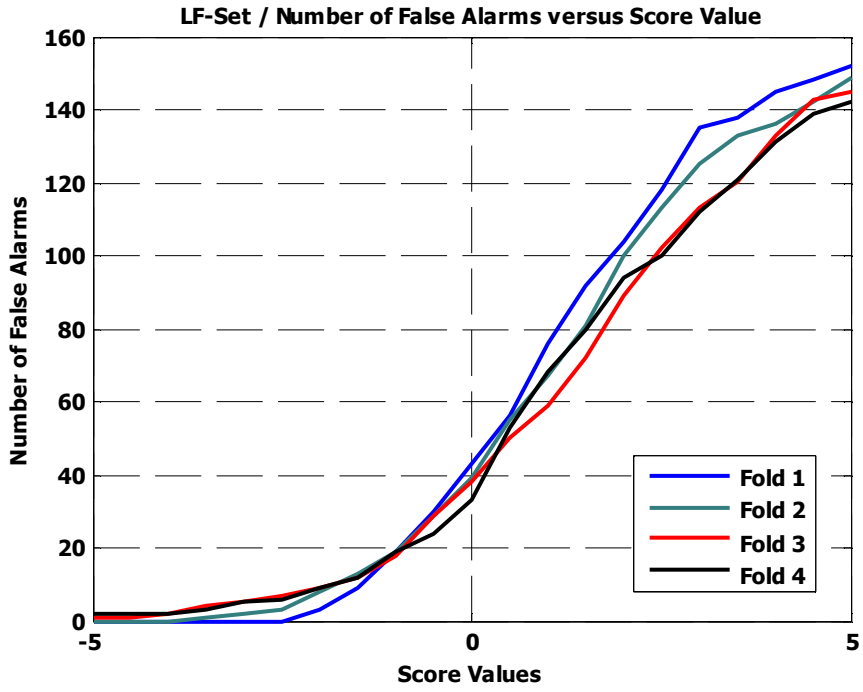

Figure 6-23: LF-Set, False vs. Score Value (All Folds)

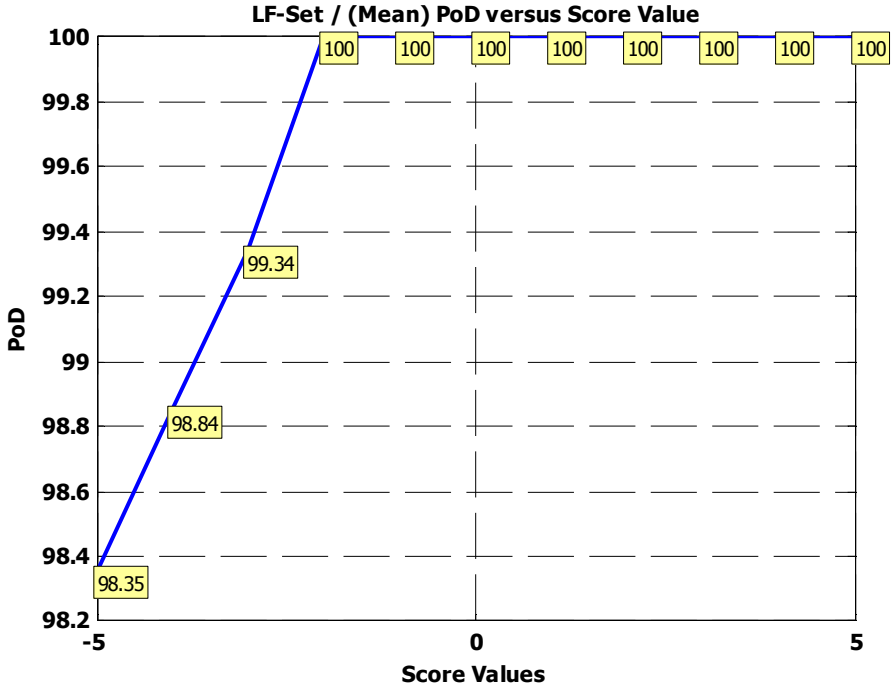

Figure 6-24: LF-Set, Probability of Declaration vs. Score Value (Mean Value)

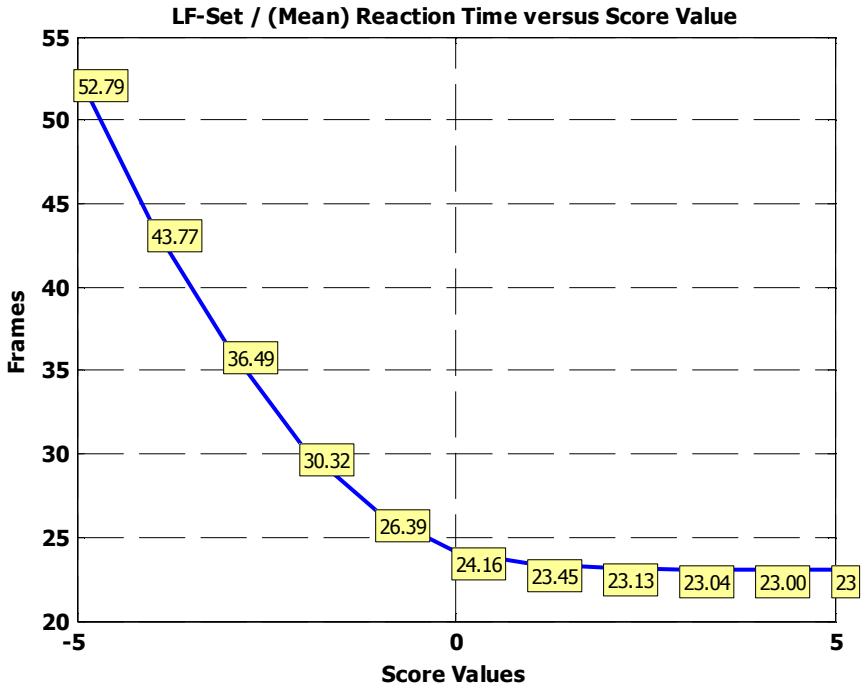

Figure 6-25: LF-Set, Reaction Time vs. Score Value (Mean Value)

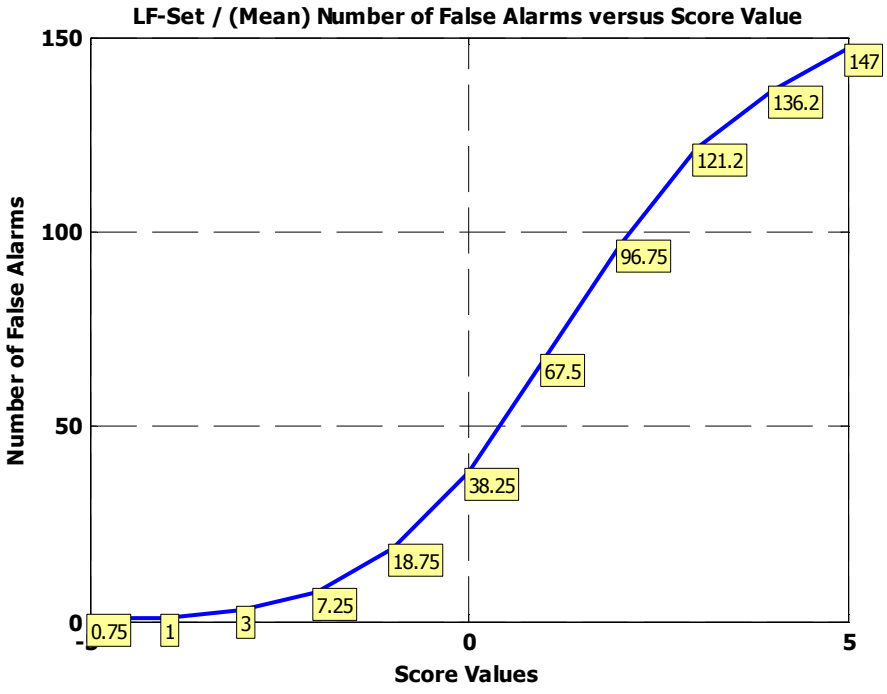

Figure 6-26: LF-Set, False Alarm vs. Score Value (Mean Value)

6.5 Comparison of AdaBoost and Bagging

In this part, the feature-based results of AdaBoost algorithm and Bagging are summarized and presented in order to compare the performance of both algorithms.

The error versus iteration curves of AdaBoost and Bagging with stumps used as base learner can be found in Figure 6-27. The following conclusions may be drawn from the figure:

- Bagging stumps does not increase the performance for both sets. The reason for this is the fact that the stumps are not unstable classifiers.
- Boosting stumps for both sets increase the performance of the ensemble.
- LF-Set performs better than the FT-Set.

The error versus iterations curves of AdaBoost and Bagging with decision trees used as base learner can be found in Figure 6-28. The following conclusions may be drawn from the figure:

- For both sets, increasing the iterations for bagging and boosting decreases the error rate.
- Bagging performs better than boosting at the first iterations; however as the iterations increase, boosting concentrates more on difficult patterns and outperforms bagging.
- LF-Set performs better than the FT-Set for each case.

