

**CLASSIFICATION OF TRANSMITTER TRANSIENTS
USING FRACTAL MEASURES AND
PROBABILISTIC NEURAL NETWORKS**

By

Donald B. Shaw

A Thesis

**Submitted to the Faculty of Graduate Studies
in partial fulfilment of the requirements
for the Degree of**

MASTER OF SCIENCE

**Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada**

Thesis Advisor: W. Kinsner, Ph.D., P.Eng.

© D. Shaw; August 1997

(xiv + 126 + A-2 + B-150 + C-24 + D-17 + E-42) = 375 pp.



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-23494-0

**THE UNIVERSITY OF MANITOBA
FACULTY OF GRADUATE STUDIES
COPYRIGHT PERMISSION**

**CLASSIFICATION OF TRANSMITTER TRANSIENTS USING
FRACTAL MEASURES AND PROBABILISTIC NEURAL NETWORKS**

by

DONALD B. SHAW

**A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba
in partial fulfillment of the requirements of the degree of**

MASTER OF SCIENCE

DONALD B. SHAW © 1997

**Permission has been granted to the LIBRARY OF THE UNIVERSITY OF MANITOBA
to lend or sell copies of this thesis, to the NATIONAL LIBRARY OF CANADA to microfilm this
thesis and to lend or sell copies of the film, and to UNIVERSITY MICROFILMS to publish an
abstract of this thesis.**

**This reproduction or copy of this thesis has been made available by authority of the copyright
owner solely for the purpose of private study and research, and may only be reproduced and
copied as permitted by copyright laws or with express written authorization from the copyright
owner.**

ABSTRACT

This thesis presents a method of identifying the source of radio transmissions by analysis of the transients exhibited at the start of the transmitted signal. It is motivated by the intriguing possibility of identifying radio transmitters used in violation of federal and international regulations. As well, such a system could be directly used for analysis or classification of other nonstationary signals such as speech or power system transients.

The system developed in this thesis uses a multifractal analysis for precise segmentation of a transmitter transient from the ambient channel noise. This is critical to ensure that the portion of the signal being analysed does not contain meaningless noise and, at the same time, represents the entire transition from noise to signal. Then, using a similar multifractal method, significant features of the transient are extracted and stored for neural network analysis. This modelling process is equally important as it provides a means to reduce the size of the data for efficient neural network processing, while providing significant emphasis on the most important features. Finally, the transient model is classified using a Probabilistic Neural Network (PNN).

Experimental results indicate that this classification system is fast and accurate. The three stages of segmentation, feature extraction, and classification are performed in about a half second for a 16 kB transient. In the most successful experiment, the system was trained with 160 out of the 415 available transients, representing eight different classes of transmitters. Testing the system with the remaining 255 transients yielded results in which 96.9% of them were classified correctly.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Dr. W. Kinsner, for his unending support and guidance throughout the writing of this thesis. Dr. Kinsner encouraged me to undertake this course of study and provided me with motivation to see it through to completion. He has shown me that research can be both rewarding and exciting.

I would like to thank my supervisors in the Canadian Armed Forces, past and present, for their support and understanding of the time commitment involved for a project of this magnitude. Specifically Major Al Deutscher, Captain Kathy Boulet, and Captain Rob McIntosh are acknowledged. I would also like to thank Mr. Tom Silletta, Head Coach of the Canadian Military Pentathlon Team, for his encouragement throughout the completion of this thesis.

I would like to thank researchers from the Communications Research Centre (CRC) for providing the set of radio transmitter transients used for testing the system developed in this thesis.

The friendship and discussions provided by my colleagues in the Delta Research Group are also acknowledged. I would especially like to thank Jason Toonstra, who was also studying transmitter transients, for his discussions about these “nasty little signals”.

I would like to thank my family for their constant support and encouragement during this research and in all of my other endeavours. Finally, I would like to thank Tracie for her patience and love.

TABLE OF CONTENTS

COPYRIGHT PAGE	i
THESIS APPROVAL FORM	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS.	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
LIST OF ABBREVIATIONS.	xii
LIST OF SYMBOLS	xiii
CHAPTER I INTRODUCTION.	1
Background and Motivation	1
The Preprocessing Stage	2
The Feature Extraction Stage	3
The Classification Stage	3
Thesis Statement and Objectives	4
Thesis Organization	4
CHAPTER II FRACTAL DIMENSION AND MULTIFRACTALITY	6
Fractal Dimensions	6
Morphological-Based Dimensions	7
Entropy-Based Single and Multifractal Dimensions	9
Spectrum-Based Dimensions	12
Variance-Based Dimensions	13
Fractal Dimensions for Transient Analysis	14
Calculating the Variance Dimension	15
Fractal Measures vs. Time	18
Local Fractal Dimensions and Multifractality	18

Variance Fractal Dimension Trajectory 19

Summary of Chapter 2 20

**CHAPTER III MULTIFRACTAL MODELLING OF TRANSIENTS FOR SEGMENTATION AND
FEATURE EXTRACTION 21**

The Preprocessing Stage 21

 Motivation 21

 Fractal Segmentation 22

 Segmentation Parameters 23

 Alignment Issues 25

The Feature Extraction Stage 26

 Motivation 26

 Multifractal Feature Extraction Parameters 27

Summary of Chapter 3 29

CHAPTER IV THE PROBABILISTIC NEURAL NETWORK 30

Neural Network and PNN Foundation 31

 Brief History of the PNN 31

 Classification with Complex Class Distributions 31

 Advantages of the PNN Over the MLFN 33

 Disadvantages of the PNN 35

The Basic PNN 36

 Architecture 36

 Parzen's Method of Density Estimation 37

 PNN Processing 40

Training the PNN 43

 An Error Function for the PNN 43

 Optimizing Sigma 45

Improving the Basic PNN 48

 The Multiple-Sigma PDF Estimator 49

 A Continuous Error Function for the PNN 50

	Derivatives of the Continuous Error Function	54
	Training the Multiple-Sigma PNN Using Conjugate Gradients	56
	Other Extensions to the PNN Classifier	58
	Bayes Classification and Confidence Levels	59
	Bayes' Strategy for Classification	59
	Implementing Bayes' Method in the PNN	61
	Bayesian Confidence Measures for the PNN	62
	Summary of Chapter 4	63
CHAPTER V	SOFTWARE IMPLEMENTATION AND VERIFICATION	64
	Using the TAC-MM Software Package	65
	The User Display Area	65
	The File Menu	68
	The Edit Menu	70
	The View Menu	73
	The Neural Net Menu	75
	The Help Menu	80
	Verification of the System Software Modules	80
	Multifractal Segmentation and Feature Extraction	81
	The PNN Classifier	88
	Summary of Chapter 5	94
CHAPTER VI	CLASSIFYING TRANSMITTER TRANSIENTS	95
	The Thesis Test Set	95
	Transient Capturing System	95
	The Transient File Structure	96
	Composition of the Test Set	98
	Testing and Results	99
	Training Set # 1 (First 20 Transients)	99
	Training Set # 2 (Random Selection of 10)	104
	Training Set # 3 (Random Selection of 30)	106

Rejection of Unknown Transients 108
Multimodal Segmentation 110
Transformation of Fractal Trajectory Model 113
Confidence Measures and the PNN 117
Summary of Chapter 6 119

CHAPTER VII CONCLUSIONS AND RECOMMENDATIONS. **121**
Conclusions 121
Contributions 122
Recommendations. 123

REFERENCES **124**

APPENDIX A TAC-MM FILE STRUCTURES **A-1**
APPENDIX B TAC-MM SOURCE CODE **B-1**
APPENDIX C DATA AND SOFTWARE FOR FRACTAL DIMENSION VERIFICATION... **C-1**
APPENDIX D TRANSIENTS IN THE THESIS TEST SET **D-1**
APPENDIX E EXPERIMENTAL CLASSIFICATION RESULTS. **E-1**

LIST OF FIGURES

1.1	Ambient channel noise followed by radio transmission	3
2.1.	Covering set made up of symmetrically aligned vels (After [Kins94a]).	8
2.2.	A graph of $\log(P(f))$ vs $\log(f)$ for finding the power spectrum exponent, β	13
2.3.	Example of dyadic sequence with $NT = 256$ and $K = 5$	16
2.4.	Pseudocode for variance fractal dimension calculation.	17
2.5.	Window for calculating the local fractal dimension of a signal.	19
3.1.	Separating the transient from the ambient noise in a raw signal.	22
3.2.	Alignment of the fractal trajectory.	26
3.3.	An example of multifractal feature extraction.	28
4.1.	Classes X and O plotted on a plane with one unknown (After [Mast93]).	32
4.2.	Architecture of the PNN.	36
4.3.	Parzen's approximated PDF.	38
4.4.	Bracketing triplet and test point in golden section minimization.	47
4.5.	Fractal model of transient.	49
5.1.	TAC-MM user display area; default view options.	65
5.2.	TAC-MM user display area; secondary view options.	67
5.3.	File: New dialog box.	68
5.4.	Edit: Fractal Parameters dialog box.	72
5.5.	PNN: Classify results dialog box.	76
5.6.	PNN: Classify - Add Transient to Database dialog box.	76
5.7.	PNN: Mode Parameters - Select Classification Mode dialog box.	78
5.8.	Parameters for verification of TAC-MM variance dimension calculations.	82
5.9.	Comparison of expected and calculated fractal dimensions.	84
5.10.	FBm signal generated using direct spectral filtering with $\beta = 1.0$	85
5.11.	Parameters to show contrast between dyadic and linear time increments.	87
5.12.	Parameters for verification of PNN module in TAC-MM.	88
5.13.	Test signal generated to cause the basic PNN to fail.	91
5.14.	Parameters for verification of enhanced PNN structure in TAC-MM.	92

6.1.	Transient in unresolved circular buffer.	97
6.2.	Resolution of circular buffer containing noise and a transient.	98
6.3.	TAC-MM parameters for testing Training Set #1.	101
6.4.	Consistently misclassified Force 1 transient.	113
6.5.	Comparison of (a) multifractal model and (b) transformed model.	114
6.6.	TAC-MM parameters for testing eight element transformed model.	116

LIST OF TABLES

5.1.	Verification of fractal dimension trajectory calculations.	83
5.2.	Contrasting dyadic and linear time increments for D_G calculations.	87
6.1.	Transmitters used for testing the thesis.	99
6.2.	Confusion matrix with results from Validation Set # 1.	103
6.3.	Confusion matrix with results from Validation Set # 2.	105
6.4.	Confusion matrix with results from Validation Set # 3.	106
6.5.	Confusion matrix with results from test for rejection ability.	110
6.6.	Confusion matrix from Validation Set # 1 using multimodal segmentation. ..	111
6.7.	Confusion matrix from Validation Set # 2 using multimodal segmentation. ..	112
6.8.	Confusion matrix from Validation Set # 3 using multimodal segmentation. ..	112
6.9.	Classification results using transformed fractal model.	116
6.10.	Classification results using eight element model size.	116

LIST OF ABBREVIATIONS

1-NN	Single Nearest Neighbour
BPN	Backpropagation Network
CRC	Communications Research Centre
DIB	Device Independent Bitmap
EMD	Euclidean Minimum Distance
fBm	fractional Brownian motion
FFT	Fast Fourier Transform
GCNN	Gram-Charlier Neural Network
GUI	graphical user interface
kB	kilobytes
k-NN	k Nearest Neighbour
ksps	kilosamples per second
LSR	least squares regression
MB	megabyte
MFC	Microsoft Foundation Class
MLFN	multiple layer feedforward network
NIST	U.S. National Institute of Standards and Technology
NRML	Normal Parametric Classifier
PDF	probability density function
PNN	Probabilistic Neural Network
QMD	Quadratic Minimum Distance
RBF	Radial Basis Functions
SDI	single document interface
SNR	signal to noise ratio
TAC-MM	Transient Analyser and Classifier using Multifractal Modelling
WSNN	Weighted Several Nearest Neighbour

List of Symbols

a	lower bound for search range in univariate optimization
b	upper bound for search range in univariate optimization
β	power spectrum exponent
$B(t)$	a time-varying signal
c	index for different classes in a training set
C	number of different classes in training set
$d(X, X_r)$	Euclidean distance, scaled by sigma, between X and X_r
$\delta(X, X_r)$	Euclidean distance, scaled by multiple sigmas, between X and X_r
Δt_k	time increment at the k th pair in variance dimension calculation
D	fractal dimension
D_β	spectral dimension
D_c	correlation dimension
D_E	Euclidean dimension
D_{HB}	Hausdorff-Besicovitch dimension
D_{HM}	Hausdorff Mesh dimension or Box-Counting dimension
D_I	information dimension
D_q	Rényi dimension
D_σ	variance dimension
$D_\sigma(t)$	time-varying variance dimension (trajectory)
$e(X)$	continuous error function for PNN processing
E	embedding Euclidean dimension
$E_{Y X}(X)$	expected value of the vector Y given the vector X
f	frequency
$f_c(X)$	actual PDF for a given class, c
$f_{Xy}(X, y_c)$	joint PDF between the vector X and the vector element y_c
γ	bounded minimum in univariate optimization
$g_c(X)$	estimated PDF for a given class, c
$g_{Xy}(X, y_c)$	estimated joint PDF between the vector X and the vector element y_c
h_c	prior probabilities of encountering a member of class, c
H	Hurst exponent
H_v	the Shannon entropy of the fractal in information dimension calculation
H_q	generalized entropy function for Rényi dimension calculation
i	general purpose index variable
j	index for vels in fractal dimension calculation
k	index for ordered pairs in variance dimension calculation
K	number of ordered pairs in variance dimension calculation
l_c	loss associated with misclassifying a case that belongs to a class, c
m	consecutive search point multiplier in univariate optimization
$\mu_{D\sigma(t)}$	mean of variance dimension trajectory
n_c	number of training cases for a given class, c
n_j	frequency of vel intersection in correlation dimension calculation
n_k	number of samples between points in variance dimension calculation
N_F	frequency summation in information/correlation dimension calculation

N_k	number of comparisons at the k th time increment in D_G calculation
N_s	number of points in univariate global search range
N_T	number of samples in time interval, T
N_v	number of vels in fractal dimension calculation
p	number of data points in an input vector for PNN processing
p_j	probability of vel intersection in correlation dimension calculation
$P(f)$	power spectrum density
q	the moment order in generalized entropy function of Rényi dimension
$q_c(X)$	internal confidence measure of X in class c in PNN processing
r	index for training case in training set (index for pattern layer neuron)
R	number of training cases in training set (number of pattern layer neurons)
$s(X)$	sum of all summation neuron activation functions in PNN processing
σ	scaling parameter in Parzen PDF estimator
σ^2	variance
$\sigma_{D\sigma(t)}$	standard deviation of variance dimension trajectory
S	window spacing in local fractal dimension calculation
t	time
τ	threshold in transient triggering mechanism
T	size of time interval
$u_c(X)$	an activation function for summation neuron, c
v	size of vel in fractal dimension calculation
$V(f)$	Fourier transform of a signal
w	golden-section number for univariate optimization
W	fixed window width in local fractal dimension calculation
$W(x)$	weight function in Parzen PDF estimator
x_i	i th element in sample vector, X , for PNN processing
x	value on x-axis of log-log plot for fractal dimension calculation
X	sample vector for PNN processing
X_r	r th sample vector in PNN training set
y_i	i th element of PNN output vector, Y
y_{rc}	c th element of PNN output vector, Y_r (for training case r)
y	value on y-axis of log-log plot for fractal dimension calculation
Y	vector at output of PNN
Ψ	exponent for sequence of time increments in variance dimension calculation
z	trial point in univariate optimization

CHAPTER I

INTRODUCTION

1.1 Background and Motivation

A system which can accurately identify the source of radio transmissions would be an invaluable tool for government, military, or civilian situations where unauthorized use of the electromagnetic spectrum occurs [Shaw94], [Marc92], [CRC92]. For example, it would be especially useful for authentication of air traffic control or police dispatch radio transmissions. Alternatively, such a system would provide evidence for the prosecution of persons engaging in illegal use of radio transmitters. In a military theatre of operations, the ability to identify the source of radio transmissions, in combination with standard electronic warfare assets such as direction-finding and triangulation systems, would provide commanders with invaluable strategic information.

The concept of transmitter transient analysis and classification has been proposed by Kinsner in 12 internal reports, starting from 1993, and different parts of the implementation have been studied by his students [Diet94], [Ruda94], [Shaw94], [Ande95], [Khan95], [Kwok95], [Toon95], [Toon97]. Specifically, the transient data acquisition system used for collecting data for this research is described in [Kwok95], [Toon95], and [Toon97]. It is also summarized in Section 6.1 of this thesis.

This thesis focuses on a method of classifying radio transmitters by analysing the transient which occurs at the start of each transmission. The method requires three distinct stages for the classification of a raw radio transmission. These stages are as follows:

1. **Preprocessing;**
2. **Feature extraction; and**
3. **Classification.**

A short discussion about each stage, along with a description of the various problems which exist in their implementation, will now follow.

1.1.1 The Preprocessing Stage

The preprocessing stage separates a transient from a raw signal recorded at 44.1 kilosamples per second (ksps) and 16 bits per sample. Each recording contains 16,348 samples or almost 32 kilobytes (kB) of data. The signals contain ambient channel noise which is followed by the start of a radio transmission similar to that shown in Fig. 1.1. Due to the nonstationary nature of transmitter transients, though, the task of separating the transient from the channel noise is very difficult. It involves finding the exact time when the ambient channel noise, which is correlated to some unknown degree, ceases, and the transient begins. However, despite being completely deterministic, many transients exhibit characteristics similar to noise due to their high degree of irregularity. Thus, to some extent, we are left with the problem of separating noise from noise with a different degree of correlation. The approach presented in this thesis utilizes a multifractal analysis to characterize the degree of irregularity along the duration of the signal. If, within this characterization, a significant change occurs, it triggers the start of the transient and a fixed number of consecutive samples can be removed starting at that point. In this thesis, the separated transient contains 2 kilosamples (4kB) and is indicated between the dashed vertical lines in Fig 1.1.

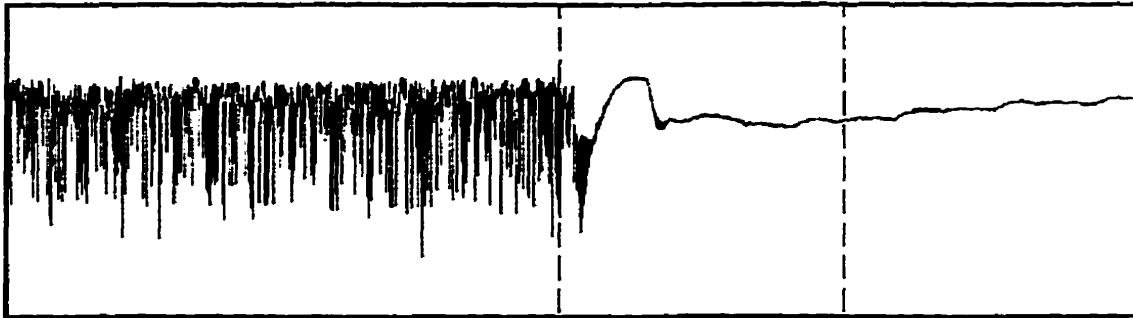


Fig. 1.1. Ambient channel noise followed by radio transmission.

1.1.2 The Feature Extraction Stage

After the transient has been separated from the less relevant data in the raw signal, it must be further reduced in size before it can be classified in a reasonable amount of time and to a high degree of accuracy. The feature extraction stage provides a more efficient representation of the transient in terms of data relevance and storage requirements. Characterization of significant features within a transient is achieved through the use of a multifractal analysis similar to that used in the preprocessing stage. The resulting signal is a much more consistent representation of the primarily nonstationary transient signal and, more importantly, can be significantly reduced in size. Success has been achieved with a reduced transient size down to 32 elements. Also, using a transformation of the multifractal model, data reduction down to 8 elements is accomplished.

1.1.3 The Classification Stage

The final stage of this system is the classifier itself. We use a neural network model for classification. Two potential neural network models are worth consideration. These are the standard feedforward neural network, often referred to as the Backpropagation Network (BPN) [RuMc86], and the increasingly popular Probabilistic Neural Network (PNN) [Spec88], [Mast95]. After an analysis of both types of networks, the PNN is chosen

because of its fast training speed, highly accurate results, and sound mathematical foundations.

1.2 Thesis Statement and Objectives

The purpose of this thesis is to present a reliable system for classification of transmitter transients using fractal segmentation, fractal modelling, and neural networks.

To accomplish this purpose, the following objectives are met:

- A consistent method for separating transmitter transients from ambient channel noise is required;
- A technique which yields compact and highly representative models of transmitter transients is required; and
- An effective neural network architecture for classification of multifractal transient models must be implemented.

1.3 Thesis Organization

The organization of this thesis broadly reflects the sequence of the classification scheme and, more specifically, the logical development of the research. Sufficient background information is provided to fully describe each module in the classification scheme. This includes an analysis of problems encountered throughout the research along with the motivation for selecting various techniques used to solve these problems.

Chapter 2 begins with a comparative discussion of fractal dimensions and the standard integer (Euclidean) dimensions. Various fractal dimensions are outlined, and multifractality is discussed as an appropriate method for characterization of nonstationary signals. The variance fractal dimension is selected for use in this application. Details for

calculating the variance fractal dimension are provided and the concept of fractal trajectory is explained.

Chapter 3 provides the basis for implementing the variance fractal dimension trajectory in the roles of segmentation and feature extraction. The scope of this discussion is limited to the direct application of processing the nonstationary transients from radio transmitters. The parameter settings required to distinguish between these two roles are discussed, along with an explanation of how efficient data reduction can be realized.

Chapter 4 commences with an analysis of the PNN relative to the popular BPN and standard statistical classification techniques. This demonstrates the specific motivation for the selection of the PNN as the classifier used in this system. Primarily though, a detailed description of the PNN is given along with specific information about its implementation in this thesis.

Chapter 5 presents the detailed design of the system. This includes a description of the program designed for testing the thesis, TAC-MM, with specific regard to its various features and limitations. This chapter then discusses the methodology used for verifying or proving the implementation of each module.

Chapter 6 presents the experimental results achieved in this thesis. This is, however, preceded by a description of the test data used in the experiments. An analysis of the results explains exactly what has been accomplished and addresses the important issue of confidence measures.

Chapter 7 provides conclusions, based on the experimental results, as to the feasibility of the system described and the contributions made by this thesis. The thesis closes with recommendations for further research into this topic.

CHAPTER II

FRACTAL DIMENSION AND MULTIFRACTALITY

This chapter focuses on definitions, applications, and computational details of fractal dimensions and multifractality. Section 2.1 begins with a general discussion on this topic. Several different classes of fractal dimensions are discussed, and the variance-fractal dimension is chosen as the most appropriate modelling method for this application. Section 2.2 presents a detailed procedure for calculating the variance fractal dimension complete with pseudocode for implementation of the main iterative loop. Specific emphasis is placed on the proper selection of various parameters to ensure that the calculation is accurate. A measure of local fractal dimension is discussed in Section 2.3, and multifractality is further defined to include signals which exhibit time-varying fractal dimensions. Finally, the variance fractal dimension trajectory is described as the multifractal measure to be used for segmentation and feature extraction of transients.

2.1 Fractal Dimensions

The Euclidean dimension, D_E , of an object is often considered to be the smallest possible integer space onto which the object can be embedded. It is well known that the dimension of a point is 0, the dimension of a line is 1, the dimension of a plane is 2, and the dimension of a volumetric object is 3. However, the concept of dimension can be generalized further to include fractional quantities as well as integers. These are referred to as fractal dimensions.

A fractal dimension, D , can be interpreted as the “degree of meandering” (or roughness, brokenness, and irregularity) of an object [Kins94a]. For example, although a coastline is immeasurable in terms of length, it has a certain characteristic degree of meandering. If the object is regular in shape, it will have a fractal dimension which is the same as its topological dimension. If, however, the object is irregular, the fractal dimension will be higher. Thus, the concept of fractal dimension provides us with a means to characterize the geometry of any shape, object, or signal.

Various formal fractal dimension definitions exist. According to the information content under consideration, these can be classified into the general categories of morphological, entropy, spectral, and variance based dimensions [Kins94a]. Each of these general categories will now be discussed along with a brief description of their implementation. Then, it will be shown why the variance fractal dimension is best suited for fractal modelling of univariate temporal signals, such as the transmitter transients being analysed in this thesis.

2.1.1 Morphological-Based Dimensions

A morphological-based dimension characterizes the geometrical complexity of a fractal object. The morphological dimension of a fractal object is also known as the Hausdorff-Besicovitch dimension, D_{HB} . In practice, however, D_{HB} is nearly impossible to calculate, and many different ways to approximate it have been developed. One of the more popular methods, due to its relative simplicity in calculation, is the Hausdorff mesh dimension, D_{HM} . It is calculated by first defining a covering set made up of volume elements, or vels for short [Kins94a], of size v and aligned symmetrically as in Fig. 2.1. Then, a quantity, N_v , is obtained by counting the minimum number of boxes required to

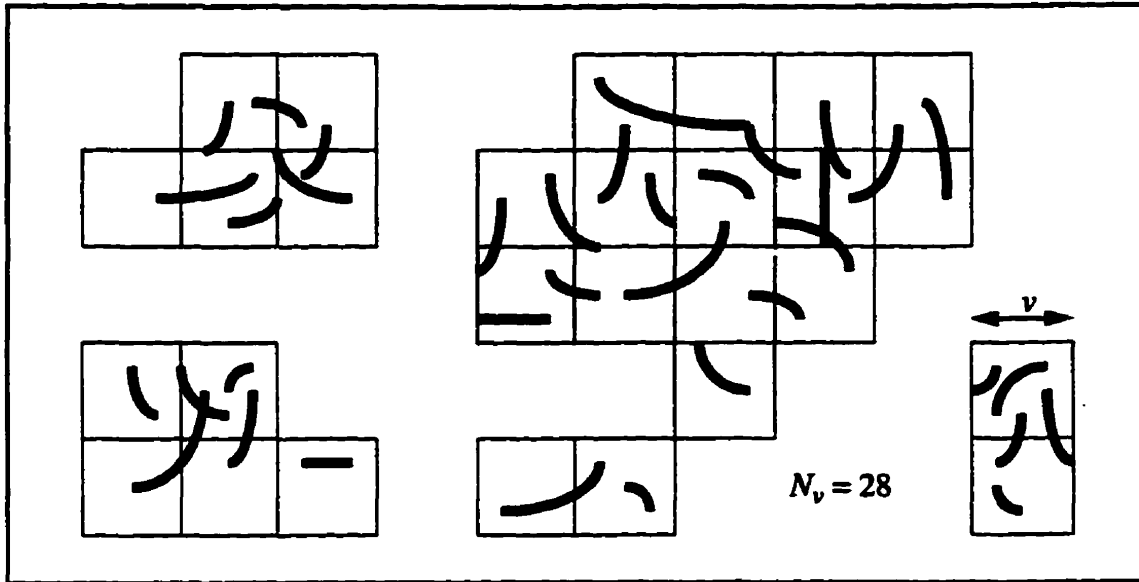


Fig. 2.1. Covering set made up of symmetrically aligned vels (After [Kins94a]).

completely cover a given object. Appropriately, the dimension discussed here is commonly called the box counting dimension.

If we assume that a power-law relationship exists between the size of the vel and the number of covering vels at each vel size as in

$$v \sim N_v^{-D_{HM}} \quad \forall v \quad (2.1)$$

then D_{HM} can then be obtained from

$$D_{HM} = \lim_{v \rightarrow 0} \frac{\log N_v}{\log\left(\frac{1}{v}\right)} \quad (2.2)$$

Solving for D_{HM} in this equation requires an iterative process in which N_v is determined for different sizes of $v = v_i$ and plotted on a log-log plot where

$$\psi_i = \log N_{v_i} \quad (2.3)$$

and

$$z_i = \log\left(\frac{1}{v_i}\right) \quad (2.4)$$

for every value of v_i . D_{HM} is the slope of the approximating straight line in the log-log plot.

2.1.2 Entropy-Based Single and Multifractal Dimensions

As the previously described morphological-based dimensions consider only geometric features, they are suitable only when the distribution of the fractal is uniform. Entropy-based dimensions, however, can be used when the distribution of a fractal is not uniform. In these dimensions, the probability distribution of a dynamical system or the distribution of a measure of a spatial fractal is taken into account along with its geometrical features. One popular entropy-based dimension is the information dimension, D_I [Kins94a]. To calculate the information dimension, we will again consider a covering of N_v vels, each with a diameter v . Then, D_I can be obtained from

$$D_I = \lim_{v \rightarrow 0} \frac{H_v}{\log(1/v)} \quad (2.5)$$

where H_v is the Shannon entropy of the fractal given by

$$H_v = - \sum_{j=1}^{N_v} p_j \log p_j \quad (2.6)$$

and p_j is the probability of intersection of the fractal with the j th vel. To calculate p_j , we determine the relative frequency, n_j , with which the fractal intersects the j th vel as

$$p_j = \lim_{N_v \rightarrow \infty} \frac{n_j}{N_F} \quad (2.7)$$

where

$$N_F = \sum_{j=1}^{N_v} n_j \quad (2.8)$$

Notice that N_F refers only to the part of the fractal covered by the vels, and increases as the size of the vels decreases. The information dimension, D_I , as described by Eq. 2.5, can now be obtained from the slope of a log-log plot as before.

An improvement to the information dimension is the correlation dimension, D_C . It provides consideration to more than just entropy, it accounts for the correlation between pairs of neighbouring points on the fractal [Kins94a]. To describe D_C we will consider a covering set made up of N_v vels of size v as in the information dimension. We will again assume that the j th vel is intersected by the fractal with a frequency n_j . The probability of the j th vel, p_j , will then be defined as

$$p_j = \lim_{N_v \rightarrow \infty} \frac{n_j}{N_F} \quad (2.9)$$

where

$$N_F = \sum_{j=1}^{N_r} n_j \quad (2.10)$$

Notice that this probability definition is identical to that used in the information dimension. Now, assume that the power-law relationship shown in Eq. 2.11 holds between the sum of the squared probabilities over all of the vels of size ν [Kins94a],

$$\sum_{j=1}^{N_r} p_j^2 \sim \nu^{D_C} \quad (2.11)$$

Then, the correlation dimension is given as

$$D_C = \lim_{\nu \rightarrow 0} \frac{\log \sum_{j=1}^{N_r} p_j^2}{\log(\nu)} \quad (2.12)$$

Solving for D_C in the above equation can be done as before, by an iterative process where data for different values of ν are plotted logarithmically and the slope of the approximating line is determined.

The information dimension, D_I , and the correlation dimension, D_C , are both special cases of the generalized entropy dimension, or, the Rényi dimension [Rény55]. The Rényi dimension utilizes a generalized entropy function as given by

$$H_q = \frac{1}{q-1} \log \sum_{j=1}^{N_r} p_j^q \quad -\infty < q < \infty \quad (2.13)$$

where q is a value called the *moment order*. Notice that for $q = 2$, H_2 becomes the correlation integral from the numerator in Eq 2.12. It can also be shown that for $q = 1$, H_1 is equivalent to the information dimension [Kins94a]. Based on the generalized entropy function, Rényi's generalized fractal dimension is given by

$$D_q = \lim_{v \rightarrow 0} \frac{1}{q-1} \frac{\log \sum_{j=1}^{N_v} p_j^q}{\log(v)} \quad (2.14)$$

If different dimension measurements are obtained while varying q , a fractal object has a non-uniform probability distribution and is said to be multifractal. Calculating the Rényi dimension of a multifractal object for different values of q gives a range of variation in D_q which could indicate the characteristic degree of complexity for that object [Kins94a]. This type of multifractal measurement lends itself to a wide assortment of applications such as modelling of dynamical systems, texture analysis of images [FeKi95], and the study of nonstationary signals.

2.1.3 Spectrum-Based Dimensions

A spectrum-based dimension characterizes a fractal signal using spectral analysis techniques such as the Fourier transform. Specifically, the spectral dimension, D_β , of a signal is determined from its power spectrum density, $P(f)$, which is calculated by

$$P(f) = \lim_{T \rightarrow \infty} \frac{1}{T} |V(f)|^2 \quad (2.15)$$

where $V(f)$ is the signal's Fourier transform. Then, a value called the power spectrum exponent, β , can be determined from

$$P(f) \sim \frac{1}{f^\beta} \quad (2.16)$$

by applying a curve-fitting algorithm to estimate the slope of a log-log plot, as in Fig. 2.2.

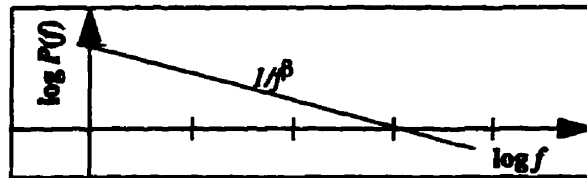


Fig. 2.2. A graph of $\log(P(f))$ vs $\log(f)$ for finding the power spectrum exponent, β .

Finally, we can calculate D_β from

$$D_\beta = E + \frac{3-\beta}{2}, \quad (1 \leq \beta \leq 3) \quad (2.17)$$

where E is the embedding Euclidean dimension, which is one for a time series with a single independent variable.