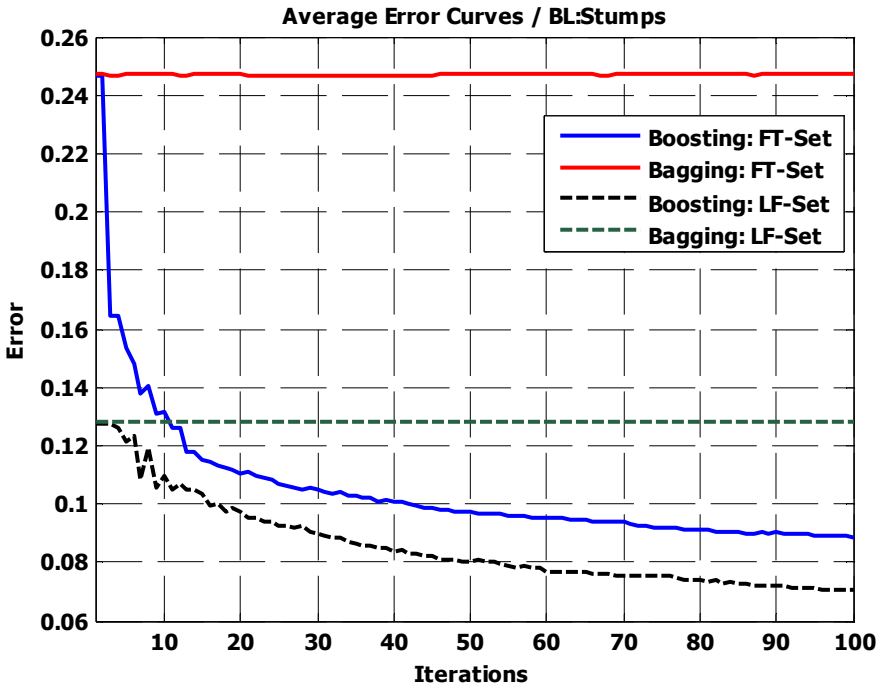

Figure 6-27: Comparison of Bagging and Boosting / Base Learner: Stumps

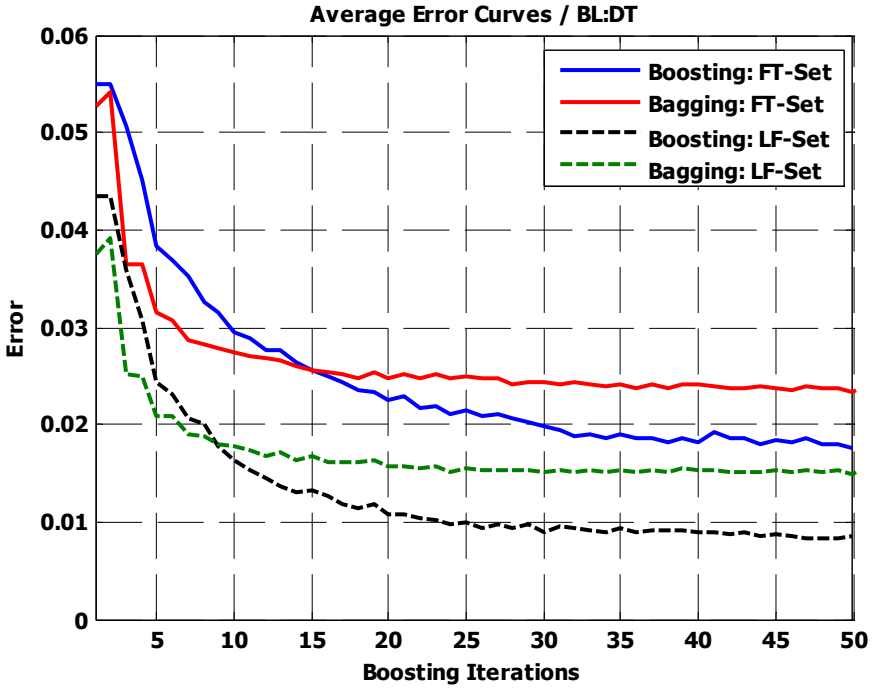

Figure 6-28: Comparison of Bagging and Boosting / Base Learner: Decision Tree

6.6 Feature Grouping

The main aim of this section is to investigate the effect of combining the decision of classifiers trained on features extracted from each primary time series. The general architecture is summarized in Figure 6-29.

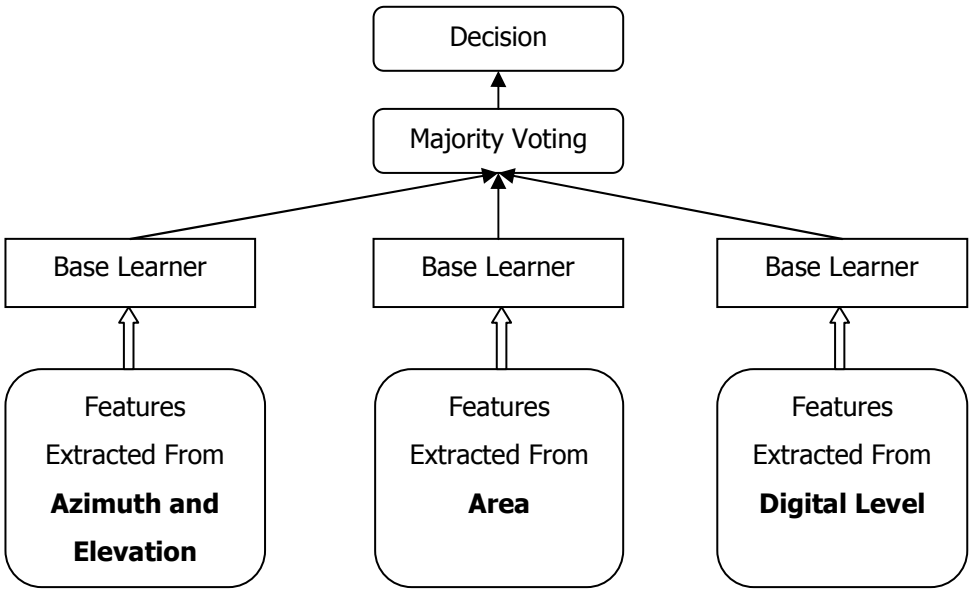

Figure 6-29: Feature Grouping Setup

The features are grouped in a natural manner. The first group consists of features coming from Azimuth and Elevation. The second group is composed of features coming from Area and the last group is composed of features coming from Digital Level. Decision trees and stumps are used as base learners. Majority voting is applied as the combining rule. Results are organized in the following manner:

1. Base Learner Stumps

2. Base Learner Decision Trees

6.6.1 Feature Grouping Result – 1

The results in Figure 6-30 and Figure 6-31 indicate that the stumps as a base learner did not yield good performance. The results for all folds and their average values are presented in the figures.

The output of the ensemble is slightly better than the best classifier with FT-Set and worse than the best classifier in LF-Set. This result is mainly due to the lack of classifier diversity that cannot be achieved with stumps. The individual classifiers should correctly classify different patterns in order the ensemble decision could achieve a better performance than the best classifier which, in the case with stumps, cannot be observed.

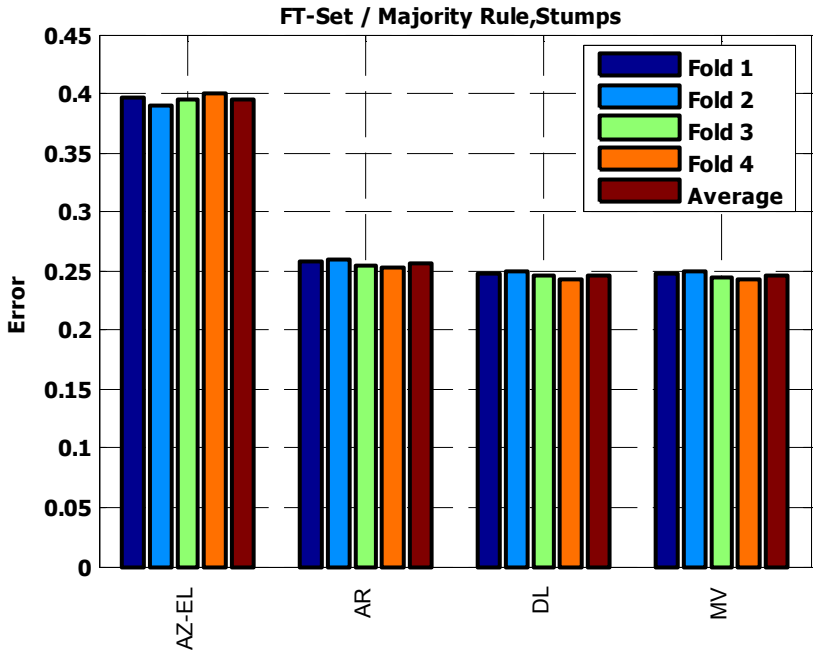

Figure 6-30: FT-Set Feature Grouping / Base Learner: Stumps

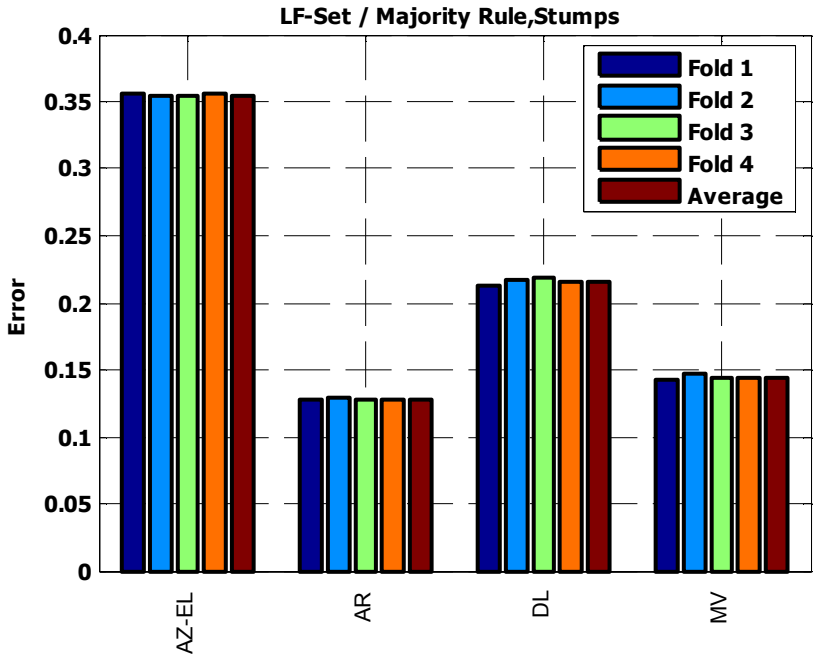

Figure 6-31: LF-Set Feature Grouping / Base Learner: Stumps

6.6.2 Feature Grouping Result – 2

Feature grouping seems to be effective when used with decision trees as a base learner. In Figure 6-32 and Figure 6-33, the results for both sets can be examined. The overall classification error of the system is better than the best individual classifier in the set. It shows that the patterns that are misclassified by the best classifier are correctly classified by the other two classifiers in the set, which, in turn helps the system to output a better performance.