2.1.4 Variance-Based Dimensions

Variance-based dimensions, like the spectrum-based class, are used for characterizing the fractal components of a time series. To calculate the variance dimension, D_σ , let the time series of interest be defined as a signal, $B(t)$, which is continuous in time, t . Then, the variance, σ^2 , of its amplitude changes over a time increment and is related to the time increment according to

$$\text{Var}[B(t_2) - B(t_1)] \sim |t_2 - t_1|^{2H} \quad (2.18)$$

where H is a value called the Hurst exponent. By setting $\Delta t = |t_2 - t_1|$, and $(\Delta B)_{\Delta t} = B(t_2) - B(t_1)$, we can determine H from

$$H = \lim_{\Delta t \rightarrow 0} \frac{1}{2} \frac{\log [\text{Var}(\Delta B)_{\Delta t}]}{\log(\Delta t)} \quad (2.19)$$

Finally, the variance dimension can be determined from

$$D_{\sigma} = E + 1 - H \quad (2.20)$$

where E is the embedding Euclidean dimension and has a value of one for a time series with a single independent variable.

2.1.5 Fractal Dimensions for Transient Analysis

In this application, we are required to determine the dimension of a single variable temporal signal. For the first two fractal classes, morphologically-based and entropy-based, the transient would first have to be plotted onto some two dimensional surface before analysis can be performed. In this situation, a problem arises because we do not know what the relative scale between the time axis and the signal should be in order to achieve accurate dimension calculations. However, assuming that an appropriate scaling factor has been found, we are still left with a massive ($2^{13} \times 2^{16}$ bits) array of data to process. The memory required, and especially, the processing time involved here would be far too large for a practical implementation.

The spectral and variance dimensions are, perhaps, more appropriate for dimension calculation of a univariate temporal signal. However, in calculating the spectral dimension, we are limited to window sizes which are powers of two if the Fast Fourier Transform (FFT) is used. Discrete Fourier Transform methods for other window sizes would require unreasonably large amounts of processing time. Also, the choice of window size in the Fourier transform may introduce artifacts which could seriously affect the accuracy of the results [Kins94a]. In comparison, the variance dimension can be performed on any sufficiently large window size. Its primary advantage, though, is that since it does not require a window in the Fourier sense, no window artifacts are introduced. Thus, the variance dimension, D_G , is the best choice for this particular problem.

2.2 Calculating the Variance Dimension

In this section, the algorithm used for calculating the variance fractal dimension is discussed. Emphasis is on the various parameter settings which must be considered in a general implementation of this algorithm. Specific details for analysis of transients in the segmentation and feature extraction roles will be left for Chapter 3. For this discussion, consider a signal sampled over a time interval, T , with a constant sampling rate of $1/\delta t$. This produces a sample space with N_T samples collected at equal time intervals, δt .

Prior to calculating the variance dimension of a signal, we must establish the number, K , of ordered pairs, and thus the number of time increments, which will be required for finding H in Eq. 2.19. This must be considered in conjunction with the

increasing sequence of time increments, $\{\Delta t_1, \Delta t_2, \dots, \Delta t_K\}$, which will be used. It is very important that the time increments not exceed the length of the signal,

$$\Delta t_K \leq T \tag{2.21}$$

as unpredictable results will occur. Since the variance pairs are to be spread evenly on a log-log plot, successive time increments should follow a sequence which is either linear or dyadic. Thus, to satisfy the condition at Eq. 2.21, it is required that

$$\Psi^K \leq N_T \tag{2.22}$$

where $\Psi = 1$ for a linear sequence and $\Psi = 2$ for a dyadic sequence. Fig 2.3 shows an example of a dyadic sequence with $N_T = 256$ and $K = 5$.

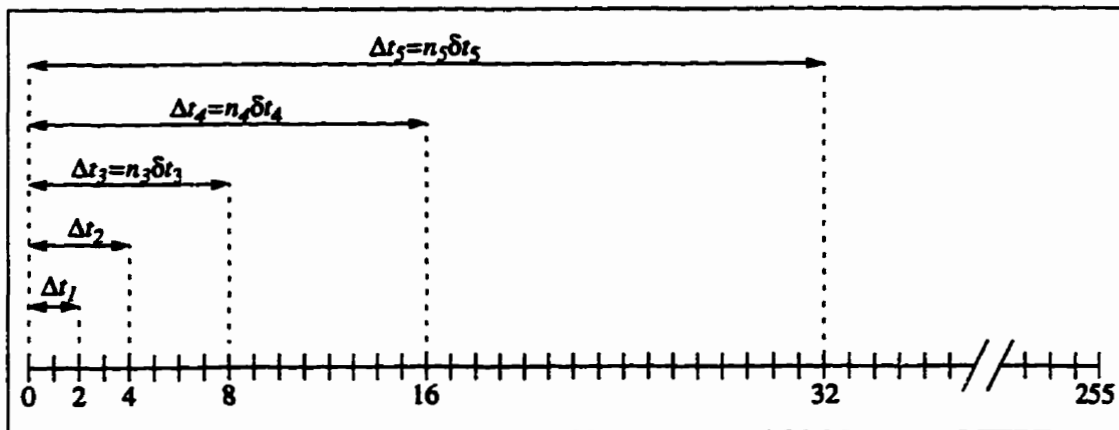


Fig. 2.3. Example of dyadic sequence with $N_T = 256$ and $K = 5$.

Now that the sequence and number of time increments have been selected, $Var(\Delta B)_k$ can be calculated for each time increment according to

$$\text{Var}(\Delta B)_k = \frac{1}{N_k - 1} \left[\sum_{j=1}^{N_k} (\Delta B)_{jk}^2 - \frac{1}{N_k} \left(\sum_{j=1}^{N_k} (\Delta B)_{jk} \right)^2 \right] \quad (2.23)$$

where $N_k = N_T - n_k$. Then the values (x_k, ψ_k) can be calculated for the log-log plot by

$$x_k = \log n_k \quad (2.24)$$

$$\psi_k = \log [\text{Var}(\Delta B)_k] \quad (2.25)$$

Implementation of Eqs. 2.23, 2.24, and 2.25 for all time increments can be done using nested loops such that the outer loop cycles through each time increment and the inner loop cycles through each sample in the signal according to the pseudocode of Fig. 2.4.

```

for k = 1 to K {
  sum1=sum2=0.0
  n_k = \Psi^k
  N_k = N_T - n_k
  for n = 1 to N_k {
    \Delta B = B_{n+n_k} - B_n
    sum1+ = \Delta B * \Delta B
    sum2+ = \Delta B
  }
  Var(\Delta B)_k = (sum1 - (sum2^2/N_k))
                  / (N_k - 1.0)
}

```

Fig. 2.4. Pseudocode for variance fractal dimension calculation.

Then, rather than from a plot, H can be determined using least squares regression (LSR) as in

$$2H = \frac{K \sum_{i=1}^K x_i \psi_i - \left(\sum_{i=1}^K x_i \right) \left(\sum_{i=1}^K \psi_i \right)}{K \sum_{i=1}^K x_i^2 - \left(\sum_{i=1}^K x_i \right)^2} \quad (2.26)$$

Finally, the variance dimension, D_G , is given by Eq. 2.20 using the value obtained for H . Typical values for D_G are 1.0 for a highly periodic and well behaved function such as a sine wave and increase to 2.0 for completely uncorrelated white noise.

2.3 Fractal Measures vs. Time

2.3.1 Local Fractal Dimensions and Multifractality

A discussion on the concept of local fractal dimension is required before a more detailed description of multifractality can be given. A local fractal dimension refers to a dimension value obtained from calculations on a limited area of the fractal shape, object, or signal. The purpose of using this limited area is to assign a precise dimension to a specific point in the fractal. A common approach to calculating local fractal dimensions is by using a moving window that selects data from only within its boundaries. It is straightforward to apply this technique to a univariate signal, $B(t)$, by sliding the window, which has fixed width, W , along the time axis as in Fig. 2.5.

A shape, object, or signal can be referred to as a pure fractal if its local fractal dimension is the same everywhere. However, a transient or other nonstationary signal, is not pure fractal because its fractality is time varying. This type of signal could be more appropriately characterized by a model consisting of several local fractal dimensions taken at different points in time. An extension to the concept of multifractality, as previously

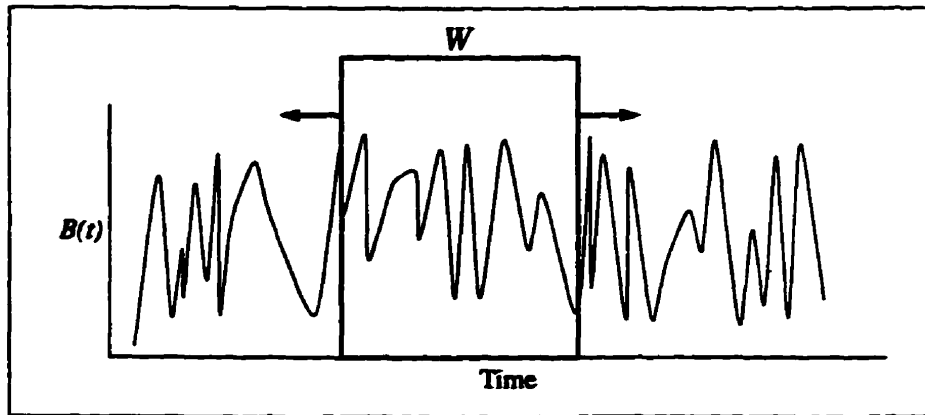


Fig. 2.5. Window for calculating the local fractal dimension of a signal.

discussed, deals with signals or objects which have varying local fractal dimensions. A fractal measure over a support set is called multifractal if local measures on different parts of the support have different dimensions [Vics92].

2.3.2 Variance Fractal Dimension Trajectory

It should now be clear that the variance fractal dimension calculation can readily be applied to a multifractal signal by finding the local fractal dimensions for successive portions of the signal. When these local fractal dimensions are considered in sequence, we have a multifractal characterization which can be referred to as the variance fractal dimension trajectory [Grie96].

Referring back to Fig 2.5, the ideal size for the window width, W , is somewhat ambiguous and depends on the nature of the signal being analysed. A relatively large window size will cause the dimensions of locally distinct fractals to be buried within the dimension of their most significant neighbouring fractal and, at the same time, will be computationally expensive. However, using a window size which is too small will provide insufficient data for the analysis. Theoretically, the proper size of the window should result in the variance dimension being equal to a constant value Rényi dimension. Selecting the

window size for signal segmentation and feature extraction in transients will be further discussed in Chapter 3.

It is also very important to select an appropriate spacing, S , between successive windows. To ensure that the entire signal is considered, selection of the window spacing should be in the range from a single sample up to the width of the window. If the window spacing is too small, calculation time becomes an issue. If it is too large, important details in the trajectory could be lost. Chapter 3 will also describe this parameter in the specific contexts of signal segmentation and feature extraction.

2.4 Summary of Chapter 2

This chapter described the use of fractal dimensions to characterize shapes, objects, and signals. After a brief summary of various dimension calculation techniques, the variance fractal dimension was selected for transient analysis and described in detail. Multifractality was then discussed and the variance fractal dimension trajectory was presented as a useful extension to the variance dimension. Chapter 3 will now describe the parameter settings used for finding the variance fractal dimension trajectory in both the segmentation and feature extraction modes of this system.

CHAPTER III

MULTIFRACTAL MODELLING OF TRANSIENTS FOR SEGMENTATION AND FEATURE EXTRACTION

In this chapter, specific information is provided for implementing the variance fractal dimension trajectory in the roles of segmentation and feature extraction. Specifically, Section 3.1 presents the motivation for using this technique in the preprocessing stage where segmentation of the transient from the ambient channel noise occurs. The parameter settings required to implement fractal segmentation are presented in detail and the problem of aligning the trajectory to the original signal is addressed. In Section 3.2, the variance fractal dimension trajectory is explained in its role as a feature extractor for the second module of our classification system. Again, the parameter settings required for this technique are explained in detail. It is also shown how significant data reduction can be realized simply by changing S , the window spacing parameter.

3.1 The Preprocessing Stage

3.1.1 Motivation

In the context of removing a transient from a raw signal for neural network classification, finding the exact start of the transient is critical. If a portion of the signal containing the transient is removed at a position from before the actual start of the transient, the classifier will be forced to deal with an arbitrary amount of irrelevant channel noise. This would lead to highly unreliable results despite that the neural network would eventually learn to ignore the beginning of each transient. Obviously, when

successive transients like this are processed, the network would be constantly trying to adapt to a varying duration of leading noise. The problem compounds itself when the transient is extracted too late from the raw signal. In this situation, part or all of the most relevant information could be completely truncated. No statistical, human, or neural network classifier stands a chance under such circumstances.

3.1.2 Fractal Segmentation

As discussed earlier, the variance fractal dimension trajectory is a good tool for studying the local fractality of a signal. However, for this method to work for segmentation of a transient from the ambient channel noise before the transient, it must be assumed that the channel noise and the transient exhibit different multifractal characteristics. Figure 3.1 shows how a transient can be segmented from noise using the fractal trajectory.

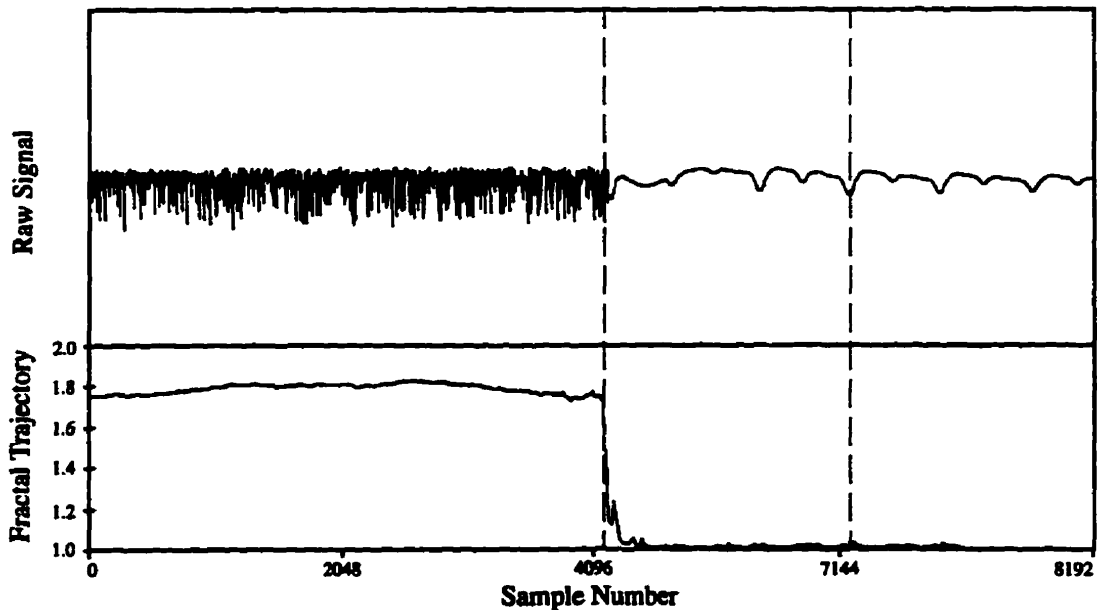


Fig. 3.1. Separating the transient from the ambient noise in a raw signal.

The figure shows the ideal scenario where the dimension of the transient is much lower than the dimension of the ambient channel noise. In this case, it is straightforward to find the start of the transient. However, in many situations, it is difficult to determine what type of change in the fractal trajectory actually marks this point in time. Sometimes, the dimension of the transient is much closer to or, in rare circumstances, even higher than the dimension of the ambient channel noise.

A suitable way to trigger the start of the transient is to find the earliest time when the dimension, $D_{\sigma}(t)$, compared with the mean, $\mu_{D_{\sigma}(t)}$, of the portion of the raw signal containing noise is sufficiently different as follows:

$$|D_{\sigma}(t) - \mu_{D_{\sigma}(t)}| > (\tau \cdot \mu_{D_{\sigma}(t)} + \sigma_{D_{\sigma}(t)}) \quad (3.1)$$

where τ is a certain threshold. The term, $\sigma_{D_{\sigma}(t)}$, is the standard deviation of the signal and is intended to compensate for ambient channel noise which has time-varying fractality within itself. Also, in this implementation, the mean, $\mu_{D_{\sigma}(t)}$, and the standard deviation, $\sigma_{D_{\sigma}(t)}$, are calculated only for $t=1...T/4$. Thus, we have imposed a simple limitation on the format of the raw signal: that it must contain a minimum $T/4$ samples of ambient channel noise before the onset of the transient.

3.1.3 Segmentation Parameters

This section will discuss the variance dimension parameters which are best suited for segmentation. The most straightforward parameter here is setting the spacing parameter, S , for the sliding window. This should be set as near as possible to 1 such that calculation time remains within the tolerance of the user. Theoretically, setting this value to 1 will ensure that no significant transition is missed and allow for exact thresholding in

the time scale as determined by Eq. 3.1. Setting the window size is not as critical. Since we are looking for a large transition and not fine details, we can set it a little on the high side to ensure statistical validation. It has been found that a window size, W , on the order of 2^9 is sufficient for this purpose.

The sequence of time increments, Δt , and the number of them, K , must also be set. For segmentation, a linear sequence of time increments, $\Delta t_k = \{1\delta t, 2\delta t, \dots, K\delta t\}$, is most appropriate [GrKi94]. It should be noted that adjacent samples, among other pairs, are compared here. This ensures that highly uncorrelated noise will yield high dimensions and more correlated signals will show lower dimensions.

The number of time increments to consider, and thus the number of ordered pairs obtained for the LSR algorithm, must be selected carefully. It was found that if too few increments were considered, spikes were occurring in the trajectory during significant transitions. At first, the notion that these spikes could be used as trigger points was considered. However, it was quickly determined that these spikes were merely artifacts of the dimension calculation caused by sudden scale changes at transition points. Since the transition points seldom had consistent, if any, scale changes, this idea was abandoned. The spikes found at the transition points could virtually be eliminated by increasing the number of ordered pairs significantly. We are, however, limited by the size of the window, W , and the amount of time that it takes to calculate the trajectory. It was found that a value of about $K = 10$ produced reasonable results for this class of signals.

Finally, the threshold, τ , must be set to trigger the start of the transient. At this stage, we have determined τ by trial and error. Despite the existence of the compensation term, $\sigma_{D\sigma(t)}$ in Eq. 3.1, there is still no guarantee that the channel noise will not show

significant changes as to falsely trigger a transient. Also, some transients may not have fractal dimensions which are as different from the channel noise as others. In this case, the extraction will not be triggered until after the onset of the transient and much valuable information will be lost. Some success has been achieved by setting τ to some value around 0.08 (8%), but this depends largely on the settings of the fractal parameters. Also, it should be noted that selection of this value, in conjunction with the comparison scheme of Eq. 3.1, is the most significant obstacle in this system and warrants future consideration.

3.1.4 Alignment Issues

Since finding the exact start of the transient is so critical, it is necessary to correctly align the fractal trajectory with the original signal. At first, it was unclear whether the dimension calculated from each window should be aligned with the start of the window, the end of the window, or with the middle of the window to represent some sort of average. To resolve this issue, an extreme case, as shown in Fig. 3.2, was considered. This figure depicts white noise, followed by a pure sine wave, and then white noise again. The dimension was aligned with the first sample in each window of size 1024 samples. In theory, the dimension should be 2.0 for the noise portions and 1.0 for the sinusoid. However, analysis of the dimension shows that the transition from noise to signal occurs in the correct location while the transition from signal to noise occurs far too early. In fact, the transition from signal to noise occurs exactly 1024 samples too soon (the width of the window). The dimension of each window, in fact, becomes roughly the dimension of the most uncorrelated part of the signal within that window. Thus, it can be concluded that the dimension is to be aligned with the start of the window if we are searching for a

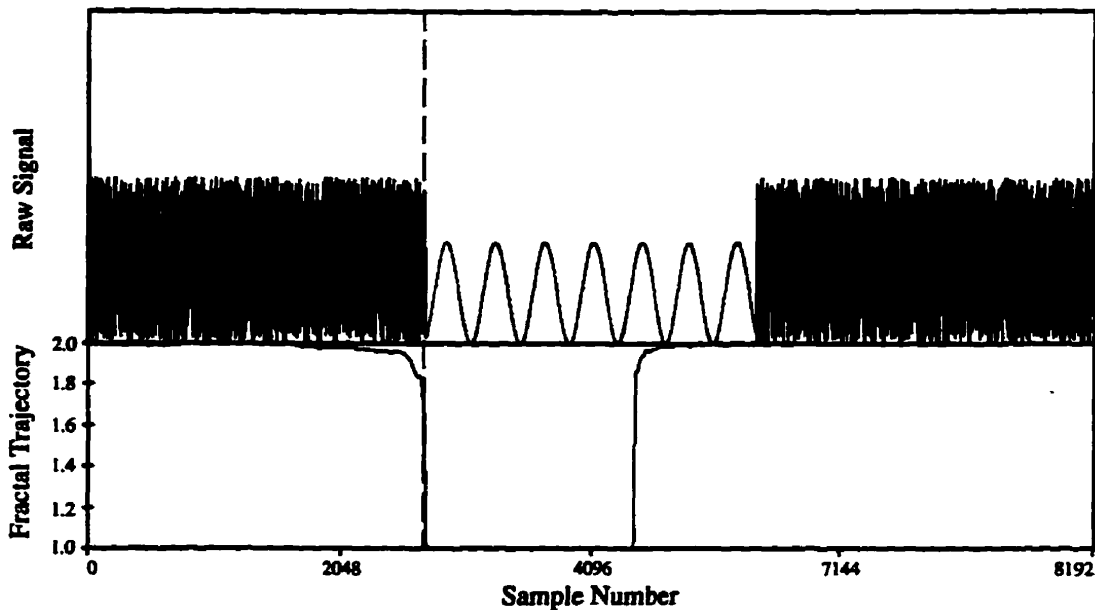


Fig. 3.2. Alignment of the fractal trajectory.

transition from high to low and with the end of the window for the opposite case. In the context of searching for the start of a transient, the vast majority of which exhibit a high to low transition, the dimension should therefore be aligned with the start of the windows.

3.2 The Feature Extraction Stage

3.2.1 Motivation

The next stage in this scheme is referred to as feature extraction. In general, feature extraction provides a more meaningful and significant representation of a signal [Lang96]. At the same time, the new representation contains fewer data points and thus facilitates more efficient manipulation and storage. When dealing with computer classification algorithms, this type of data reduction is very important. For example, the size of the transients being analysed in this system is 2048 samples, far more than could be realistically handled by most classifiers.

As previously stated, the variance fractal dimension trajectory is a multifractal model of a signal. It is made up of several local fractal dimensions and reveals the multifractal nature of the original signal. This system relies heavily on the existence of consistent and unique multifractality for a given class of signal to produce a sufficient characterization. Data reduction is achieved simply by increasing the amount that the window is shifted for successive calculations. For instance, if the transient is 2048 samples in length, a window shift parameter, S , set at 32 will yield 64 dimension values. This represents a significant data reduction and is much more manageable for neural network processing. Another feature provided by this analysis is data normalization. By yielding results which, when accurately calculated, are between 1.0 and 2.0 regardless of the range of the original signal, further scaling is unnecessary for neural network processing.

3.2.2 Multifractal Feature Extraction Parameters

The parametric settings for feature extraction are significantly different from those used for segmentation. Figure 3.3 shows the same transient selected in Fig. 3.1, except that it is already separated from the channel noise and the fractal trajectory reflects feature extraction parameters. Compared to the dimension trajectory calculations with segmentation parameters, significantly increased detail is noticeable. This provides the classifier with more distinctive features and a higher degree of separation between different classes of transients.

The parameter settings should begin with the window spacing being selected, as previously described, to determine the amount of data reduction which is to be achieved. It is important that this be selected conservatively so that significant fractal characteristics are not neglected in the modelling process. The size of the sliding window, W , should be

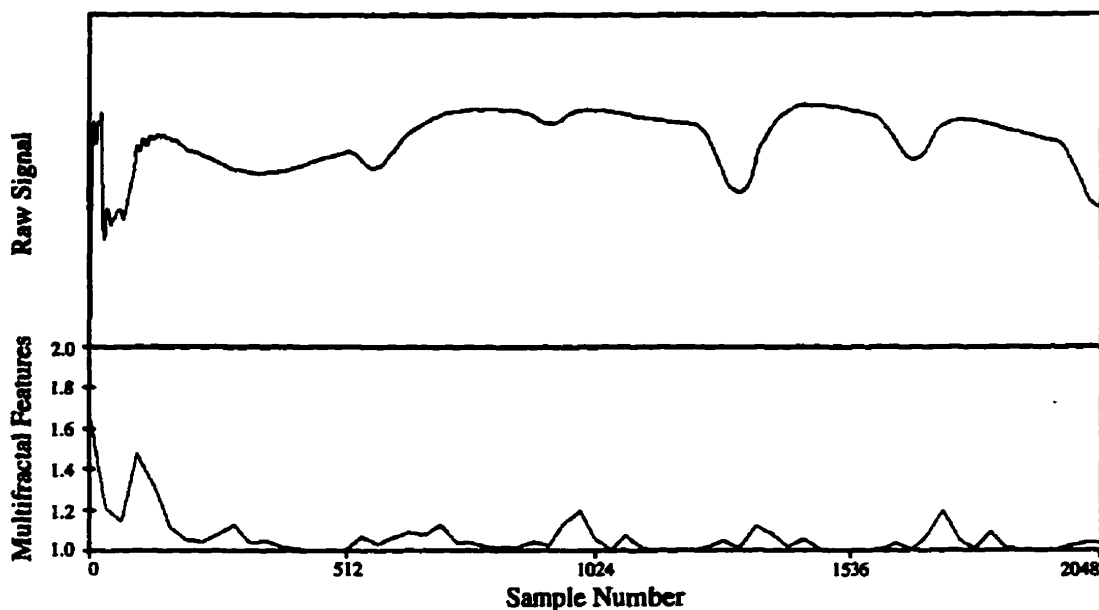


Fig. 3.3. An example of multifractal feature extraction.

set to a value that is close to that used for the segmentation. Setting $W = 2^8$ or 2^9 should be sufficient to allow statistical validity while not burying important fractal characteristics in an excessively generalized fractal dimension.

The sequence of time increments, Δt , is also quite different for feature extraction. A dyadic selection, such that $\Delta t_k = \{2\delta t, 4\delta t, \dots, 2^K\delta t\}$, is most appropriate for feature extraction [KiGr95]. Notice first that adjacent samples are not compared. This, combined with generally larger time shifts, causes the dimension to be somewhat higher and more varied than with the segmentation parameters. The number of time increments, K , considered here should be somewhat lower. This is largely due to the physical limitations imposed by the smaller sliding window and the faster growing dyadic time increments, Δt_k . A value near $K = 8$ has been found to produce favourable results.

3.3 Summary of Chapter 3

This chapter has shown how to use the variance fractal dimension trajectory for segmentation and for feature extraction. The most significant difference between the two modes is that the sequence of time increments is linear for segmentation and dyadic for feature extraction. It has also been shown that by simply adjusting the spacing between successive windows in the feature extraction mode, highly efficient data reduction can be achieved.

The first two modules of this transient classification system, preprocessing and feature extraction, will be revisited in Chapter 5 where the TAC-MM software modules are described and verification is performed. The next chapter, however, deals completely with the PNN and how it is implemented in the classification module of this system.

CHAPTER IV

THE PROBABILISTIC NEURAL NETWORK

Neural networks are often used as classifiers. The standard neural network used for classification is the multiple layer feedforward network (MLFN), often called the Backpropagation Network (BPN). It has proved to work well in many different applications and thus warrants consideration for any classification problem. Furthermore, this type of network, given sufficient training time and neural processing units, can model any linear or non-linear deterministic function. However, problems arise when we try to define "sufficient" with reference to the number of processing units and the required training time. It is not known how many processing units are necessary and, compounding this problem, training could take weeks for a large data set. However, through estimation, trial, and error, it is possible to find an architecture which works well. Another problem arises when more classes or samples must be added to the training set; a new architecture must be found and in all likelihood, the network would have to be trained again from the start. Worse still, as mathematical verification of a MLFN is impossible, its predicted results must usually be accepted on faith [Mast93]. This is unacceptable in our application.

A more suitable candidate for the classifier in this application is the Probabilistic Neural Network (PNN). Section 4.1 gives some general background information on neural networks and the PNN. It compares neural networks to standard statistical classification schemes and shows why neural networks are superior for most applications. Specific

problems associated with the popular MLFN neural network are discussed in detail and it is shown that the PNN overcomes these problems while improving classification. One important point is that the PNN has a strictly defined architecture, as shown in Section 4.2, which is not subject to trial and error testing. In Section 4.3, training of the basic PNN is discussed and algorithms are presented. Powerful extensions to the basic PNN architecture, as used in this thesis, are discussed in Section 4.4. Finally, Section 4.5 addresses the Bayes Classification paradigm and shows how the PNN can provide mathematically sound confidence levels.

4.1 Neural Network and PNN Foundation

4.1.1 Brief History of the PNN

The PNN is based on a statistical algorithm first proposed in 1972 [Meis72], long before neural networks even existed. Due to relatively large computational and storage requirements for its implementation, though, the algorithm was all but forgotten for several years. However, in 1988, Donald Specht showed how the algorithm could be split up into a number of simple processes which could operate in parallel, much like in neural networks [Spec88], [Spec90a]. Present day computers allow us to implement his technique, thus making the PNN a practical reality [Mast95].

4.1.2 Classification with Complex Class Distributions

The primary use of the PNN is for classification. Despite that good statistical techniques for classification have existed for years, there are some potential characteristics of the distribution of the transient data which would preclude their use. Thus, we will digress further to show the motivation for selecting a neural network classifier in this system.

Much like the MLFN and other neural network models, the PNN can handle decision surfaces which are as complex as necessary or as simple as desired [Spec90b]. To fully understand the impact of this capability we will consider a simple classification problem with two classes, X and O. Assume that members of these classes are fully described by two variables such that they can be plotted on a plane as in Fig. 4.1. An unknown class, ?, is also shown on the plot. The job of the classifier, given the information shown on the plot, would be to determine if the unknown class belongs to X or O.

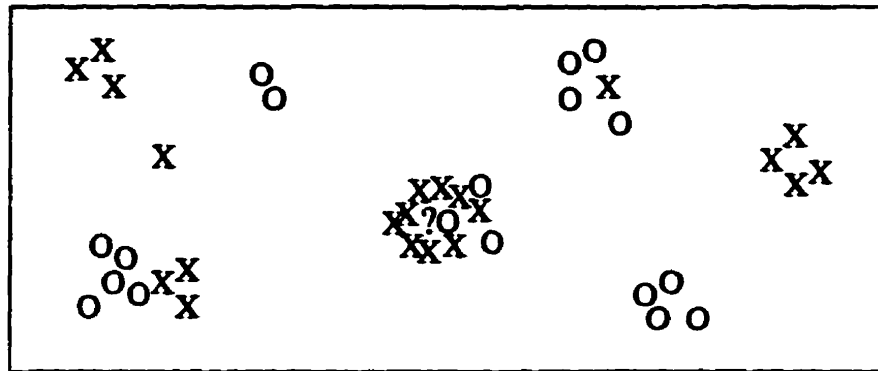


Fig. 4.1. Classes X and O plotted on a plane with one unknown (After [Mast93]).

Examination of Fig 4.1 shows that the unknown is probably a member of Class X. However, certain features of the distribution of these two classes would cause failure for standard statistical classification algorithms [Mast93], [Mast95]. First, the multimodal distribution of the classes, observed as several independent clusters for a single class, immediately precludes the use of centroid-based classification schemes which make the assumption that each class has a multivariate normal distribution and a single mean. In analysis of transmitter transients, these multimodal distributions will likely occur due to variances in transmission frequency, ambient channel noise, electronic component tolerances, and segmentation inconsistencies. A reasonable alternative would be to

abandon the normality assumption and use a nonparametric nearest-neighbour classifier. However, further inspection of the example in Fig 4.1 shows that this would also fail. Such an effect could be prevalent in our transient analysis due to inadequate representation of certain classes during training and, again, from segmentation inconsistencies.

Neural networks perform well under conditions which cause complex class distributions. They are known to be far superior in situations where a multimodal distribution of the population data exists. Furthermore, rather than considering only the nearest neighbour(s) around a certain data point, most neural networks can be configured to consider population densities throughout the entire domain of the distribution.

4.1.3 Advantages of the PNN Over the MLFN

The PNN has useful characteristics drawn from both neural networks and statistical analysis. First of all, as explained previously, the PNN has the neural network characteristic of being able to effectively handle even the most complex data distributions. At the same time though, it is based on established statistical principles which allow full insight into its operation and, in some situations, mathematically sound confidence levels. In fact, the classification abilities of the PNN approach optimal Bayesian, as will be further discussed in Section 4.5.

Another advantage of the PNN, in comparison to the MLFN, is that it has a strictly defined architecture. A corollary to this is that there are virtually no parameters which must be set by the user, allowing its implementation to be relatively straightforward. The tremendous impact of this characteristic can best be realized after attempting to implement a MLFN and select parameters such as the number of layers, the number of neurons in

each hidden and output layer, and various learning rate settings depending on the training algorithm being used.