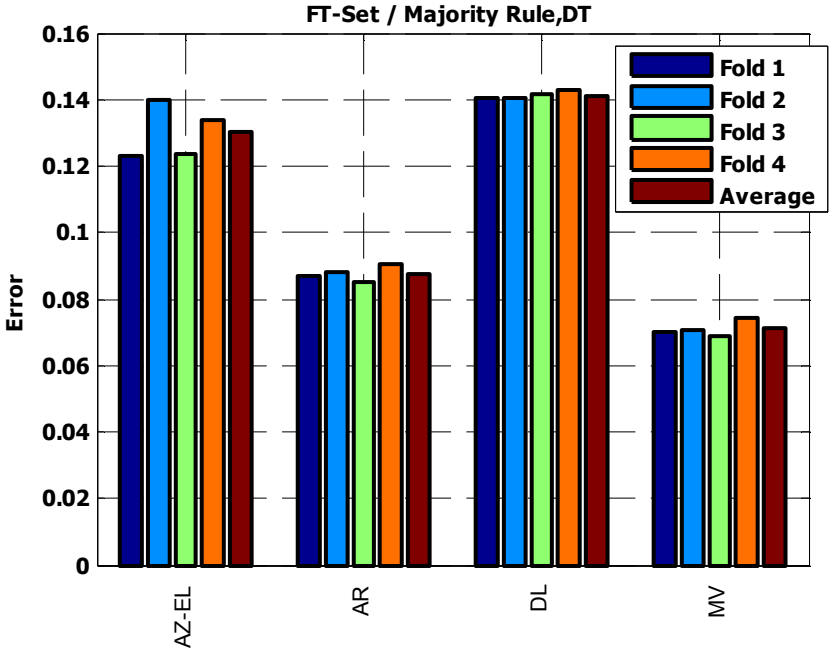

Figure 6-32: FT-Set Feature Grouping / Base Learner: Decision Tree

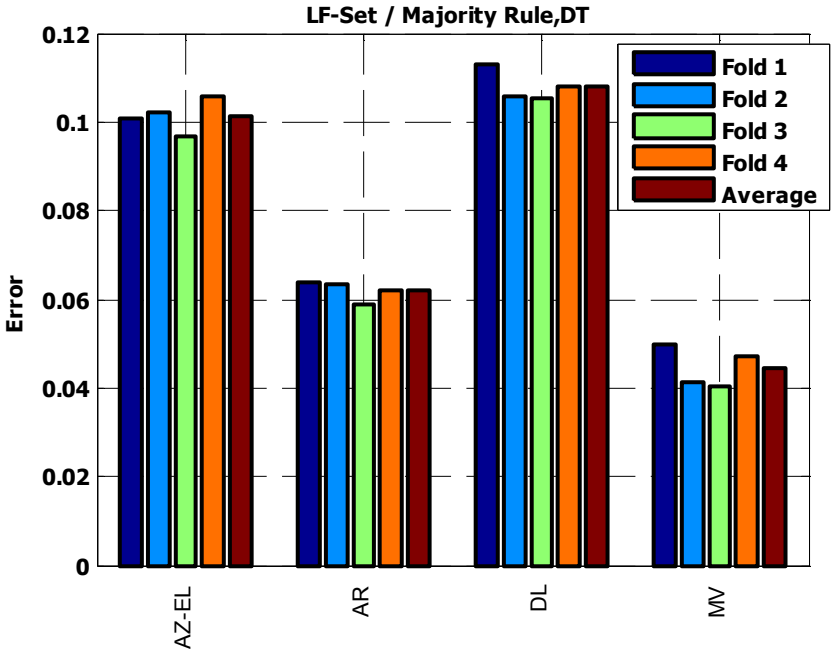

Figure 6-33: LF-Set Feature Grouping / Base Learner: Decision Tree

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this thesis, a decision making algorithm for a Generic Missile Warner based on Multiple Classifier Systems is proposed. For this purpose, simulation data are generated. A method for generating a wide range of real life scenarios is proposed. Two feature extraction methods, namely Fourier Transform and linear line fit, are applied.

For the classification part, experiments are carried out by using the MCS methods bagging, AdaBoost and feature grouping. Decision trees and stumps are used as base learners. 4-fold cross validation method is used in evaluating the performance of the classifiers.

Two kinds of evaluation methods are proposed; one is feature based evaluation and the other is scenario based evaluation. In the feature based evaluation, AdaBoost algorithm with decision trees as base learners performed better than the other methods. The scenario based evaluation is carried out for only AdaBoost and corresponding ROC curves for probability of declaration, number of false alarms and reaction time are calculated and very promising results are achieved.

In all classification methods, the features extracted by using linear fitting outperformed the features extracted by using the Fourier Transform.

As a future work, effect of different base learners to the MCS should be investigated and the experimental results should be tested with real data. Also different feature extraction methods using statistical approaches should be tested.

REFERENCES

[1] Duda, O., Hart, E., Stork, G., "*Pattern Classification"*, Wiley-Interscience Publication, 2001

[2] Olszewski, R. T.,"*Generalized Feature Extraction for Structural Pattern Recognition in TimeSeries Data*"**,** February, 2001

[3] Zhang, H., Ho T. B., and Lin, M. S., "A Non-parametric Wavelet Feature Extractor for Time Series Classification" PAKDD 2004, LNAI 3056, pp. 595–603, 2004.

[4] Avan, S., M. Sc. Thesis, Middle East Technical University, "*Feature Set Evaluation for a Generic Missile Detection System*", February 2007.

[5] Weisstein, E. W. *"Least Squares Fitting." From* MathWorld--*A Wolfram Web Resource. http://mathworld.wolfram.com/LeastSquaresFitting.html*

[6] Tumer, K., Ghosh, K. J. "*Analysis of Decision Boundaries in Linearly Combined Neural Classifiers*" Pattern Recognition 29 p 341-348, 1996

[7] Dietterich, T.G., "*Ensemble methods in machine learning*", Multiple Classifier Systems, Springer-Verlag, LNCS, Vol. 1857, 2000

[8] Hansen, L., Salamon, P., "*Neural network ensembles*" IEEE Trans Pattern Analysis and Machine Intelligence, Vol. 12, p 993–1001, 1990

[9] Kuncheva, L. I., "*A Theoretical Study on Six Classifier Fusion Strategies*" IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 24, No. 2, February 2002

[10] Breiman, L., "*Bagging Predictors*" Machine Learning, Vol. 24, p. 123–140, 1996

[11] Efron, B., "*Bootstrap Methods: Another Look at the Jackknife*" The Annals of Statistics, Vol. 7, No. 1, pp. 1-26, Jan. 1979

[12] Schapire, R. E., "*The strength of weak learnability*" Machine Learning, 5(2), p. 197-227, 1990.

[13] Freund, Y., Schapire, R. E., "*A decision-theoretic generalization of on-line learning and an application to boosting*", Journal of Computer and System Sciences, Vol. 55, p. 119-139, 1997.

[14] Ho, T.K., "*Random subspace method for constructing decision forests*" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, p. 832–844, 1998.

[15]Cherkauer, K. J. "*Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks*" Working Notes, Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms Workshop, Thirteenth National Conference on Artificial Intelligence Portland, 1996.

[16] Xu, L., Krzyzak, A., and Suen, C.Y., "*Methods of combining multiple classifiers and their applications to handwriting recognition*" IEEE Transactions on Systems, Man and Cybernetics, Vol. 22, No. 3, p. 418–435, 1992.

[17] Kittler, J., Hatef, M., Duin, Robert P.W., and Matas, J. *"On Combining Classifiers"* IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 20, No. 3, March 1998

[18] Huang, Y.S., and Suen, C.Y., "*Behavior-knowledge space method for combination of multiple classifiers*" Proc. of IEEE Computer Vision and Pattern Recog., p. 347–352, 1993

[19] Raudys, S., Roli, F., "*The Behavior Knowledge Space Fusion Method: Analysis of Generalization Error and Strategies for Performance Improvement*", 4th Int.

Workshop on Multiple Classifier Systems (MCS 2003), Guildford, United Kingdom, T. Windeatt and F. Roli Eds., LNCS 2709, p. 55-64, June 11-13 2003,

[20] Wolpert, D.H., "*Stacked generalization*", Neural Networks, Vol. 5, No. 2, p. 241–259, 1992.

[21] Kohonen, T., *"The Self-Organizing Map",* Proceeding of IEEE, Vol. 78, No. 9, September 1990

[22] SOM Toolbox Team, Helsinki University of Technology P.O. Box 5400, FIN-02015 HUT, Finland http://www.cis.hut.fi/projects/somtoolbox/

[23] Lam, L., "*Classifier combinations: implementations and theoretical issues*" In J. Kittler and F. Roli, editors, Multiple Classifier Systems, Lecture Notes in Computer Science, 1857, p 77-86. Springer, Cagliari, Italy, 2000.