When a MLFN encounters a training sample which is grossly erroneous, weight updates will be of a commensurately large order. This causes the entire network to overcompensate for a single sample and thereby reduces its ability to effectively model the entire data distribution without increasing the number of processing neurons. The PNN, in comparison, will handle such a situation with minimal problem and, as an added benefit, produce accurate results if the sample is in fact a valid data vector. For instance, it is possible that samples which may seem to be grossly erroneous only appear as such due to the common occurrence of having only sparse training data available. Thus, another feature of the PNN is that it works well in situations where training samples are relatively sparse [Spec90b].

The most important advantage of the PNN is that it trains orders of magnitude faster than a similarly tasked MLFN. In many cases, training the PNN is virtually instantaneous. The implications of this speed improvement are numerous. First, research time normally spent training the neural network can be better spent on other portions of a project or in reducing the total time spent on the project. For instance, with the transient classification system described in this thesis, more trials could be attempted with various parameter settings for the first two stages of the system. Alternatively, this training time reduction could lead to the future development of a system which could capture and learn transients directly in real-time.

As a final note, two studies conducted by the U.S. National Institute of Standards and Technology (NIST) will be cited. The first study compared several classical and

neural network techniques for classification on a database of FBI fingerprints [NIST5163]. The second study compared the same techniques for classification using a U.S. Postal Service database of handwritten digits [NIST5209]. The techniques compared were Euclidean Minimum Distance (EMD), Quadratic Minimum Distance (QMD), Normal Parametric Classifier (NRML), Single Nearest Neighbour (1-NN), k Nearest Neighbour (k-NN), Weighted Several Nearest Neighbour (WSNN), MLFN, Radial Basis Functions (RBF1 and RBF2), and the PNN. A significant result of both studies is that the PNN yielded the lowest error rate of all methods tested.

4.1.4 Disadvantages of the PNN

The principal disadvantage of the PNN is that the entire training set must be stored for the classification of unknown patterns. However, in the case of these transients, this will likely not become a significant problem. Using a very conservative data reduction parameter, each transient could be characterized by 64 double precision numbers for a total of 512 bytes of storage space. This would allow for storage of around 50,000 transients on a typical desktop computer (32 MB) with ample space remaining for an operating system and processing software.

The only other criticism of the PNN is that its classification speed is slow and therefore it is rarely suitable for real-time applications [Mast93]. However, with recent microprocessor technology, this no longer appears to be a problem. Tests done in this thesis, for a moderate sized problem, show classification time to be on the order of hundredths of a second on a standard desktop PC.

4.2 The Basic PNN

4.2.1 Architecture

At first glance, the architecture of the PNN resembles that of the MLFN. Figure 4.2 shows that the PNN consists of several layers of parallel processing units, or neurons.

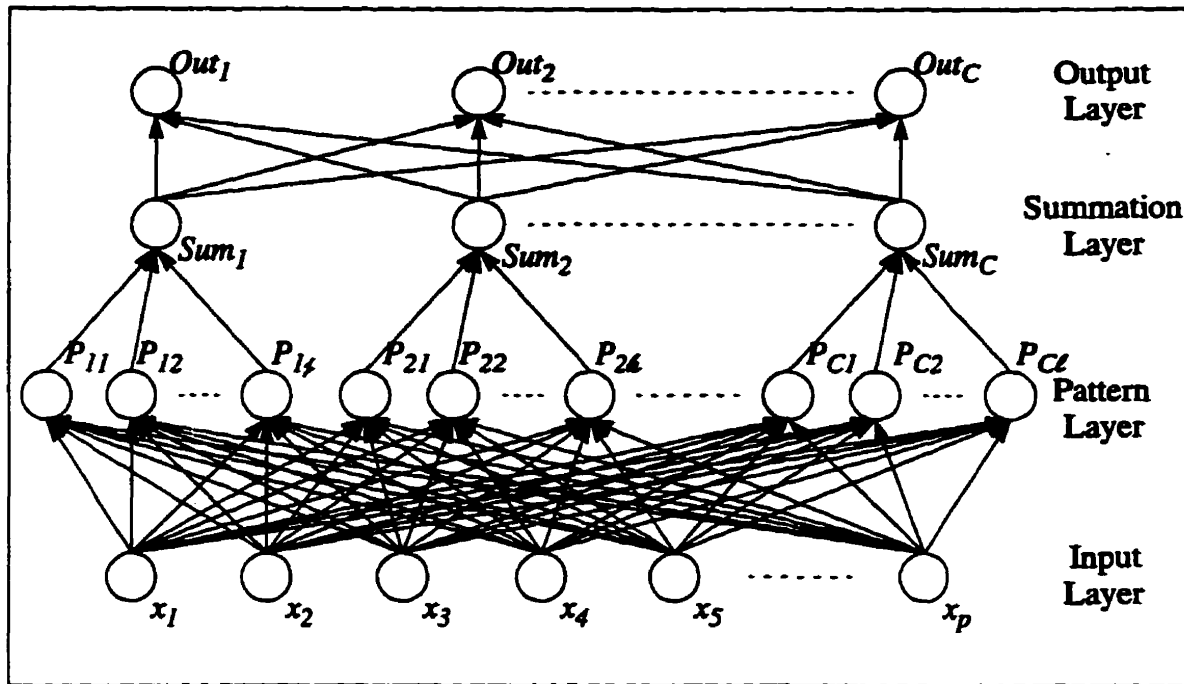


Fig. 4.2. Architecture of the PNN.

The exact number of neurons in each layer is determined by the data which exists for the training set. The number of input neurons, which serve no functional purpose other than to store the data for distribution to the next layer, is equal to the number of separable parameters used to describe the objects to be classified. Thus, the number of input neurons is equal to the number, p , of data points in a sample vector, X . The pattern layer can be considered to be two-dimensional and contains one neuron for each training case. Specifically, it represents a certain number of training samples ($\mu, \epsilon, \ell, \dots$) from each of C different classes for a total of R training samples. In brief, each neuron in the pattern layer

computes a distance measure between the unknown input and the training case represented by that neuron. An activation function, known as the Parzen window, is then applied to the distance measure as will be explained later in this section. In the summation layer, Fig 4.2 shows that there is one neuron for each of the C classes. These neurons sum the values of the pattern layer neurons corresponding to that class to obtain an estimated density function of the class. The number of neurons in the output layer is also equal to the number of classes, C . The output layer is often a simple threshold discriminator which activates a single neuron to represent the projected class of the unknown sample. In more advanced implementations, the neurons in this layer can bias the results to compensate for prior class probabilities and the cost of misclassifying a sample from a certain class. These factors will be further addressed in Section 4.5 during the discussion on Bayes Classification.

4.2.2 Parzen's Method of Density Estimation

Before a detailed description of processing in the PNN can be given, a suitable class-conditional probability density function (PDF) estimator must be reviewed. In 1962, Parzen presented a method of estimating a univariate probability density function from a random sample [Parz62]. His PDF estimator converges asymptotically to the true density as the number of samples increases towards a fully comprehensive representation of the class data. As shown in Fig 4.3, it finds the PDF by summing several bell-shaped weight functions, $W(x)$, each of which is assigned to a sample from the class. In this example, there are 8 samples from a single class in the training set centered at 0.15, 0.22, 0.27, 0.30, 0.40, 0.60, 0.65, and 0.85. Notice how this scheme allows for a multimodal distribution within the class. Parzen's density approximation is simply the sum of the 8 weight

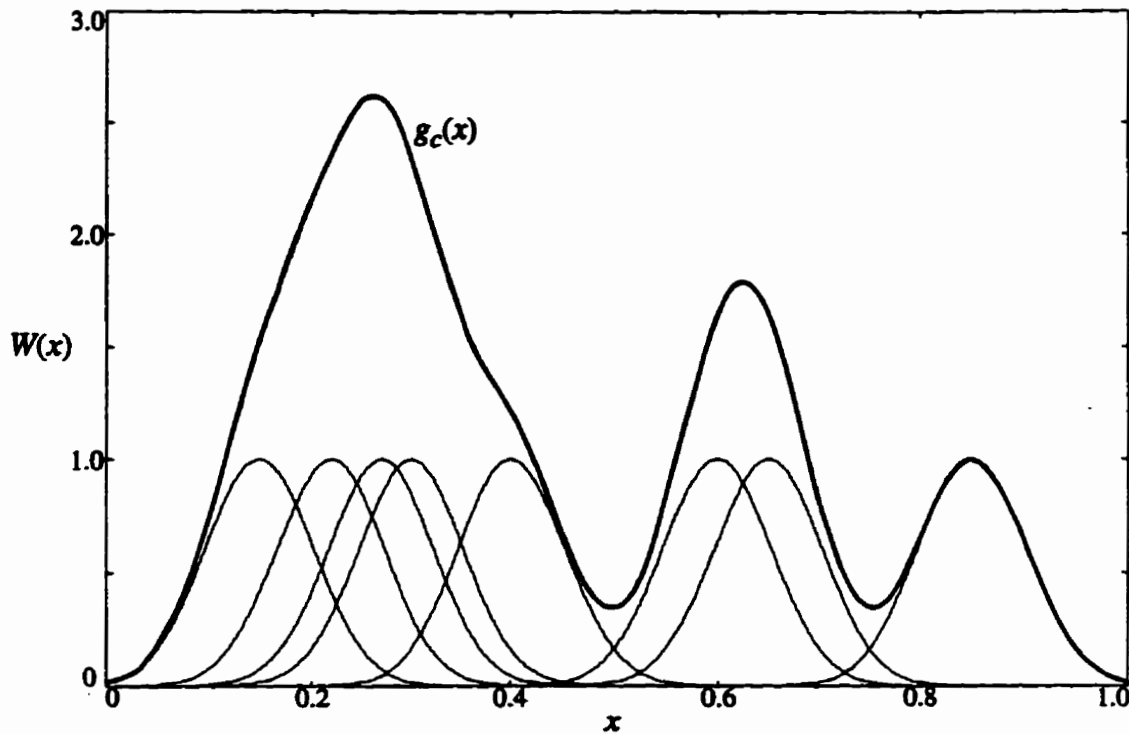


Fig. 4.3. Parzen's approximated PDF.

functions multiplied by a constant. Formally, if there are n_c training cases for a given class, c , then the estimated PDF for that class, $g_c(x)$, is

$$g_c(x) = \frac{1}{n_c \cdot \sigma} \cdot \sum_{i=1}^{n_c} W\left(\frac{x-x_i}{\sigma}\right) \quad (4.1)$$

where σ defines the width of the bell curve that surrounds each sample point. Proper choice of the value for σ is critical to the performance of the PNN. If σ is too small, individual training cases will be considered only in isolation and we will be left with essentially a nearest-neighbour classifier [Mast93]. However, if the value of σ is too high, details of the density will be blurred together and, unless the different classes are very well

separated, confusion would certainly result. Selecting a suitable value for σ will be discussed in Section 4.3.

The weight function, $W(x)$, is almost always chosen to be the normalized Gaussian function as

$$W(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{x^2}{2\sigma^2}\right)} \quad (4.2)$$

While other weight functions are theoretically possible, the Gaussian is well-behaved and ideally shaped for this implementation. It has been shown in practice to perform well and is almost always the function of choice for practical applications.

Referring back to Fig. 4.2, we see that the PNN has p inputs. Therefore the PDF estimator must take into account all of these input variables in order to achieve an accurate density estimation. In 1966, a means to extend Parzen's method to the multivariate case was introduced [Caco66]. However, it is somewhat more complicated because it allows each of the input values to have its own scale factor, σ . This fully general density estimator for a class, c , is given by

$$g_c(x_1, \dots, x_p) = \frac{1}{n_c \sigma_1 \sigma_2 \dots \sigma_p} \cdot \sum_{i=1}^n W\left(\frac{x_1 - x_{1,i}}{\sigma_1}, \dots, \frac{x_p - x_{p,i}}{\sigma_p}\right) \quad (4.3)$$

where, as in Eq 4.1, n_c is the number of samples which exist for the class. To reduce the complexity of Eq 4.3, we make a few common simplifications. First, for the basic PNN model, we will assume that all scale factors are equal such that $\sigma_1 = \sigma_2 = \dots = \sigma$. Then we

can achieve further economy by letting the multivariate weight function become the product of the univariate weight functions as shown in

$$W(x_1, \dots, x_p) = \prod_{i=1}^{n_c} W_i(x_i) \quad (4.4)$$

Now, with these two simplifications, the density estimator can be stated explicitly as

$$g_c(\mathbf{X}) = \frac{1}{(2\pi)^{p/2} \sigma^p n_c} \cdot \sum_{i=1}^{n_c} e^{-\frac{\|\mathbf{X}-\mathbf{X}_i\|^2}{2\sigma^2}} \quad (4.5)$$

where \mathbf{X} represents the input vector x_1, \dots, x_p . Notice that the distance function, $\|\mathbf{X}-\mathbf{X}_i\|^2$, is simply the standard Euclidean distance for the vector, \mathbf{X} . The density estimator given in Eq. 4.5 is the foundation of the original PNN proposed by Specht [Spec88], [Spec90a]. It is also the estimator used in most PNN implementations and has a history of good performance [Mast95].

4.2.3 PNN Processing

Examine once again the architecture of the PNN in Fig. 4.2. To classify the unknown case, \mathbf{X} , at the input layer, the network propagates a signal forward as shown by the arrows on the neuron connections. The input layer, as stated previously, holds the unknown sample for distribution to each node in the pattern layer. To explain the exact calculation which takes place in the pattern neuron we shall again consider Eq. 4.2. Since we are classifying using relative class-conditional PDFs among different classes, we can

eliminate any constants which exist in the functions. Thus, we are left with some constant multiple of the densities for all classes, which when considered in relative terms, would yield identical results to those obtained by using true densities. The weight function given at Eq. 4.2 can be replaced by the unnormalized Gaussian in

$$W(\mathbf{X}, \mathbf{X}_r) = e^{-d(\mathbf{X}, \mathbf{X}_r)^2} \quad (4.6)$$

where $d(\mathbf{X}, \mathbf{X}_r)^2$ is a σ -scaled, squared Euclidean distance computed between the unknown sample, \mathbf{X} , and the training case, \mathbf{X}_r , as calculated by

$$d(\mathbf{X}, \mathbf{X}_r)^2 = \frac{1}{\sigma^2} \cdot \sum_{i=1}^p |x_i - x_{r,i}|^2 \quad (4.7)$$

Note that the constant 2 in the exponential of Eq. 4.2 has been absorbed into the value of σ as a further means to simplify the calculation.

Computations performed by the pattern neurons can now be specifically explained. Each pattern neuron computes the σ -scaled Euclidean distance, given by Eq. 4.7, between the unknown and the training case which it represents. Then the weight function is applied to this distance as described by Eq 4.6. These weighted distance measurements are then propagated forward, as shown by Fig 4.2, to the summation neuron which corresponds to the class of the training case represented by each pattern neuron.

The summation neurons, just as their name implies, perform a simple summation of the weighted distance measurements for the class they represent. This computation is similar to Eq 4.5. However, due to the simplifications derived in Eqs. 4.6 and 4.7, a

significantly reduced classification function, which is merely a multiple of the PDF described by Eq. 4.5, can be used as in

$$g_c(\mathbf{X}) = \frac{1}{n_c} \cdot \sum_{r=1}^{n_c} e^{-d(\mathbf{X}, \mathbf{X}_r)^2} \quad (4.8)$$

It should be noted that the factor, $1/n_c$, in Eq 4.8 is not constant unless there is an equal number of training samples for each class, and thus, could not be eliminated in the general case. Specifically, this factor compensates for unequal representation of different classes which may exist in the training set. Otherwise, the network would be significantly biased toward choosing the class with the most cases in the training set. In some cases though, this bias may be desirable if the number of samples from each class in the training set is carefully selected to be proportional to their probability of being encountered. The idea of incorporating prior probabilities in a manner similar to this, will be expanded upon in Section 4.5.

The output layer of the PNN is usually a simple threshold discriminator. All of the outputs from the neurons in the summation layer are examined and the largest one is selected. Then, only the output layer neuron which corresponds to that neuron in the summation layer is activated. The index of the activated neuron in the output layer represents the predicted class of the sample at the input layer. Section 4.5 will discuss some possible extensions to the computations performed by the output layer.

Examination of Eq. 4.8 reveals a potential problem which may be encountered. Taking the exponent of a negative number will quickly yield a value of zero as the magnitude of the number increases. If the unknown case, \mathbf{X} , and the training case, \mathbf{X}_r , are

extremely different, the summation obtained for that class may be zero. While this may appear to be a desirable effect, it would become impossible to select the right class if the unknown is substantially different from all training cases and all of the summations become zero. A simple solution to this problem is to flag its occurrence during classification and yield a result which shows that this case belongs to none of the known classes. In practice, this problem will be a rare occurrence, but an example will be shown in Subsection 5.2.2 where it is encountered. A potentially positive consequence of this situation is that it provides a simple *reject mechanism* when a wildly different and probably unknown sample is encountered.

4.3 Training the PNN

Like the MLFN, the PNN utilizes a supervised learning scheme. This means that during the learning process, selected training cases are presented to the neural network along with the correct output. The network then adapts itself to produce the correct output when similar cases are seen. However, unlike the MLFN, the PNN does not need to be trained extensively to produce good results. With most problems, the PNN will produce sufficient results with the optimization of just one scaling parameter, σ , in the Parzen density estimation. This task can be performed quickly, using one of several standard univariate optimization methods.

4.3.1 An Error Function for the PNN

For training a neural network, a means to measure its performance, as a function of σ , is required so that we know how well the network is performing and when to cease training. Since the PNN is a classifier, an intuitive measure of error would be to simply iterate through the entire training set, classifying each known training case as if it were

unknown. The number of correct classifications could be counted and this would become the measure of error for the network. However, examination of Eq. 4.6 and Eq. 4.7 shows that this would not be an effective error measure. The problem occurs when the training case at the input is compared with itself at its corresponding pattern layer neuron. The Euclidean distance would be zero, and therefore, the weight function would reach full activation for every input in the training case, regardless of the value of σ . Such a tremendous bias would certainly cause the network to correctly classify every case in the training set, rendering this error measure useless.

A similar, yet effective, alternative to the above technique is called the holdout method [Spec90b]. It involves the same iteration through the training set as described above, except for one small deviation. When a training case is presented to the network for classification, the corresponding neuron in the pattern layer is temporarily removed and the network classifies using $R - 1$ pattern layer neurons. Using this technique, no bias is introduced as a result of comparing a training case to itself. This method is especially useful in the common situation where it is desired to make maximum use of a limited data set. In practice, it has been shown to produce very good results. However, it should be noted that the results achieved in training using this technique will still be partially biased by the training set because the classification of the training case being left out is still involved in the choice of σ . Typically, this would cause the training error to be slightly better than that obtained using completely unknown samples. Whenever possible, efforts should be made to leave a significant number of samples aside during training for a true validation of the network.

4.3.2 Optimizing Sigma

Now that we have a means to measure the performance of the PNN for a given value of σ , a method for finding the optimal value of σ can be established. Since we are optimizing a function with one variable, any of several different univariate optimization algorithms could be employed. However, there are some considerations which should be made before selecting a technique. First, since we are iterating through the entire training set for each evaluation of σ , the time required can quickly become large as it is proportional to the square of the number of training cases. Thus, we must do as few evaluations as possible for each value of σ . Also, due consideration should be given to the possibility that multiple minima may exist. It would be somewhat careless to accept the first local minimum found because better results may be achieved at a different value.

The optimization technique used in this thesis is performed using two distinct steps [Mast95]. First, a trivial global search is conducted at N_s points over a range from a to b . The error measure is determined at the lower bound of the selected range, a , and then at logarithmic intervals until the upper bound, b , is reached. Points along the search range are incremented by a multiple, m , as calculated in Eq. 4.9, such that the ratio of adjacent points is equal

$$m = e^{\frac{\log(b/a)}{N_s - 1}} \quad (4.9)$$

This logarithmic spacing, as opposed to linear spacing, is used because the effect of σ in the error function is multiplicative rather than additive. Now that the interval has been evaluated at N_s points, the point with the lowest error is selected, along with the adjacent

points on both sides, for use in computations during the next step. The only potential problem with this global search occurs when the minimum exists at one of the endpoints of the interval. In this situation, the routine continues stepping out in that direction until the error function turns up. Despite this safety mechanism, though, care should be taken to try to avoid this situation by setting the endpoints appropriately. Values on the order of 0.001 up to 100 may be required, depending on the size and complexity of the problem. Also, in order to ensure that a global minimum is actually found, the number of search points, N_s , should be set as high as possible while remaining within the tolerance of the user in terms of processing time. At the termination of this algorithm, a bounded minimum, γ , will be found. The values of a and b again bound the interval of the minimum, except that they have become the calculated values which were adjacent to γ in the initial search.

After a rough bounded minimum has been found using the rudimentary minimization algorithm described above, the next step is to refine the value of σ using the golden section technique. In this minimization scheme, three points are kept track of such that the error measure at the centre point is less than that of either of its neighbours. Again, we will refer to these points as a , b , and γ for the two endpoints and the bounded minimum respectively.

The golden section technique considers the bracketing triplet of points and evaluates the measure of error at an intermediate point between either a and γ or between γ and b . Suppose a point, z , is selected between a and γ , as shown in Fig 4.4, and the measure of error is evaluated there.

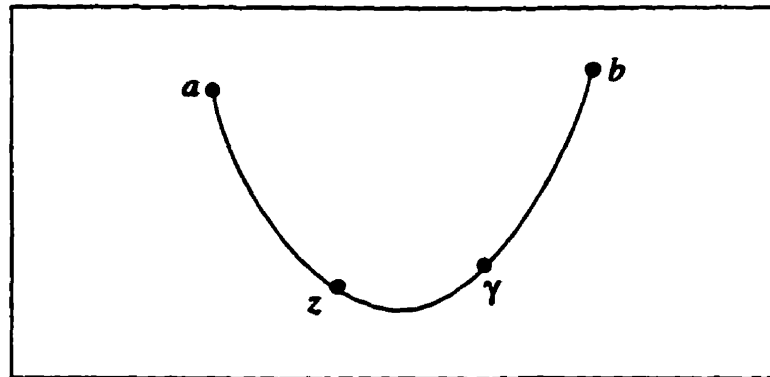


Fig. 4.4. Bracketing triplet and test point in golden section minimization.

If the measure of error at z is less than the error measure at γ , then z becomes the new middle point and γ becomes the new endpoint (i.e. $b = \gamma$ and $\gamma = z$). Otherwise, z becomes the new endpoint on the left side (i.e. $a = z$). In either case, one of the endpoints is dropped and the middle point in the new triplet is the best minimum achieved so far. Then, the process is repeated until the difference in the error achieved for successive iterations becomes insignificant, or until the width of the interval becomes small enough.

The issue of placing the test point, z , must be addressed. It is reasonable to select the test point such that it is in the wider of the two intervals because, given no other information, this is the area where the minimum likely exists. It can be shown that, within the wider of the two intervals, an optimum placement rule exists [PTFV92]. That is, given a bracketing triplet of points, the next point to be tried is a fraction 0.38197 into the larger of the two intervals from the current minimum. This value is determined from

$$w^2 - 3w + 1 = 0 \rightarrow w = \frac{3 - \sqrt{5}}{2} = 0.38197 \quad (4.10)$$

where w is called the *golden mean* or the *golden section*, hence the name of this minimization procedure.

4.4 Improving the Basic PNN

Examine the modelled transmitter transients shown at Fig. 3.3 and Fig 4.5. A common characteristic between these models is that the variance dimension trajectory is higher at the start of the transient and then tapers off. This feature exists in many of the transient models because the initial response of the system is somewhat noisy before smoothing off near the end of the transient. Additionally, since consistent segmentation of the transient from the ambient channel noise is sometimes not achieved, the beginning of different transients may in fact consist of some unknown quantity of irrelevant noise. Given this relatively consistent feature among the transient models' values, it follows that the classification algorithm may benefit by allowing different scaling factors for the Parzen PDF estimators at each of the inputs. Existence of variable scale factors for different inputs enables a relative measure of importance, in terms of affecting classification ability, to be assigned to different input variables. This could assist in dealing with the somewhat ambiguous duration of noise possible at the beginning of the segmented transients. The overall contribution of insignificant inputs could be weighted such that their influence in the classification would be minimized. This section gives details on implementing the concept just described in order to achieve a very powerful extension to the basic PNN [Mast95].

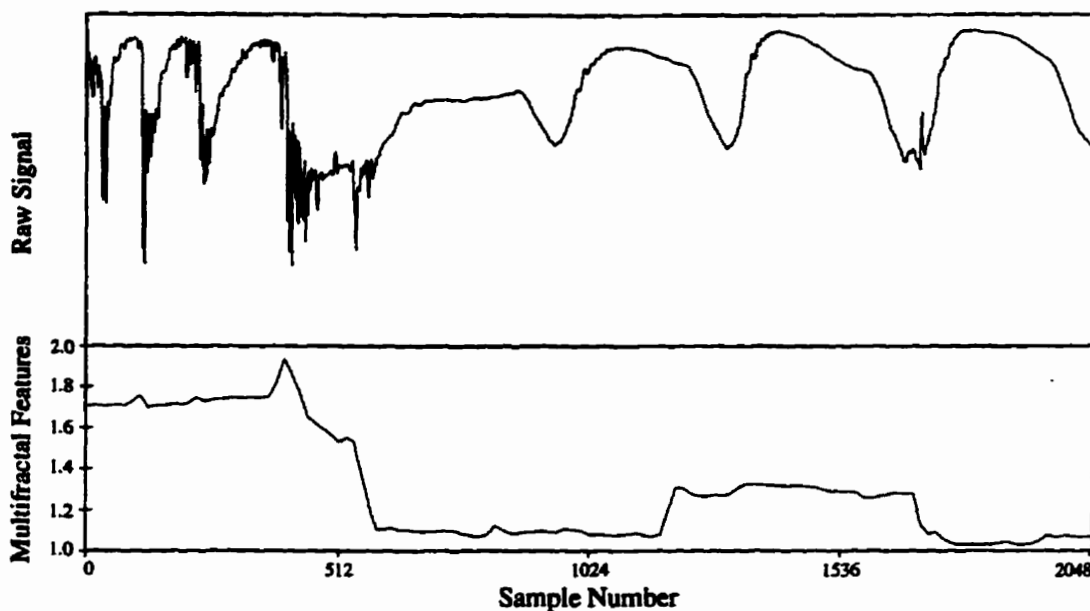


Fig. 4.5. Fractal model of transient.

4.4.1 The Multiple-Sigma PDF Estimator

Now that there exists a clear motivation to allow individual scaling parameters for each of the P input variables in the PNN, a different weighted and squared Euclidean distance must be used as in

$$\delta(X, X_r)^2 = \sum_{i=1}^P \left(\frac{x_i - x_{r,i}}{\sigma_i} \right)^2 \quad (4.11)$$

where r is the index of the current training case. Note that there is a minor computational expense, relative to Eq. 4.7, in that a division is required for each iteration of the summation rather than a single division after the summation.

The classification function can now be updated for the multiple- σ PNN. If we assume that the Gaussian weight function is retained, the multiple of the PDF described in

Eq. 4.8 remains identical, except that the new distance measure, $\delta(X, X_r)$, is included as shown in Eq. 4.12.

$$g_c(\mathbf{X}) = \frac{1}{n_c} \cdot \sum_{r=1}^{n_c} e^{-\delta(\mathbf{X}, \mathbf{X}_r)^2} \quad (4.12)$$

With the introduction of multiple values for σ in the PNN, a different method for finding their optimal values is also required because the simple univariate optimization schemes presented in the previous section are not sufficient. Instead, a multivariate optimization scheme is necessary. The conjugate gradients algorithm, to be explained in Subsection 4.4.4, is recommended [Mast95] and used in this thesis. A significant problem exists in implementing such an optimization technique in that derivatives of the error function, with respect to the scaling parameters, are required. Unfortunately, with the simple counting criterion used as an error function in the basic PNN, meaningful derivative calculations are impossible.

4.4.2 A Continuous Error Function for the PNN

To allow for derivative calculations, and to break possible ties in the results achieved from the discrete error criterion, a continuous function of the σ weights is required as a measure of network error. To begin this discussion, consider a vector, \mathbf{Y} , such that it represents the output vector of the PNN. Thus, the vector will have C elements such that $\mathbf{Y} = (y_1, \dots, y_C)$. After a classification by the PNN, \mathbf{Y} will be filled with 0s except for a single 1 in the position representing the predicted class of the case at the input of the network. For instance, if there is a network intended to classify from five different classes,

and the case at the input is a member of the second class, the correct output of the network is $Y = (0, 1, 0, 0, 0)$.

The PNN can now be considered as a mapping function between the input vector, X , and the output vector, Y . The implications of this are significant in that it shows the potential for the PNN to act as more than just a classifier; perhaps even as a general mapping function like the MLFN [Spec91], [Mast95]. However, since the PNN implementation in this thesis is purely for classification, this discussion will focus on how the notion that the PNN provides a mapping between X and Y leads to an interesting validation of the PNN [SCHA91]. Details of this validation can then be used to derive a continuous error function.

In statistical terms, we can say that X is an independent vector and that each element of Y is dependent upon X . If the joint PDF of X and each element of Y , $f_{Xy}(X, y_c)$, is known, then the conditional expectation of Y for a given X , $E_{Y|X}(X)$, can be obtained from

$$E_{Y|X}(X) = \left(\frac{\int_{-\infty}^{\infty} y_1 \cdot f_{Xy}(X, y_1) dy_1}{\int_{-\infty}^{\infty} f_{Xy}(X, y_1) dy_1}, \dots, \frac{\int_{-\infty}^{\infty} y_c \cdot f_{Xy}(X, y_c) dy_c}{\int_{-\infty}^{\infty} f_{Xy}(X, y_c) dy_c} \right) \quad (4.13)$$

where the ratio of integrals is applied separately for each element of Y . The joint PDF of X and y_c can be estimated by appending y_c to the end of the X vector and using the multivariate Parzen estimator of Eq. 4.12. Now, the estimated joint PDF, $g_{Xy}(X, y_c)$, can be substituted directly into Eq. 4.13, giving the following expansion:

$$E_{Y|X}(X) = \left(\frac{\int_{-\infty}^{\infty} y_1 \cdot g_{Xy}(X, y_1) dy_1}{\int_{-\infty}^{\infty} g_{Xy}(X, y_1) dy_1}, \dots, \frac{\int_{-\infty}^{\infty} y_C \cdot g_{Xy}(X, y_C) dy_C}{\int_{-\infty}^{\infty} g_{Xy}(X, y_C) dy_C} \right) \quad (4.14)$$

Solving the above equation by numerical means would be impractical. Fortunately, since the Gaussian weighting function was used for finding $g_{Xy}(X, y_c)$, Eq. 4.14 can be simplified immensely to obtain

$$E_{Y|X}(X) = \left(\frac{\sum_{r=1}^R y_{r,1} \cdot e^{-\delta(X, X_r)^2}}{\sum_{r=1}^R e^{-\delta(X, X_r)^2}}, \dots, \frac{\sum_{r=1}^R y_{r,C} \cdot e^{-\delta(X, X_r)^2}}{\sum_{r=1}^R e^{-\delta(X, X_r)^2}} \right) \quad (4.15)$$

where R is the number of training samples for the entire training set. Notice that each element of Y is simply a scaled density estimator which is similar to Eq. 4.12, except that it is not normalized for the number of training cases in each class. However, the class count normalization factor can be directly added to each element of Y without affecting this derivation.

During training, since Y is known, it can be used to calculate the error function for the PNN directly. An activation function for each of the C summation neurons will now be defined as

$$u_c(X) = \frac{1}{n_c} \cdot \sum_{r=1}^R y_{r,c} \cdot e^{-\delta(X, X_r)^2} \quad (4.16)$$

which is the same as Eq. 4.12 because the y_{rc} term allows only the training cases from class c to contribute to the summation. An internal confidence measure, $q_c(\mathbf{X})$, of \mathbf{X} belonging to class, c , can be defined as in

$$q_c(\mathbf{X}) = \frac{u_c(\mathbf{X})}{s(\mathbf{X})} \quad (4.17)$$

where

$$s(\mathbf{X}) = \sum_{c=1}^C u_c(\mathbf{X}) \quad (4.18)$$

which, except for the $1/n_c$ factors, is identical to the denominator in Eq. 4.15. Finally, we can state the continuous error function, $e(\mathbf{X})$, for a given training sample, \mathbf{X} , as follows:

$$e(\mathbf{X}) = [1 - q_c(\mathbf{X})]^2 + \sum_{i=1}^C [q_{i, i \neq c}(\mathbf{X})]^2 \quad (4.19)$$

where c represents the correct class for the training sample. This error function agrees with intuition because if $q_c(\mathbf{X}) = 1$ and $q_i(\mathbf{X}) = 0$ for all $i \neq c$, then $e(\mathbf{X})$ will become 0. However, when the outcome is not perfect, all available information is considered, including the level of activation for classes other than the correct one, c . To find the average error for a single pass through the entire training set, simply iterate through the training set, cumulating a sum of $e(\mathbf{X})$ for each case, and then divide the sum by the number of cases in the training set, R .