[24] Roli, F., Giacinto, G., "*Design of multiple classifier systems*" In: H. Bunke and A. Kandel, Editors, Hybrid Methods in Pattern Recognition, World Scientific ,pp. 199–226, 2002

[25] Quinlan, R., "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[26] Kuncheva, L. I., "*Combining Pattern Classifiers*", Wiley-Interscience Publication, 2004

APPENDIX A

SCENARIO PARAMETER LIST

Table A-1: Scenario Parameters

	Platform properties				Position of the missile		Other	
Scenario Number	Altitude (m)	Velocity (m/s)	Heading Angle (degrees)	Climb Angle (degrees)	x-axis (m)	y-axis (m)	Atmospheric Attenuation (1/km)	Firing Angle (degrees)
001	907.562	20.401	146.440	15.782	-608.897	750.120	1.007	43.209
002	915.610	17.133	-177.618	15.223	-1589.270	-1530.812	1.154	22.535
003	443.620	66.050	137.361	17.815	-655.228	1807.920	1.213	12.990
004	700.132	55.919	-148.259	1.247	2331.399	-1418.004	0.826	14.390
005	871.726	39.643	-124.453	-10.769	947.724	377.690	0.888	40.512
006	697.844	57.272	-78.498	-3.509	-826.766	1688.352	0.933	20.365
007	921.656	23.171	-0.158	11.344	1062.372	-2101.130	0.808	21.378
008	801.794	66.164	100.752	-12.365	-428.152	-2913.260	0.800	15.232
009	307.762	36.041	25.874	11.681	-481.580	195.219	0.960	30.636
010	919.265	43.587	143.602	-17.927	1851.615	-990.573	1.441	23.642
011	857.921	27.859	-119.762	-18.875	2732.219	1081.744	0.883	16.275
012	951.275	54.416	149.438	-9.732	2313.709	2520.257	1.388	15.539
013	258.713	61.392	-148.587	9.151	-312.662	907.492	0.940	15.085
014	625.159	50.704	-173.954	-1.185	2317.956	-2315.829	0.836	10.802
015	727.638	23.582	161.232	7.771	-1759.161	328.574	1.054	22.126
016	646.286	60.187	141.374	13.674	-2214.860	-1865.076	1.403	12.582
017	596.460	20.653	83.362	-15.330	1476.249	1858.735	0.941	14.105
018	469.715	46.746	-11.115	-16.509	1972.305	1115.668	1.296	11.711
019	975.587	14.702	-71.621	-3.553	-1581.065	-1829.668	0.914	21.971
020	925.285	54.403	5.353	0.883	-2382.489	2981.283	0.701	13.629
021	700.194	31.469	-176.257	1.811	54.662	-1519.326	0.987	24.729
022	873.384	3.860	-65.757	11.337	2834.400	518.784	0.736	16.862
023	552.428	10.630	-21.765	1.906	-629.182	-610.368	1.236	32.219
024	617.880	39.235	-147.253	-9.966	-314.646	827.765	1.301	34.905
025	994.096	74.576	-145.982	18.142	-2023.226	2822.731	1.268	15.973
026	509.651	33.683	50.148	11.502	-1380.033	2063.894	0.900	11.600
027	860.882	14.575	-155.574	4.414	1209.317	-2330.294	1.292	18.155
028	678.267	64.979	112.619	-16.423	1387.672	2423.139	0.777	13.653
029	256.550	19.302	83.008	-18.380	-452.863	241.291	1.062	26.564
030	853.356	66.404	-3.924	10.429	2490.648	2405.850	1.185	13.844
031	861.501	24.077	-41.569	6.030	1904.223	1597.597	0.823	19.116
032	351.890	51.720	-177.709	-8.684	831.604	552.447	0.999	19.416
033	991.161	9.859	84.444	-13.735	-392.116	1993.459	0.960	26.006
034	260.990	44.554	-80.933	-14.718	1197.929	-84.571	0.988	12.261
035	549.821	53.527	17.075	4.362	2178.809	-715.824	0.898	13.482
036	470.589	74.239	142.636	14.028	-1459.248	-1287.027	1.184	13.596
037	761.116	39.401	167.425	-0.954	2969.433	-56.528	1.324	14.374
038	815.018	31.047	-16.733	-14.686	1551.038	391.422	1.103	26.999
039	838.446	17.636	128.125	16.189	-1247.881	1355.219	1.219	24.471
040	544.949	38.052	102.109	-14.774	-2691.866	765.039	0.841	11.019
041	653.003	30.193	115.128	-7.805	-1083.761	1709.879	1.332	17.878

Table A-1: Scenario Parameters (continued)

	Platform properties				Position of the missile		Other	
Scenario Number	Altitude (m)	Velocity (m/s)	Heading Angle (degrees)	Climb Angle (degrees)	x-axis (m)	y-axis (m)	Atmospheric Attenuation (1/km)	Firing Angle (degrees)
042	815.641	65.414	86.072	10.330	2767.170	-201.450	0.759	16.382
043	538.054	75.498	-178.529	19.251	421.165	-921.177	1.330	27.978
044	439.828	12.726	59.162	7.368	1754.456	-908.253	1.146	12.551
045	476.003	26.291	153.040	10.243	-1270.568	637.079	0.900	18.516
046	876.922	72.158	34.263	-17.259	-1691.740	2216.451	1.313	17.458
047	728.967	62.679	-90.257	2.177	-1622.488	-2958.523	1.031	12.191
048	797.139	63.974	145.992	18.983	-2280.819	113.819	0.706	19.242
049	709.618	76.313	160.000	18.664	-2595.935	-374.791	1.358	15.139
050	306.972	33.538	2.456	-7.027	1108.133	-341.439	0.908	14.828
051	437.249	44.509	167.929	7.565	1307.288	354.175	0.732	17.892
052	900.579	31.448	-15.012	-11.670	1543.639	280.152	1.127	29.857
053	760.798	8.738	-176.633	3.892	955.056	480.039	0.986	35.441
054	708.805	42.045	-86.055	-17.953	1391.701	-2014.252	1.428	16.146
055	657.933	12.437	0.848	2.709	-1870.369	-1054.821	1.424	17.035
056	642.343	11.383	-42.830	-4.137	460.450	-2883.591	1.273	12.406
057	945.757	8.550	83.110	18.821	653.313	1317.994	1.162	32.738
058	567.218	3.842	-41.044	-5.531	-1274.496	1900.276	0.942	13.923
059	845.306	63.214	-77.701	-17.268	-2670.416	825.136	1.060	16.827
060	924.425	33.386	-123.847	1.600	2622.548	965.730	1.039	18.303
061	348.708	23.316	-13.724	-6.121	-1091.006	-240.513	1.161	17.335
062	222.202	52.676	-122.148	12.107	-548.521	-1035.649	0.889	10.736
063	797.082	13.916	-136.919	-13.038	764.455	2051.342	1.297	20.007
064	887.441	41.870	-8.104	15.595	-2609.543	56.737	1.369	18.778
065	786.854	18.400	-171.170	-14.440	1617.045	2818.591	1.197	13.612
066	451.995	12.746	-124.363	-14.522	1258.819	-210.813	1.133	19.501
067	760.703	14.399	108.713	0.558	290.630	-1752.931	0.791	23.177
068	621.192	45.682	-27.909	8.846	-2561.170	569.172	1.328	13.320
069	559.041	52.206	-70.357	4.298	-1326.586	1797.374	1.390	14.050
070	963.270	35.547	-15.436	3.993	2055.731	-2812.797	1.337	15.455
071	954.876	75.834	-16.832	12.433	2573.275	1036.304	0.850	18.994
072	524.557	35.106	63.956	-1.397	2719.585	-871.812	0.998	10.408
073	916.688	43.635	89.244	-15.005	-280.591	-2551.503	0.971	19.653
074	762.921	73.516	57.306	7.604	2122.342	-192.590	1.231	19.697
075	844.883	65.981	-110.824	-18.973	-2659.110	-2142.387	1.067	13.897
076	700.676	2.361	-9.905	7.138	-2311.266	-1583.616	0.837	14.041
077	338.205	25.896	107.797	-8.015	1653.754	316.625	0.931	11.357
078	784.518	61.890	143.503	-14.473	1764.461	-1863.520	1.144	16.998
079	898.025	29.366	-94.440	-12.507	273.896	-1469.301	1.211	30.999
080	321.482	34.910	-174.460	-10.839	-1417.922	68.311	1.179	12.760
081	476.888	59.825	-30.924	-17.770	-659.831	-153.096	0.872	35.147
082	442.915	65.743	23.513	-17.825	-1439.915	534.621	1.360	16.086
083	569.908	56.583	-35.367	-19.425	-2552.157	546.409	1.021	12.318
084	941.296	7.591	-44.600	1.840	-2329.930	2426.779	1.057	15.632
085	924.329	50.444	-173.906	-7.341	-2328.776	776.779	1.207	20.632
086	739.196	38.195	-69.615	0.653	1242.169	1881.666	0.749	18.152
087	459.277	51.228	135.951	-5.054	1600.425	-1991.480	1.217	10.191
088	701.964	57.113	-69.310	-9.453	2496.021	690.206	1.116	15.166

Table A-1: Scenario Parameters (continued)

	Platform properties				Position of the missile		Other	
Scenario Number	Altitude (m)	Velocity (m/s)	Heading Angle (degrees)	Climb Angle (degrees)	x-axis (m)	y-axis (m)	Atmospheric Attenuation (1/km)	Firing Angle (degrees)
089	702.158	15.362	99.153	14.580	-998.510	-2187.516	0.775	16.278
090	454.891	20.190	-107.373	-17.240	311.464	-577.130	1.312	34.746
091	589.747	30.783	-157.016	-11.452	263.240	-536.132	1.300	44.637
092	589.048	54.361	73.076	-1.565	-814.366	-1318.409	0.979	20.813
093	555.700	13.257	-36.248	16.823	68.006	2484.854	0.761	12.601
094	313.971	68.992	-80.179	1.268	133.328	405.706	0.703	36.324
095	530.733	33.148	173.244	-17.690	-620.743	1747.923	0.966	15.967
096	447.650	72.144	-145.661	-7.239	2321.770	944.590	1.175	10.126
097	579.159	11.300	161.431	15.305	-375.369	2009.741	1.248	15.816
098	554.192	63.188	148.040	1.330	1824.459	375.962	0.739	16.568
099	717.202	10.254	-149.887	6.369	-2835.607	2911.079	0.849	10.009
100	499.068	56.539	160.173	-4.709	1157.451	612.428	1.131	20.863
101	673.467	30.094	125.532	-10.971	1781.750	2981.286	1.320	10.974
102	768.305	53.171	-30.489	-0.069	2694.738	2719.090	0.925	11.348
103	507.744	3.207	29.690	2.589	-868.904	2281.205	1.286	11.750
104	699.212	23.659	-152.265	-8.252	-1591.561	-924.616	1.200	20.800
105	825.815	21.165	-66.338	-12.672	-315.160	-1039.928	1.375	37.234
106	945.411	31.975	-43.167	3.714	-2588.960	-1768.574	0.924	16.780
107	265.352	37.127	-168.061	-2.601	347.190	832.693	0.848	16.390
108	767.920	13.546	33.431	4.323	1634.158	-2662.358	0.727	13.811
109	507.427	31.970	-62.501	2.216	-1227.479	-803.280	1.384	19.081
110	704.179	53.155	176.170	17.774	-897.913	-1842.012	0.979	18.964
111	430.954	44.069	150.118	-16.398	-1453.822	-437.712	1.436	15.846
112	919.617	17.458	167.197	-2.641	1708.935	151.403	1.162	28.192
113	722.468	3.222	1.688	15.778	-685.629	-1247.689	1.415	26.907
114	360.759	30.424	33.935	-9.265	734.656	1827.404	0.887	10.380
115	783.392	51.887	-9.074	17.316	-2421.338	594.642	0.783	17.443
116	633.671	22.565	-91.335	-8.547	2778.806	-1615.828	1.305	11.152
117	632.301	65.429	74.613	-18.272	-2124.324	-1599.986	1.423	13.374
118	336.226	18.805	-80.385	18.064	-919.974	-1216.068	0.897	12.435
119	441.790	60.585	-50.212	-15.004	703.095	-866.883	1.024	21.594
120	729.486	16.002	164.675	6.604	247.799	2213.813	1.191	18.132
121	400.876	25.238	-71.365	-18.318	167.408	-1463.866	1.491	15.220
122	958.010	73.542	-135.595	3.678	-842.087	1315.842	1.027	31.518
123	889.818	54.781	48.043	-14.347	-2524.187	2256.872	0.996	14.724
124	590.131	36.826	5.612	-9.120	-1610.521	2397.204	1.036	11.549
125	682.914	29.219	35.295	6.739	2367.385	-2475.987	1.427	11.274
126	385.804	59.187	139.259	14.392	582.365	928.513	1.132	19.392
127	546.546	23.181	47.214	-8.183	732.173	-2714.795	1.432	11.000
128	945.560	11.769	-29.798	-8.788	588.602	-2781.151	1.118	18.398
129	692.165	2.156	-63.537	-1.449	-2405.897	425.929	1.393	15.817
130	560.394	46.228	-152.206	-17.706	-1194.261	130.327	0.961	25.008
131	848.472	36.102	-89.511	18.218	-2144.098	75.380	0.888	21.578
132	718.656	49.174	-10.865	3.111	2467.879	-742.678	1.478	15.581
133	538.820	21.888	-19.845	5.101	207.848	-687.347	0.883	36.883
134	912.220	2.671	121.375	0.291	-2317.701	-57.326	0.779	21.478
135	816.787	25.104	-158.282	-18.237	1877.664	-527.560	1.495	22.723