4.4.3 Derivatives of the Continuous Error Function

Now that there is a continuous error function for the PNN, we can calculate its first and second derivatives with respect to the scaling parameters. These derivatives will be used for optimization of the scaling parameters. Since the objective of training the PNN is to minimize the average of $e(\mathbf{X})$ for all training cases, and the values of σ_p are constant for all training cases, derivative calculations of $e(\mathbf{X})$ with respect to σ_p should be calculated separately for each case and then averaged. Thus, differentiating Eq. 4.19, with respect to σ_p , gives

$$\frac{\partial e(\mathbf{X})}{\partial \sigma_p} = 2[q_c(\mathbf{X}) - 1] \left[\frac{\partial q_c(\mathbf{X})}{\partial \sigma_p} \right] + 2 \sum_{i=1}^C \left[q_{i, i \neq c}(\mathbf{X}) \frac{\partial q_{i, i \neq c}(\mathbf{X})}{\partial \sigma_p} \right] \quad (4.20)$$

for a single input vector, \mathbf{X} . The second derivative is then obtained by differentiating Eq. 4.20 to yield

$$\begin{aligned} \frac{\partial^2 e(\mathbf{X})}{\partial \sigma_p^2} &= 2[q_c(\mathbf{X}) - 1] \left[\frac{\partial^2 q_c(\mathbf{X})}{\partial \sigma_p^2} \right] + 2 \left[\frac{\partial q_c(\mathbf{X})}{\partial \sigma_p} \right]^2 \\ &+ 2 \sum_{i=1}^C \left[q_{i, i \neq c}(\mathbf{X}) \frac{\partial^2 q_{i, i \neq c}(\mathbf{X})}{\partial \sigma_p^2} \right] + 2 \sum_{i=1}^C \left[\frac{\partial q_{i, i \neq c}(\mathbf{X})}{\partial \sigma_p} \right]^2 \end{aligned} \quad (4.21)$$

To finish solving the derivatives shown above, we require the first and second partial derivatives of the internal confidence measures, $q_c(\mathbf{X})$, with respect to the scaling parameters. However before giving these derivatives, some intermediate definitions will be required to make the equations more manageable. These are given as

$$\frac{\partial u_c(\mathbf{X})}{\partial \sigma_p} \equiv v_{c,p}(\mathbf{X}) = \frac{2}{n_c} \sum_{r=1}^R y_{r,c} \cdot e^{-\delta(\mathbf{X}, \mathbf{X}_r)^2} \cdot \frac{(x_p - x_{r,p})^2}{\sigma_p^3} \quad (4.22)$$

$$\begin{aligned} \frac{\partial^2 u_c(\mathbf{X})}{\partial \sigma_p^2} &\equiv w_{c,p}(\mathbf{X}) \\ &= \frac{2}{n_c} \sum_{r=1}^R y_{r,c} \cdot e^{-\delta(\mathbf{X}, \mathbf{X}_r)^2} \left[2 \frac{(x_p - x_{r,p})^4}{\sigma_p^6} - 3 \frac{(x_p - x_{r,p})^2}{\sigma_p^4} \right] \end{aligned} \quad (4.23)$$

$$\frac{\partial s(\mathbf{X})}{\partial \sigma_p} \equiv V_p(\mathbf{X}) = \sum_{c=1}^C v_{c,p}(\mathbf{X}) \quad (4.24)$$

$$\frac{\partial^2 s(\mathbf{X})}{\partial \sigma_p^2} \equiv W_p(\mathbf{X}) = \sum_{c=1}^C w_{c,p}(\mathbf{X}) \quad (4.25)$$

Now, the first and second partial derivatives of the internal confidence measures, $q_c(\mathbf{X})$, with respect to the scaling parameters can be given as

$$\frac{\partial q_c(\mathbf{X})}{\partial \sigma_p} = \frac{v_{c,p}(\mathbf{X}) - V_p(\mathbf{X}) q_c(\mathbf{X})}{s(\mathbf{X})} \quad (4.26)$$

$$\frac{\partial^2 q_c(\mathbf{X})}{\partial \sigma_p^2} = \frac{w_{c,p}(\mathbf{X}) - W_p(\mathbf{X}) q_c(\mathbf{X})}{s(\mathbf{X})} + \frac{2V_p^2(\mathbf{X}) q_c(\mathbf{X}) - 2v_{c,p}(\mathbf{X}) V_p(\mathbf{X})}{s^2(\mathbf{X})} \quad (4.27)$$

The results shown in Eq. 4.26 and Eq. 4.27 can be directly substituted into Eq. 4.20 and 4.21 as required to complete the definitions of the first and second derivatives of the continuous error function.

4.4.4 *Training the Multiple-Sigma PNN Using Conjugate Gradients*

Details on initializing and training the scaling parameters in the improved PNN will now be discussed. However, since there are some significant differences between training the multiple- σ model compared with the basic PNN, an overview of the whole process is necessary. In training the improved PNN, there will be two distinct steps. Common to both steps is that we will continue to use the holdout method whereby the pattern layer neuron corresponding to each training case, X , is temporarily removed and the network is trained using $R - 1$ pattern layer neurons. To commence training, the scaling parameters are selected as if the network were the basic PNN model. This means that all scaling parameters will be initialized to the same value using the fast univariate minimization technique. This provides a good starting point for the more advanced multivariate minimization procedure and avoids the requirement for this procedure to conduct a global search at its relatively slow rate.

In the second training step, the network iterates through the training set as before. However, as well as calculating the error function, $e(X)$, for each training case, X , it also calculates the first and second the derivatives of the error function with respect to each of the P scaling parameters, σ_p . Then, after calculating each of these values, they are added to running totals to obtain an error function and derivatives cumulated for the entire training set. These cumulated values are then divided by the number of training cases to yield the average error and the average derivatives for the entire training set. The multivariate optimization method, conjugate gradients, then uses the derivative information, and average error measures, to find new values for all σ_p . The iteration is then

repeated, calculating new average derivatives and errors, and updating scaling factors until a sufficient accuracy has been achieved or successive improvements become negligible.

The conjugate gradient algorithm used for this training is a very powerful and popular multivariate optimization technique. In fact, in addition to its utility in the PNN, it is known to be one of the best methods for training a MLFN [Mast93]. A full explanation of conjugate gradients would require a chapter in itself and many excellent references can provide a sufficiently detailed description [Pola71], [PTVF92], [Mast95]. Instead we will discuss the relevant features of the conjugate gradient algorithm and show why this method is appropriate for use in PNNs.

The most significant feature of the conjugate gradient method is its convergence speed in comparison to other popular techniques. For instance, it is significantly faster than the popular steepest descent method of optimization which is often used for training the MLFN. Also, it eliminates the calculation, storage, and manipulation of a large Hessian matrix as would be required in Newton or quasi-Newton optimization techniques. This advantage is especially prevalent in neural network implementations which, since there are usually many variables to be optimized, would require massive Hessian matrices. Another advantage is that the conjugate gradients method does not require second derivative information, which may be impossible or inefficient to calculate in some applications. However, as shown previously, second derivative information is readily available and can be efficiently computed for the PNN. A slight modification to the conjugate gradients algorithm incorporates these second derivatives to estimate an efficient scaling factor for the line search and significantly speeds convergence [Mast95]. Specifically, the implementation in this thesis uses the Polak-Ribiere [PTVF92] conjugate

gradients algorithm with Masters' modification to take advantage of the readily available second derivatives.

4.4.5 Other Extensions to the PNN Classifier

Several other useful extensions exist for the PNN in its role as a classifier and will be mentioned here for completeness even though they are not used in this thesis. The most logical extension to the multiple- σ PNN is to allow separate scaling parameters for each of the different classes as well as for each of the input values. Such a model would enable specific input variables to have more or less importance for some classes but not necessarily for others. The utility of this is obvious in many different classification problems, perhaps even this one. However, such power is not without cost, especially in terms of time required to train the scaling parameters. For instance, if we had a training set with 100 inputs and 100 different classes, there would be a total of 10,000 scaling parameters to train. Intuition suggests that this would require 100 times the training time than without this specific modification. In practice though, because of interactions between the increased number of parameters, this modification would result in a training time which is several hundred times greater. Additionally, with this much representational power, there exists a strong possibility of overfitting the data and hindering the network's ability to generalize in the presence of slightly different input vectors.

Another modification to the basic PNN model is the Gram-Charlier Neural Network (GCNN) [KiAr92], [Mast95]. In the GCNN, a Gram-Charlier series is used instead of Parzen windows for approximating the class-conditional PDFs. This results in a network which requires negligible storage, can perform fast classification, and trains instantaneously from explicit formulas. However, there are drawbacks to this method

which limit its general applicability in comparison to the PNN. Specifically, this model cannot handle the diverse range of distributions that the PNN can. Also, multimodal distributions and distributions that are not reasonably close to normal will not be handled well by this network. Obviously, this precludes its use in this thesis, but the GCNN should still be considered as a powerful extension to the PNN when the distribution of the data is suitable.

4.5 Bayes Classification and Confidence Levels

Bayes' method of classification is a widely accepted standard for implementing decision rules [Spec90a]. This section explains the Bayes classification strategy and shows how the PNN can be structured to approach the Bayes optimal decision surface. Finally, we give an explanation of how and when Bayesian confidence estimates can be computed for the PNN.

4.5.1 Bayes' Strategy for Classification

The objective of Bayes' method is to minimize the expected risk of misclassification for a given decision surface. Assume that we have a collection of samples from C different classes indexed as $c = 1, \dots, C$. Each of these samples is a vector $X = (x_1, \dots, x_p)$. We will now define a value, h_c , for a class which defines its probability of occurring in the data set. This value will be referred to as the prior probability of a class, c . According to what is generally known as Bayes' Postulate, prior probabilities should be assumed equal when nothing is known to the contrary [Mast93]. Another value, l_c , will now be defined as the loss associated with misclassifying a case which belongs to class, c .

In practice, these values are also set to be equal unless there is compelling reason to do otherwise. Finally, we will define $f_c(\mathbf{X})$ to be the true PDF for a class, c .

With the definitions given above we can state the Bayes decision rule. An unknown sample, \mathbf{X} , is classified into class c if

$$h_c \cdot l_c \cdot f_c(\mathbf{X}) > h_i \cdot l_i \cdot f_i(\mathbf{X}) \quad (4.28)$$

for all classes i not equal to c . Any algorithm that applies the above rule to a decision surface is said to be Bayes optimal. In practice, it is difficult to build a classifier which completely satisfies this rule because the actual PDF, $f_c(\mathbf{X})$, is usually unknown. However, using Parzen's method of density estimation, we can find a reasonable estimated PDF, $g_c(\mathbf{X})$, when there is a comprehensive training set.

An intuitive discussion on the Bayes decision rule will be given as proof of Eq. 4.28. Since the PDF, $f_c(\mathbf{X})$, is proportional to the concentration of the members of class c around the unknown case, then the $f_c(\mathbf{X})$ with the highest value may well represent the correct class. However, since a mechanism exists to consider prior probabilities of encountering a case from a certain class, then a class with a higher prior probability should be favoured. Finally, we want to minimize the loss associated with misclassifying a member of class c . Therefore, if class c has a high loss associated with misclassification of one of its members, then it should be favoured in an attempt to avoid this loss. As long as the values used for these three criteria are scaled properly, simple multiplication of them should yield a sufficient balance from which to base a decision upon.

4.5.2 *Implementing Bayes' Method in the PNN*

By virtue of the design of the PNN, the PDFs for each class, $f_c(X)$, or at least estimates of them, are already taken into consideration during classification. All that remains is to add a mechanism to deal with prior probabilities, h_c , and the projected loss, l_c , associated with misclassifying a case that belongs to a class, c . In Sub-Section 4.2.3 we eluded to a method of incorporating prior probabilities into the PNN by carefully selecting the number of training cases from each class to be proportional to their probability of being encountered. This is an ideal method in cases where the structure of the available training set may be the only means available for determining prior probabilities. However, the formulas for the PNN model used in this thesis have included a factor, $1/n_c$, which is designed to remove the effect of unequal class representation from the training set. This is done to protect the user from accidentally introducing a bias due to inadequate representation of the data set, as so often is the case. Instead, if there is sufficient reason to introduce prior probabilities into the PNN decision, there is a more elegant and deliberate methodology. That would be to change Eq. 4.8 so that it is multiplied by h_c as in

$$g_c(X) = \frac{h_c}{n_c} \cdot \sum_{r=1}^{n_c} e^{-d(X, X_r)^2} \quad (4.29)$$

for all of the pattern layer neurons. Note that, in the multiple- σ PNN, the same change would be applied to Eq. 4.12 and Eq. 4.16.

The incorporation of projected losses associated with misclassification of a certain class is done exactly the same way as for the prior probabilities. Modifying Eq. 4.8 again, we could multiply by l_c as in

$$g_c(\mathbf{X}) = \frac{h_c \cdot l_c}{n_c} \cdot \sum_{r=1}^{n_c} e^{-d(\mathbf{X}, \mathbf{X}_r)^2} \quad (4.30)$$

and the same change could be applied to Eq. 4.12 and 4.16 for the multiple- σ PNN model.

4.5.3 Bayesian Confidence Measures for the PNN

Part of the appeal to the PNN is that mathematically sound confidence estimates can often be computed for its decisions [Mast95]. However, there are two conditions which must be met in order to achieve accurate confidence estimates. The first condition is that the classes in the possible data set must be mutually exclusive. This means that no case can possibly fall into more than one class. The other condition is that the training set must be exhaustive in that it represents all possible classes fairly and completely. This means that the PDF estimates, $g_c(\mathbf{X})$, must be very close to the true PDFs. Now, assuming that the training set is mutually exclusive and exhaustive, Bayes' theorem can be used to directly compute the probability that an unknown sample, \mathbf{X} , belongs to class c as in

$$Prob[c|\mathbf{X}] = \frac{g_c(\mathbf{X})}{\sum_{i=1}^C g_i(\mathbf{X})} \quad (4.31)$$

where $g_i(\mathbf{X})$ represents the output activation of each of C summation neurons representing the possible classes. Unfortunately, due to possible inconsistencies in segmentation of

transients, the data used in this thesis does not fit the first criterion. Then, assuming the segmentation problem can be resolved, it would be somewhat difficult to satisfy the second criterion outside of a laboratory. Chapter 6 will discuss more generic techniques for assessing the confidence level of the decisions.

4.6 Summary of Chapter 4

This chapter has stated that, for classification, the PNN is a more accurate and practical neural network model than the MLFN. Details of its structure, operation, and training were given. A powerful extension to the basic PNN was then discussed along with detailed derivations for an effective training algorithm. The chapter ended with a general discussion on Bayes' Classification paradigm. It showed how the PNN can approach the Bayes optimal decision surface and, under certain conditions, provide mathematically sound confidence levels.

Exact parameter settings and results achieved for PNN classification of multifractal transient models will be discussed in Chapter 6. A detailed description of TAC-MM, the software package developed to implement the transient classification system, will now follow.

CHAPTER V

SOFTWARE IMPLEMENTATION AND VERIFICATION

The processing techniques described up to this point for the three modules of the transient classification system have been assembled into a comprehensive and flexible software package called TAC-MM (Transient Analyser and Classifier using Multifractal Modelling). This software is a 32-bit, single document interface (SDI) Windows 95 application. It is written in the C++ language, with Microsoft Foundation Class (MFC) extensions, and compiled using Microsoft Visual C++ Version 5.0. The source code files are included in Appendix B to this thesis. To facilitate ease of use, an intuitive user-interface has been developed such that it conforms closely with standards for a Windows 95 program as prescribed by Microsoft.

This chapter describes the various features and limitations of the TAC-MM software and explains how to implement it for analysis and classification of transients. The discussion is structured to reflect the organization of the program menus rather than from a procedural perspective. Section 5.1 provides specific information about every menu function in the TAC-MM program. This is, however, preceded by a general introduction to the user interface and the transient display area. Then, in order to ensure the validity of this research, we must prove that the software operates according to its design and that numerical processing techniques are coded properly. Section 5.2 details the procedures used to verify the preprocessing, feature extraction, and classification modules of this system.

5.1 Using the TAC-MM Software Package

5.1.1 The User Display Area

Since TAC-MM utilizes a graphical user interface (GUI) for its implementation, a discussion on its operation should begin with a description of the user display area. Figure 5.1 shows the main viewing area of the program along with the various command interfaces for selecting program functions. Starting at the top of the viewing window, the menus are arranged in the standard fashion with push-button style controls for the most common user functions. The functions that have push-button controls are indicated with a picture of the button beside its description in this section. The different menus are categorized as File, Edit, View, Neural Net, and Help.