Table A-1: Scenario Parameters (continued)

	Platform properties				Position of the missile		Other	
Scenario Number	Altitude (m)	Velocity (m/s)	Heading Angle (degrees)	Climb Angle (degrees)	x-axis (m)	y-axis (m)	Atmospheric Attenuation (1/km)	Firing Angle (degrees)
136	618.501	71.372	-33.680	4.176	-2431.179	-984.112	1.007	13.269
137	395.904	30.318	-82.226	-11.376	802.387	1798.293	0.816	11.368
138	917.005	59.657	13.026	18.799	389.299	-1693.861	0.867	27.817
139	889.599	25.064	-61.940	13.601	-44.803	-2688.585	1.386	18.306
140	898.148	22.865	56.152	-10.725	731.724	-2549.257	1.281	18.708
141	687.976	30.697	-168.056	14.303	621.259	2087.042	1.473	17.533
142	941.995	75.537	144.368	4.164	2557.259	-2298.322	1.256	15.321
143	824.098	42.957	96.328	5.558	2358.889	-2635.858	0.943	13.115
144	533.065	59.183	140.678	-18.966	-2174.622	-455.475	0.841	13.492
145	619.550	60.359	-118.212	6.909	711.916	-2958.923	1.312	11.507
146	993.388	10.255	-49.911	-12.357	1286.386	-1932.798	1.292	23.164
147	668.724	15.259	1.232	-17.963	-2663.426	-988.816	1.116	13.245
148	913.582	53.872	66.326	7.830	1798.989	963.631	1.205	24.116
149	465.389	74.734	-90.761	0.448	1441.907	-1474.989	1.116	12.714
150	630.524	72.811	-53.447	17.413	2566.317	526.202	1.376	13.533
151	662.737	25.248	176.858	19.356	2787.287	997.591	1.063	12.618
152	502.716	27.231	-156.141	10.453	-584.027	1045.738	1.075	22.768
153	241.166	24.600	166.629	17.259	-731.840	706.021	1.141	13.342
154	864.449	76.616	-151.969	15.521	374.684	-1833.869	1.150	24.789
155	764.610	47.741	30.894	18.740	490.461	-2401.198	0.877	17.327
156	839.981	31.983	91.333	-8.192	840.035	2310.703	1.012	18.862
157	584.285	9.039	-131.612	-17.440	-2524.216	761.725	0.868	12.495
158	710.872	68.525	94.360	19.046	1689.223	2613.499	1.029	12.868
159	402.896	57.540	69.278	10.933	1483.303	-1089.698	1.291	12.347
160	818.453	45.825	162.478	-13.137	2443.411	1513.317	1.108	15.895
161	701.928	37.026	-132.661	1.975	2817.056	-347.157	0.929	13.891
162	307.474	79.888	4.840	-4.486	-1500.883	-811.485	0.987	10.216
163	941.041	39.638	39.089	-19.808	526.373	1664.775	1.019	28.323
164	622.652	66.062	165.551	-7.451	1782.510	-1288.945	1.226	15.805
165	745.005	57.163	-14.398	16.771	2933.661	2595.607	1.313	10.769
166	923.938	30.916	36.880	2.413	2075.039	-1291.170	1.069	20.709
167	681.807	52.517	-68.044	-6.734	-1870.816	-2395.900	1.231	12.642
168	483.861	42.861	175.718	-18.873	1257.192	2430.942	0.929	10.026
169	891.639	1.679	-151.498	-4.931	-2104.838	-2795.419	1.205	14.295
170	487.155	29.053	-83.255	-6.508	-2478.236	-290.642	0.884	11.047
171	223.228	50.968	-157.714	-13.232	1108.078	327.404	1.064	10.935
172	430.468	30.129	-126.555	-17.019	-235.543	-777.634	0.705	27.914
173	774.568	12.754	-79.200	5.798	-1276.775	-1068.415	0.993	24.950
174	510.131	71.659	138.821	-4.255	1052.746	-1486.715	0.824	15.644
175	699.746	16.556	-139.743	2.634	-1354.000	-2573.008	1.460	13.532
176	979.864	66.701	-57.244	4.650	-1177.463	-2464.687	0.816	19.734
177	860.233	61.114	160.255	-6.658	-661.570	-2097.523	1.117	21.362
178	642.943	44.024	-121.569	-15.316	-608.697	1988.298	0.967	17.182
179	600.629	10.105	130.537	10.666	386.121	-662.584	0.848	38.068
180	909.523	72.404	-0.577	1.169	2458.204	471.591	1.092	19.970
181	729.116	37.573	-100.323	4.104	-1894.446	-1814.942	1.321	15.531
182	868.406	29.545	57.759	15.829	-1355.406	2987.391	0.734	14.827

Table A-1: Scenario Parameters (continued)

	Platform properties				Position of the missile		Other	
Scenario Number	Altitude (m)	Velocity (m/s)	Heading Angle (degrees)	Climb Angle (degrees)	x-axis (m)	y-axis (m)	Atmospheric Attenuation (1/km)	Firing Angle (degrees)
183	833.075	52.529	15.635	-4.533	1933.393	572.007	1.367	22.450
184	990.140	63.060	126.346	-0.575	2242.536	-996.918	1.325	21.972
185	595.486	2.497	117.272	-9.430	1064.996	1763.186	0.864	16.124
186	720.607	18.987	-8.090	17.457	-1553.379	-1745.317	1.245	17.141
187	820.654	26.513	36.901	-12.639	-2475.247	-1146.508	0.918	16.743
188	927.350	74.951	-167.587	3.746	-2737.095	-450.583	0.885	18.485
189	872.272	50.002	-87.637	16.188	1603.804	375.993	1.117	27.902
190	494.464	27.018	42.550	-16.524	-747.015	-1875.714	1.418	13.761
191	778.599	53.974	148.205	12.838	292.348	-2906.750	1.348	14.923
192	611.807	76.199	122.816	17.400	2840.479	-764.842	0.776	11.749
193	826.598	16.978	164.376	17.209	-805.623	-930.526	0.957	33.885
194	773.283	64.027	33.534	-16.113	357.055	-2741.460	1.254	15.627
195	727.718	47.957	-97.736	8.549	1347.014	-934.692	1.306	23.934
196	874.893	61.624	3.453	8.279	1747.043	-1015.986	0.726	23.408
197	661.364	23.033	-83.462	15.607	-803.393	-2099.908	1.431	16.392
198	651.567	50.684	-0.680	-17.615	90.979	-2680.830	0.736	13.653
199	675.738	55.282	52.707	4.938	2723.150	-1938.336	1.375	11.429
200	873.923	39.757	93.941	-12.805	1149.732	2486.549	1.080	17.693

APPENDIX B

MEAN AND STANDART DEVIATION OF THE DATA SETS

Table B-1: Mean and Standard Deviation of T-Scenarios (Features Extracted By Using Fourier Transform)

T-SCENARIOS (Features Extracted By Using Fourier Transform)					
AZIMUTH	**a_0 (1)[5]**	**a_1 (2)**	**b_1 (3)**	**a_2 (4)**	**b_2 (5)**
Mean	-8.9838	0	0.0025	0	0.0015
Std	102.7505	0.1693	0.2932	0.1042	0.1572
ELEVATION	**a_0 (6)**	**a_1 (7)**	**b_1 (8)**	**a_2 (9)**	**b_2 (10)**
Mean	-16.5419	0.0151	-0.0749	0.0129	-0.037
Std	10.0352	0.1789	0.2889	0.1049	0.1541
AREA	**a_0 (11)**	**a_1 (12)**	**b_1 (13)**	**a_2 (14)**	**b_2 (15)**
Mean	8.7594	1.938	-2.4455	1.2848	-1.4485
Std	12.3473	9.3414	5.4014	7.876	3.694
DIGITAL LEVEL	**a_0 (16)**	**a_1 (17)**	**b_1 (18)**	**a_2 (19)**	**b_2 (20)**
Mean	63.7848	0.7071	-1.9865	0.4307	-1.0103
Std	10.6374	2.6565	4.0287	1.9025	2.3486

[5] (1) Indicates the feature number

Table B-2: Mean and Standard Deviation of F-Scenarios (Features Extracted By Using Fourier Transform)

F-SCENARIOS (Features Extracted By Using Fourier Transform)					
AZIMUTH	**a_0 (1)[6]**	**a_1 (2)**	**b_1 (3)**	**a_2 (4)**	**b_2 (5)**
Mean	-10.1180	-0.0051	0.0329	-0.005	0.0157
Std	104.6367	0.2633	0.5985	0.1679	0.3027
ELEVATION	**a_0 (6)**	**a_1 (7)**	**b_1 (8)**	**a_2 (9)**	**b_2 (10)**
Mean	-17.8549	0.0001	-0.0052	0.0006	-0.0026
Std	10.1255	0.1299	0.2353	0.0819	0.123
AREA	**a_0 (11)**	**a_1 (12)**	**b_1 (13)**	**a_2 (14)**	**b_2 (15)**
Mean	28.4078	-0.0401	0.3639	-0.0531	0.1776
Std	32.4068	2.5494	4.0462	1.5033	2.1041
DIGITAL LEVEL	**a_0 (16)**	**a_1 (17)**	**b_1 (18)**	**a_2 (19)**	**b_2 (20)**
Mean	78.2607	-0.0032	0.0127	-0.0053	0.0079
Std	13.1386	1.876	2.4885	1.402	1.6022

[6] (1) Indicates the feature number

Table B-3: Mean and Standard Deviation of T-Scenarios (Features Extracted By Using Linear Fitting)

T-SCENARIOS (Features Extracted By Using Linear Fitting)							
AZIMUTH	**a (1)[7]**	**b (2)**	**se(a) (3)**	**se(b) (4)**	**f(t) (5)**	**rmse (6)**	**r^2 (7)**
Mean	0.000	-8.940	0.014	0.960	-8.988	0.176	0.519
Std	0.045	102.760	0.012	1.115	102.790	0.148	0.353
ELEVATION	**a (8)**	**b (9)**	**se(a) (10)**	**se(b) (11)**	**f(t) (12)**	**rmse (13)**	**r^2 (14)**
Mean	0.012	-17.113	0.013	0.817	-16.428	0.155	0.519
Std	0.046	10.651	0.013	1.062	9.999	0.156	0.357
AREA	**a (15)**	**b (16)**	**se(a) (17)**	**se(b) (18)**	**f(t) (19)**	**rmse (20)**	**r^2 (21)**
Mean	0.511	-41.178	0.184	16.230	13.610	2.264	0.590
Std	1.652	152.530	1.110	99.460	26.955	13.622	0.337
DIGITAL LEVEL	**a (22)**	**b (23)**	**se(a) (24)**	**se(b) (25)**	**f(t) (26)**	**rmse (27)**	**r^2 (28)**
Mean	0.330	33.196	0.410	24.689	66.924	5.038	0.270
Std	0.598	48.806	0.169	16.446	14.716	2.079	0.276

[7] (1) Indicates the feature number

Table B-4: Mean and Standard Deviation of F-Scenarios (Features Extracted By Using Linear Fitting)

F-SCENARIOS (Features Extracted By Using Linear Fitting)							
AZIMUTH	**a (1)[8]**	**b (2)**	**se(a) (3)**	**se(b) (4)**	**f(t) (5)**	**rmse (6)**	**r^2 (7)**
Mean	-0.005	-9.885	0.025	1.304	-10.167	0.306	0.520
Std	0.090	101.970	0.016	1.250	105.140	0.196	0.294
ELEVATION	**a (8)**	**b (9)**	**se(a) (10)**	**se(b) (11)**	**f(t) (12)**	**rmse (13)**	**r^2 (14)**
Mean	0.001	-17.893	0.012	0.606	-17.847	0.146	0.452
Std	0.034	10.236	0.011	0.714	10.144	0.136	0.313
AREA	**a (15)**	**b (16)**	**se(a) (17)**	**se(b) (18)**	**f(t) (19)**	**rmse (20)**	**r^2 (21)**
Mean	-0.057	33.201	0.174	8.236	27.864	2.137	0.193
Std	0.564	46.532	0.227	10.486	32.381	2.780	0.248
DIGITAL LEVEL	**a (22)**	**b (23)**	**se(a) (24)**	**se(b) (25)**	**f(t) (26)**	**rmse (27)**	**r^2 (28)**
Mean	-0.002	79.118	0.304	15.240	78.238	3.736	0.112
Std	0.333	21.502	0.143	10.385	13.776	1.752	0.156

[8] (1) Indicates the feature number

APPENDIX C

HISTOGRAM PLOTS OF THE DATA SETS

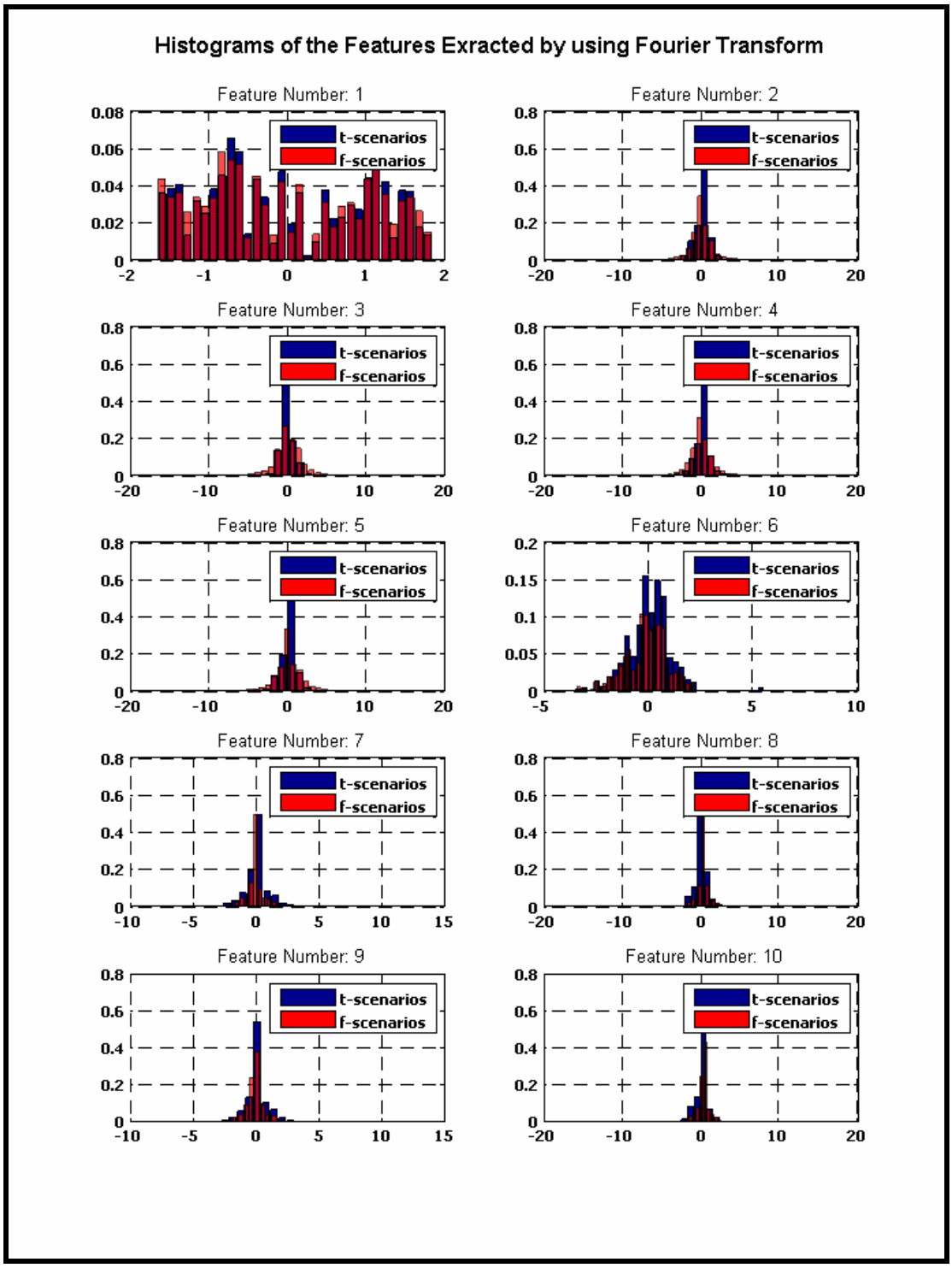

Figure C-1: Histograms of Features Extracted by Using Fourier Transform

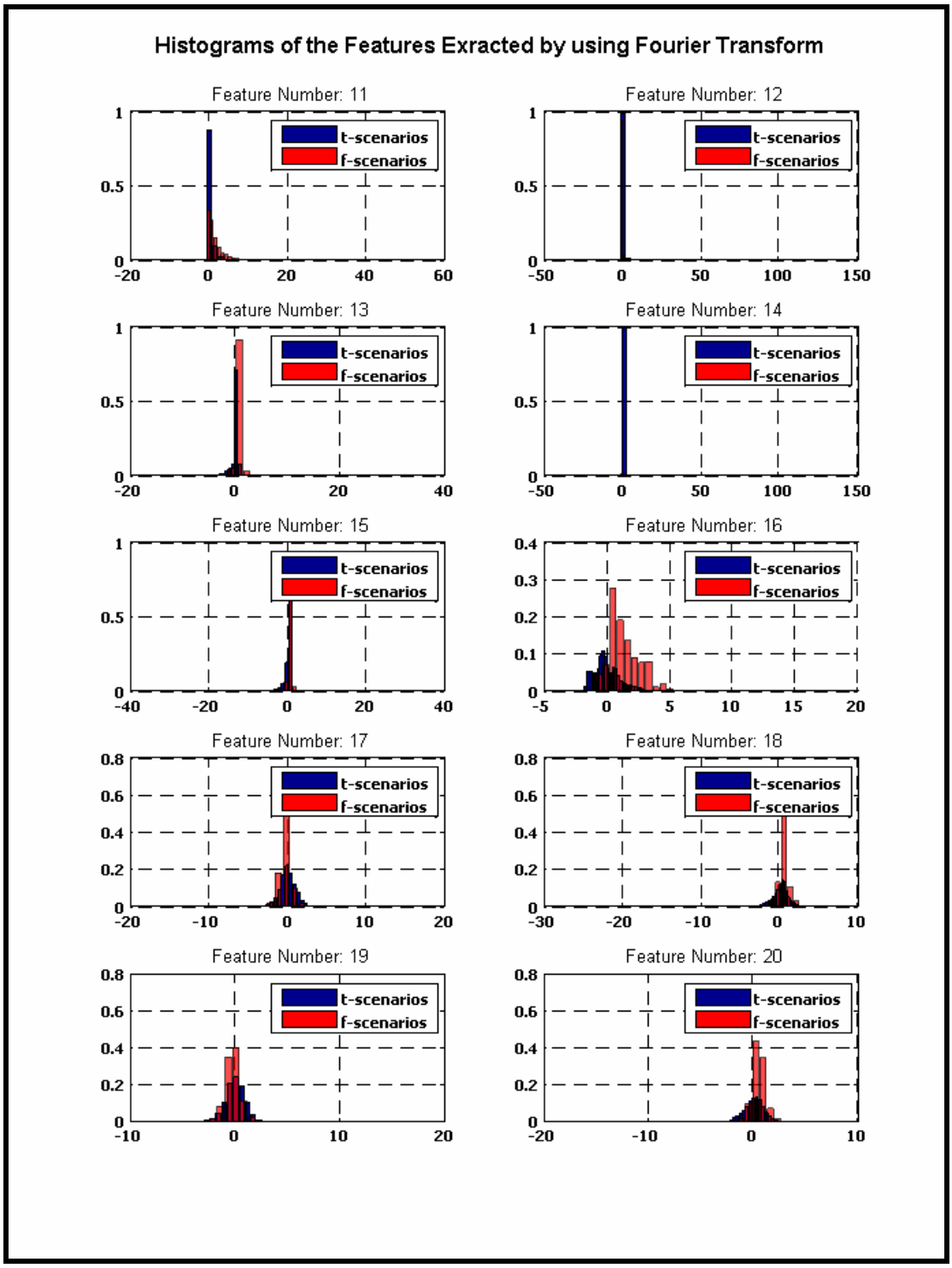

Figure C-1: Histograms of Features Extracted by Using Fourier Transform (continued)

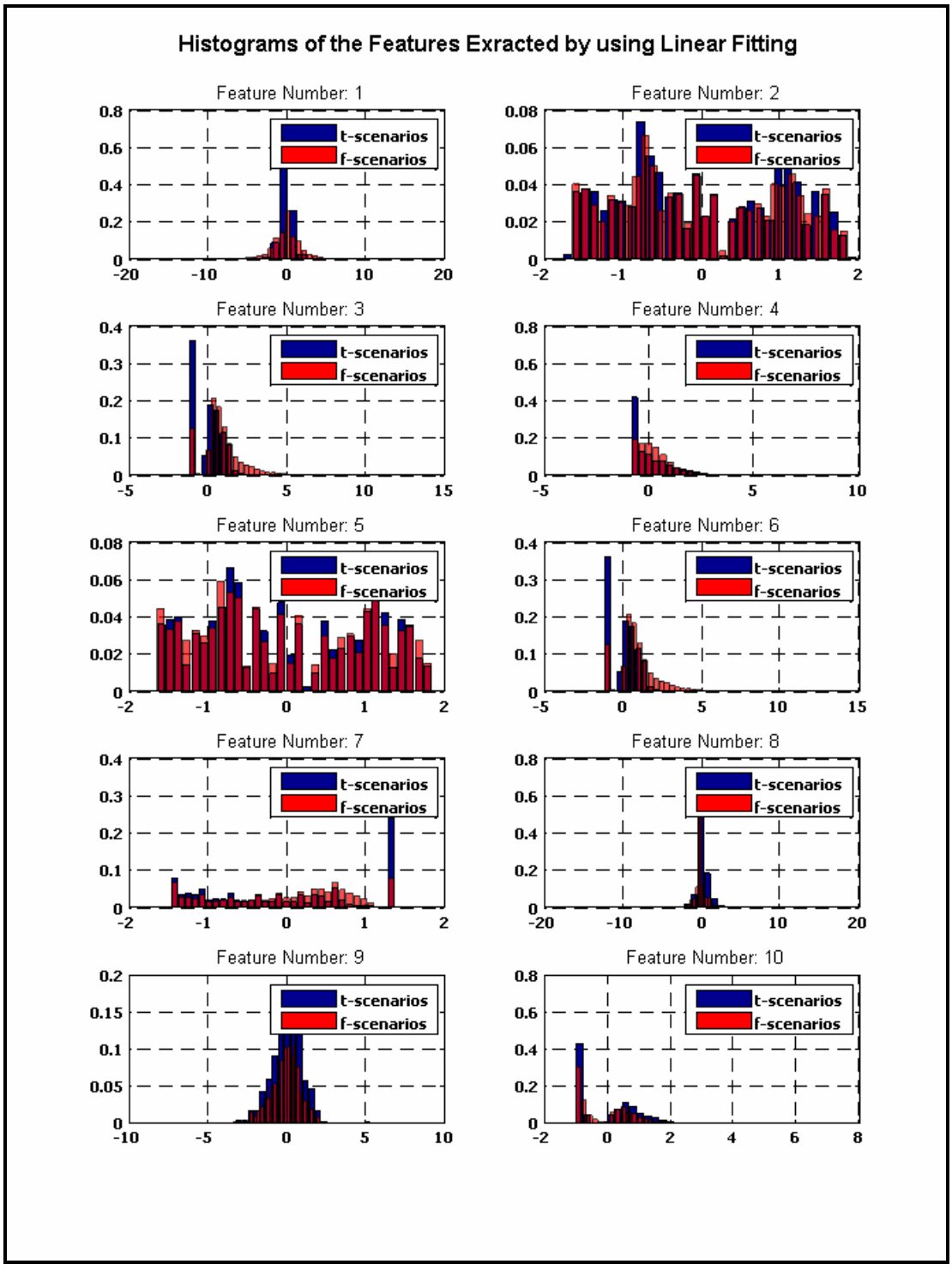

Figure C-2: Histograms of Features Extracted by Using Linear Fit

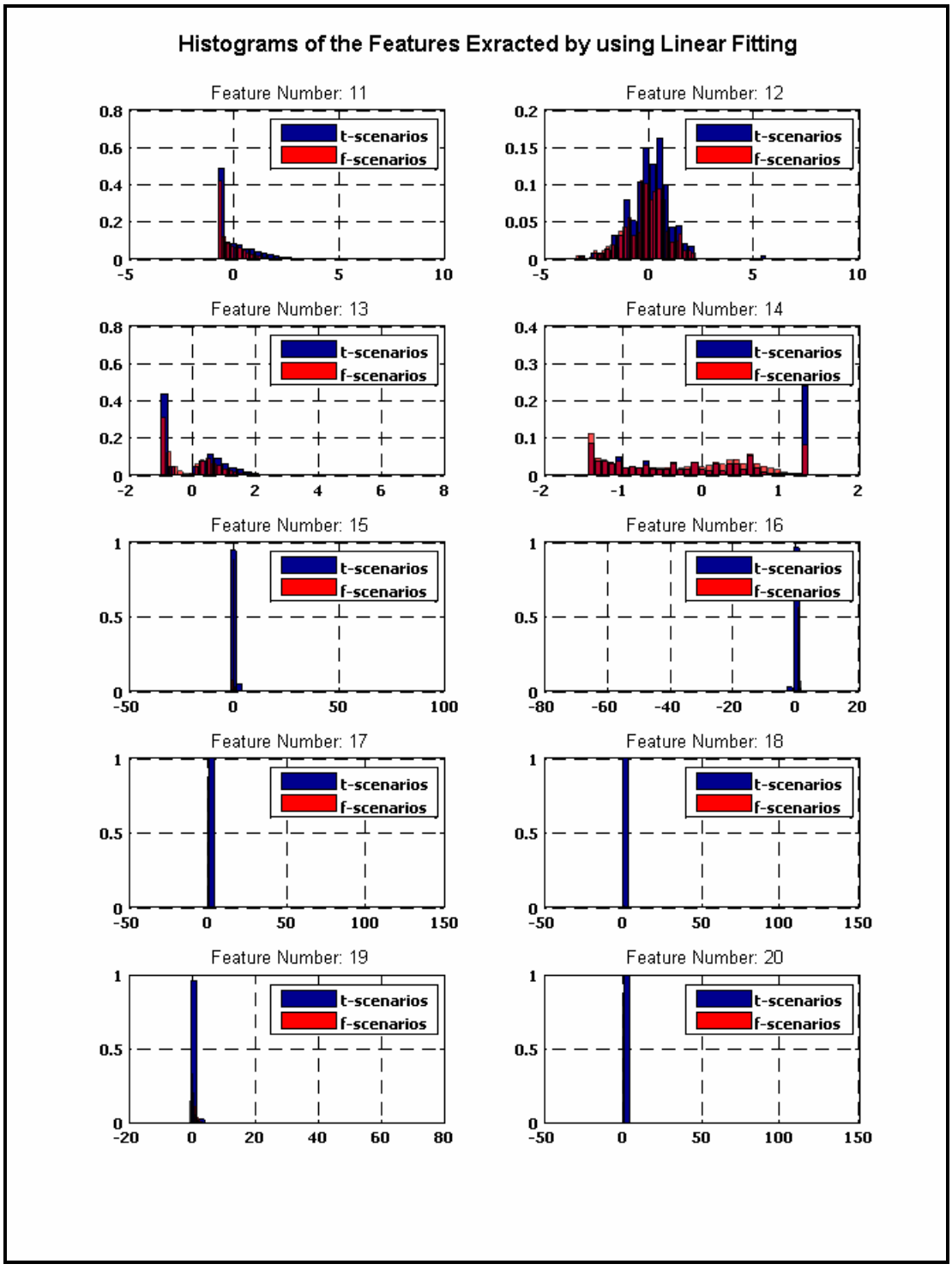

Figure C-2: Histograms of Features Extracted by Using Linear Fit (continued)

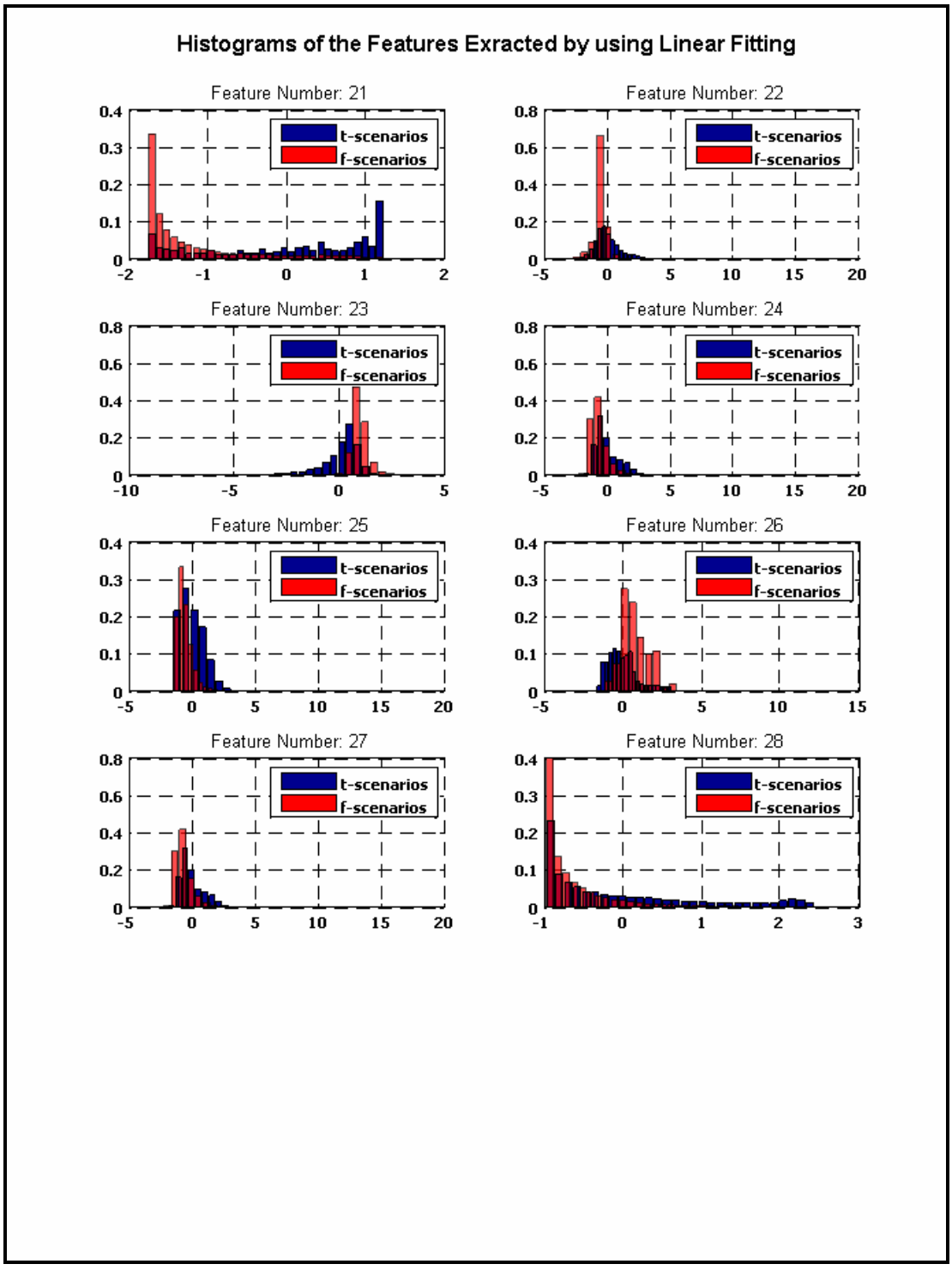

Figure C-2: Histograms of Features Extracted by Using Linear Fit (continued)

APPENDIX D

CORRELATION MATRICES

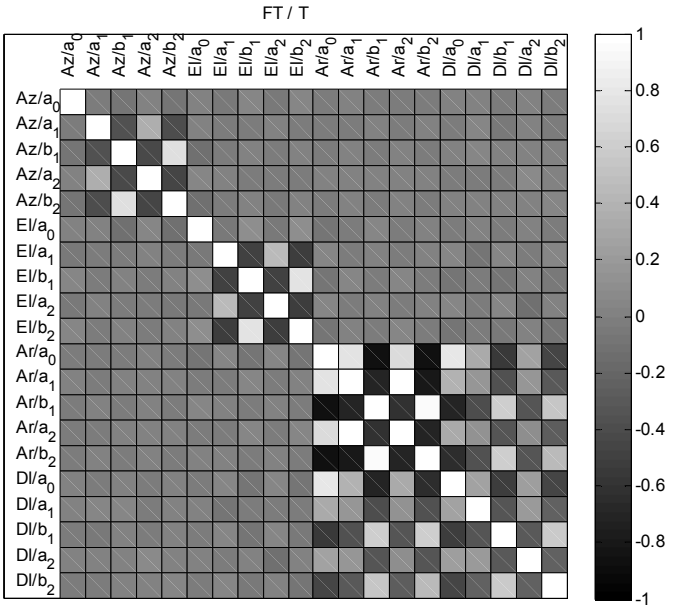

Figure D- 1: Correlation Matrix for T-Scenarios of FT-Set

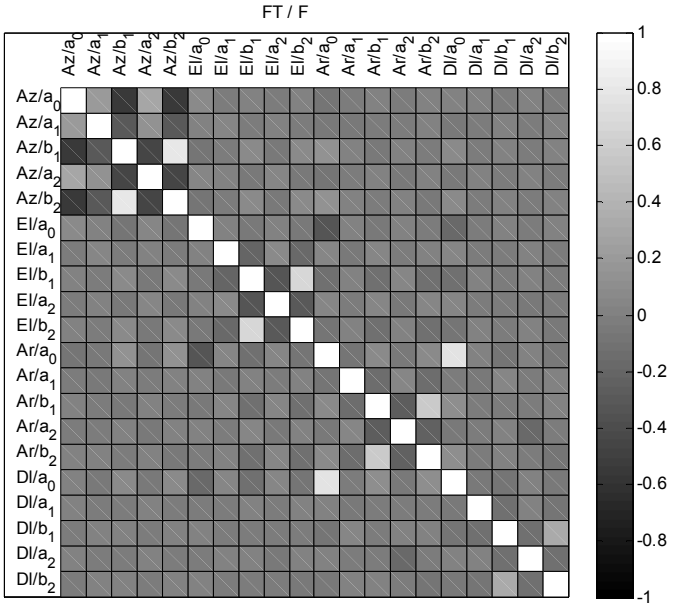

Figure D- 2: Correlation Matrix for F-Scenarios of FT-Set

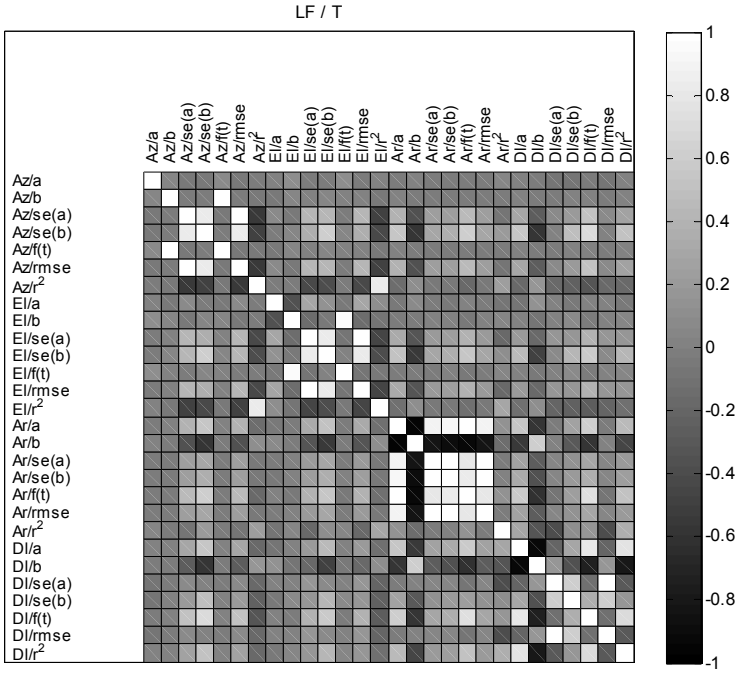

Figure D- 3: Correlation Matrix for T-Scenarios of LF-Set

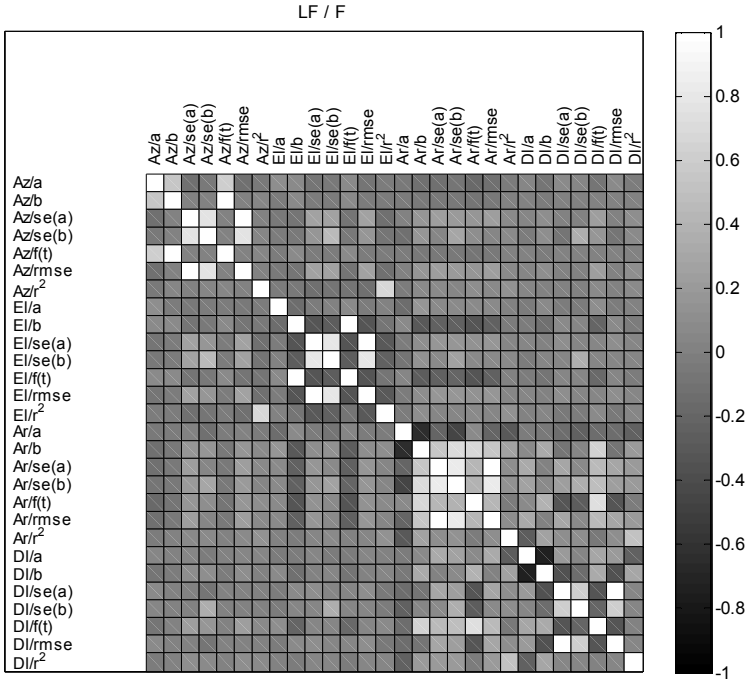

Figure D- 4: Correlation Matrix for F-Scenarios of LF-Set