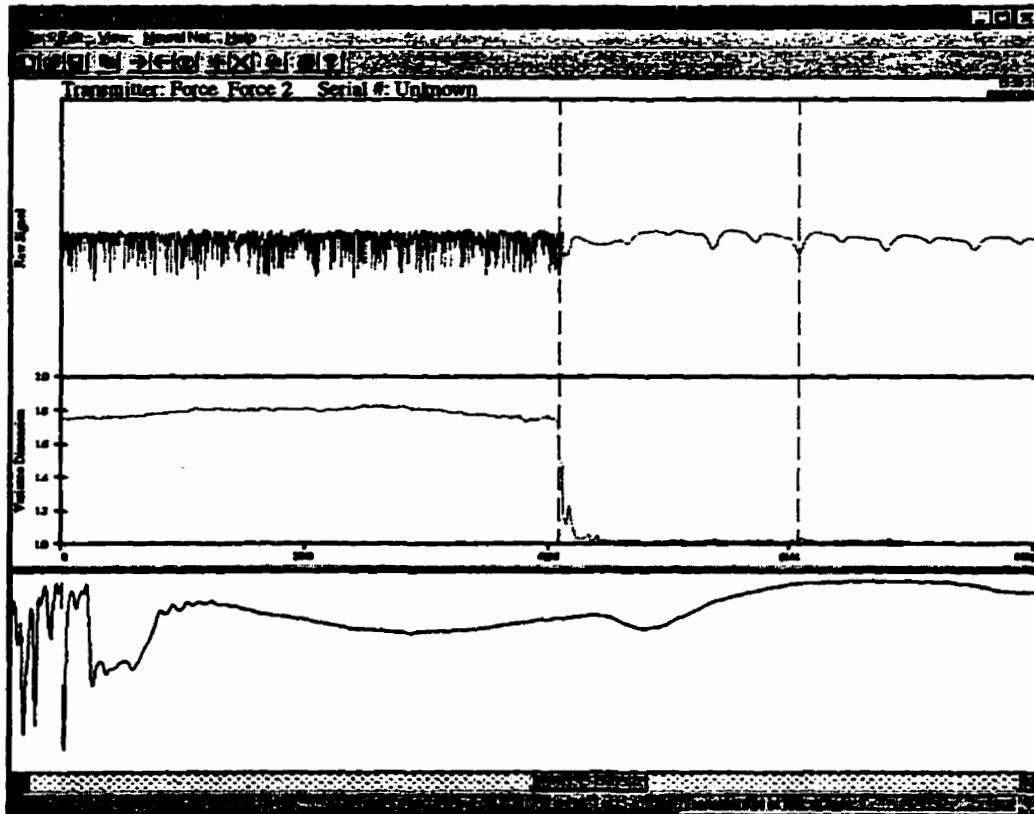


Fig. 5.1. TAC-MM user display area; default view options.

The transient viewing area has been divided into two windows as shown. In the upper viewing window, a raw signal is displayed along with the variance fractal dimension trajectory of that signal. It has two different display modes. In the first mode, the user can view the segmentation of the raw signal along with the variance dimension trajectory waveform. The transient, as detected by the preprocessing module of the system, is the part of the signal between the vertical dashed lines. In the second viewing mode, only the part of the signal selected as being the transient is displayed. The variance dimension trajectory displayed in this mode is the multifractal model of the transient obtained with feature extraction parameters. Notice that this window has no scroll bars; it is anisotropic in that it will always display the entire signal to the level of detail possible within the size of the window. Thus, a larger window size will allow a greater level of detail to be displayed.

The lower window in the view serves two purposes. In its default mode, it provides a zoomed view of the current raw signal. Every sample in the raw signal is displayed on a pixel by pixel basis. Horizontal resizing of this window does not affect the level of detail seen by the user but does change the number of samples displayed. A horizontal scroll bar is available in this viewing mode so that different parts of the signal can be viewed. Vertical resizing of the lower window, as with the upper window, will provide a zooming effect whereby the magnitude of the signal becomes more detailed. In the second view mode for the lower window, pertinent training and batch classification information is displayed. For instance, if this viewing mode is selected after training, classification results will be shown for each sample in the training set along with the total error achieved

for the training set. Figure 5.2 shows the program's display area when the alternative viewing modes have been selected for both the top and bottom viewing windows.

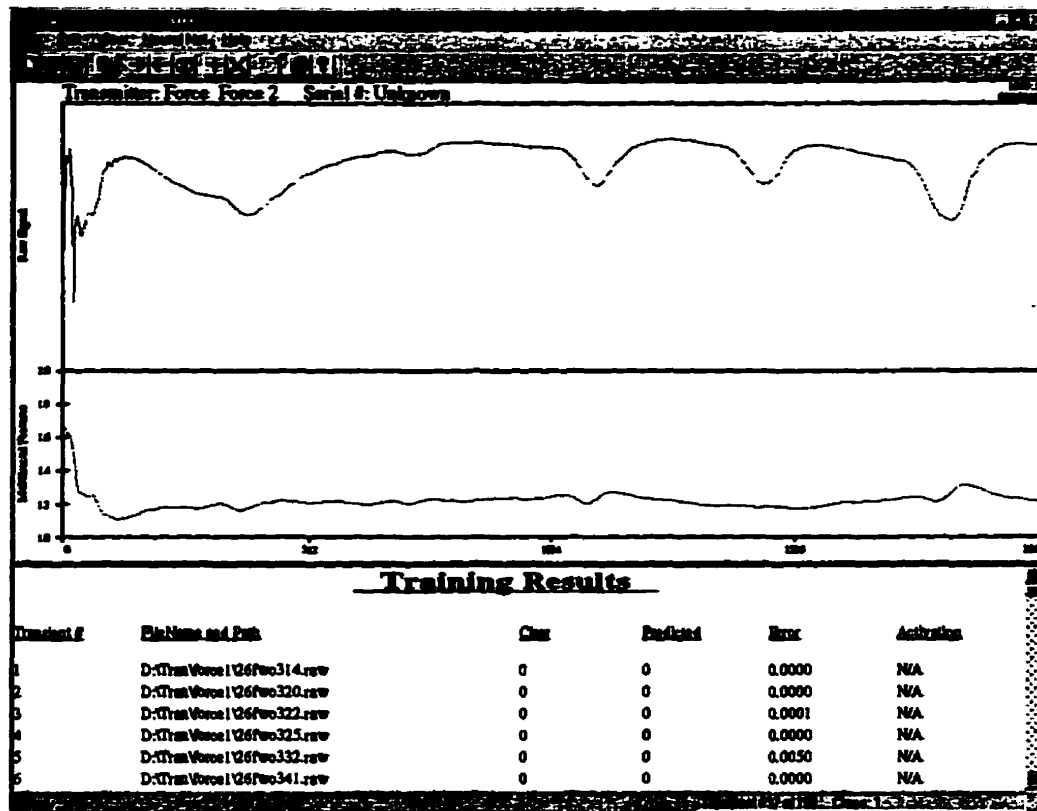


Fig. 5.2. TAC-MM user display area; secondary view options.

As with most Windows programs, TAC-MM makes use of the status bar at the bottom of the window to display pertinent information. The left side of the status bar displays a more detailed description of menu functions or push buttons as the user passes the mouse pointer over them. On the right side of the status bar, an index of the current transient in the display window is given along with the class number of the transient. For instance, if there are 50 transients in the current document, the 13th one is currently being displayed, and it belongs to class number 5, the right side of the status bar will read "Transient # 13 of 50. Class: 5".

5.1.2 The File Menu

Before discussing the user commands in the File menu, the structure of the data in TAC-MM will be explained. For this program, a document is defined as a database or collection of transient models. The document completely defines the set of transients and all pertinent parameter settings for the current classification problem. Since TAC-MM is a SDI application, only one document can be opened at one time.

File: New

This menu function allows the user to start a new document or database of transients. It brings up the dialog box shown in Fig. 5.3 for the user to enter pertinent

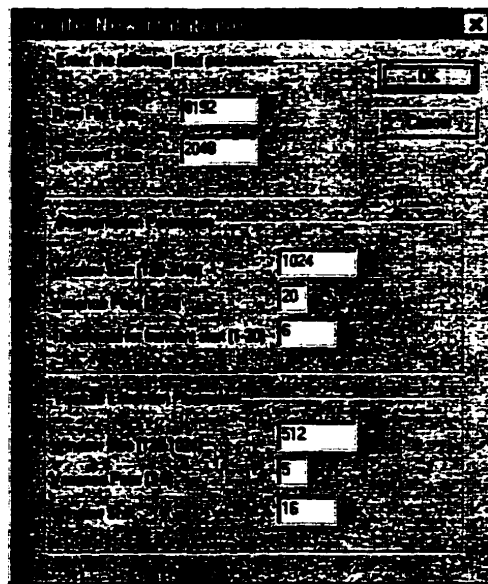


Fig. 5.3. File: New dialog box.

information about the new document. The TAC-MM software can effectively handle a raw file size of up to 25000 samples and any reasonable transient size within the raw file. Notice that various parameters for the fractal segmentation and feature extraction can be specified by the user. Specifically, the window width, W , for the segmentation and for

feature extraction can be set individually as discussed in Chapter 3. Also, the number of ordered pairs, K , for the LSR must be chosen for both the segmentation and feature extraction stages. Note that the sequence of time increments between the ordered pairs is fixed automatically as linear for the segmentation stage and as dyadic for the feature extraction stage. The threshold for triggering the start of the transient, τ from Eq. 3.1, must also be selected by the user. For the segmentation stage, the window spacing, W , is automatically set to 1. However, as detailed in Chapter 3, W must be specified for the feature extraction stage in order to establish the amount of data reduction to be achieved.

For comparison purposes, the software can accomplish the data reduction and modelling of the transient by a simple moving average instead of using the multifractal method described up to this point. In this scheme, supersamples are generated by averaging a number of contiguous samples from the raw signal as defined by the window shift parameter entered for the feature extraction. Thus, the resulting model is the same size as a multifractal model with the same window shift parameter. To enable this modelling method, the value -1 should be entered for the number of variance pairs in the feature extraction. The segmentation stage of the system is not affected in any way by using this alternative modelling technique.

File: Open  , Save  , Save As

These menu commands allow the user to load and save the document. Where appropriate, the Windows 95 common file dialogs are brought up for selecting a filename and verifying the command. It should be emphasized at this point that, in order to keep documents to a manageable size, only the modelled transients and a path to the raw source file are saved in the document. If the user chooses to view any part of the raw source file at

a later time, the program will automatically attempt to load it from the originally specified file path. If the source file is not available at this location, then it will not be displayed on the screen, but the model of the transient will still be available.

File: Print  , Print Preview, Page Setup

These commands also perform their standard functions. The print command will produce a hardcopy of either the upper or lower viewing window, depending on which one is currently active. A simple mouse click in either the top or bottom window will select either as being the active one. It is a good idea to check the print preview before printing to ensure that the correct window is indeed selected. The page setup command allows the user to select the paper size, the paper orientation, the margins, the printer, and other printer specific parameters.

File: Recent File List

After the TAC-MM software has been used several times on a single machine, the file menu will contain the names of up to four documents most recently used. Selecting any of these files will automatically load the document without having to search for the name in the Open File dialog box.

File: Exit

This command will close TAC-MM. However, before closing, it will check that no unsaved changes have been made to the document and give the user an opportunity to save the changes.

5.1.3 The Edit Menu

The Edit menu contains the common Windows copy (to clipboard) function and commands for making various changes to the current document. These changes include

the ability to change any of the fractal parameters originally set for the document and for adding new transients or deleting existing ones from the database.

Edit: Copy Image 

This command copies the image in the upper viewing window to the Windows clipboard as a device independent bitmap (DIB) for insertion into reports and other graphical or text-based presentations.

Edit: Add Transient 

Selecting this command allows the user to add a transient to the current training database (the document). It brings up the common file selection dialog box and prompts the user to choose a filename. Then, it will check to see if the file matches the expected format. Specifically, it checks for the correct number of 16 bit integer samples as defined by the user during the creation of the document. The raw file should also contain specific information about the transmission in a file trailer as per Appendix A to this thesis. Once the file is loaded into memory, the program commences with the segmentation and feature extraction process. Depending on the speed of the machine, the length of the raw file, and the parameters set for the variance dimension trajectory calculations, this process may take a few seconds. However, unless the user chooses to change the fractal parameters in the future, this will be the only time that these calculations are performed on the given transient. Then, the user will be prompted to enter the class number of the transient for the neural network training. Finally, the modelled transient, along with other pertinent information from the file trailer, is added to the document immediately following the transient currently displayed on the screen.

Edit: Delete Transient 

This command allows the user to delete a transient from the current document. When selected, it will prompt the user to confirm the action and then deletes the transient which is currently shown in the upper viewing window.

Edit: Fractal Parameters

Since TAC-MM is primarily a research tool, a feature has been included so that the parameters for the variance fractal dimension trajectories can be easily modified without starting a new document. Choosing this command brings up the dialog box shown in Fig. 5.4. The checkboxes above the segmentation and the feature extraction parameters select



Fig. 5.4. Edit: Fractal Parameters dialog box.

the group of parameters that are to be changed. Either one or both can be checked. The raw file size and the transient size cannot be changed unless a new document is started, but are displayed in the dialog box as a reminder for determining the data reduction desired. The

method of modelling the transients by averaging can again be selected by entering the value -1 for the number of variance pairs in the feature extraction. When the user selects OK, the update process commences and could take several minutes because the new models for each transient must be individually constructed from the original raw files. It is especially time-consuming if new segmentation parameters are set because of the single sample window spacing in that stage. It is important to note that when the fractal parameters have been changed, the PNN will require training to adapt itself to the new transient models.

5.1.4 The View Menu

The view menu contains the commands for changing the appearance of the window, the data that appears on the screen, or the display modes of the viewing windows.

View: Toolbar, Status Bar, Split

These are common Windows menu commands for selecting the appearance of the program window. If there is a check beside the Toolbar menu item, then the toolbar will be displayed. The same applies to the Status Bar selection. In their default modes, these features are selected and both the toolbar and the status bar are visible. Selecting the Split command allows the user to change the vertical position of the splitter bar between the two windows to increase the size of either one of them. This function can just as easily be performed by dragging the splitter bar with the mouse, however, the menu function has been retained for standardization.

View: Segmentation View / Transient View

A checkmark beside either of these menu items selects the mode of the upper viewing window. As shown in Fig. 5.1, the default mode is Segmentation view. The Transient viewing mode is shown in Fig. 5.2.

View: Raw Signal

A checkmark beside this menu item allows the raw signal of the current recording to be displayed in the upper viewing window. By default, this menu item is checked, but the user may choose to disable it in order to increase the display detail of the variance dimension trajectory.

View: Fractal Trajectory

A checkmark beside this menu item allows the fractal dimension trajectory of the current recording to be displayed in the upper viewing window. By default, this menu item is checked, but the user may choose to disable it in order to increase the display detail of the raw signal.

View: Zoomed Raw Signal / Classification Stats

A checkmark beside either of these menu items selects the mode of the lower viewing window. As shown in Fig. 5.1, the default mode displays the zoomed raw signal. The classification stats viewing mode is shown in Fig. 5.2.

View: Next 

This command causes the next transient in the database to be displayed. Depending on the type of view selected, either the segmentation process or the model of the transient will be displayed. If, however, the original path of the raw file is no longer valid, then only the modelled transient can be displayed.

View: Previous 

This command causes the previous transient in the database to be displayed in the viewing area.

View: Search 

This command brings up a dialog box which allows the user to enter search parameters to find a particular transient in the database. The user can choose to search for a specified class number, transmitter make, transmitter model, transmitter serial number, transmit date, or transmit time.

5.1.5 The Neural Net Menu

The Neural Net Menu contains all commands which directly pertain to the operation and training of the PNN classifier.

Neural Net: Classify 

This command brings up the standard Windows dialog box for selecting a file. When a file is chosen, it is checked for conformance to the expected format and loaded into memory. However, since the transient may be completely unknown, missing information in the fields of the file trailer is acceptable. Segmentation and modelling of the transient then take place in accordance with the parameters set for the current document. Finally, the PNN classifies the modelled transient and computes a Bayesian confidence level. At this point, if the result obtained at each summation neuron was equal to zero, the software assigns a meaningless predicted class of -1 and a confidence level of 0.0%. This means that the unknown case is dramatically different from all cases in the test set and the PNN is unable to render a decision. The results are displayed in a dialog box as shown in Fig. 5.5. Notice also that the activation of the winning neuron is displayed. If this value is

below a certain threshold, a meaningless predicted class of -1 and a confidence level of 0.0% will be displayed. The neuron activation information is required for selecting rejection thresholds as will be discussed later in this section and in Chapter 6.

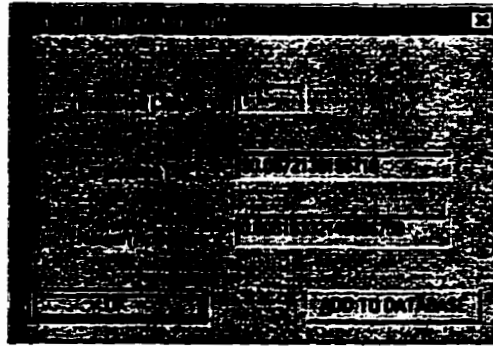


Fig. 5.5. PNN: Classify results dialog box.

After the results are displayed, the user is given the option to simply accept the classification results or to add the transient to the current database in order to increase the number of training samples for that class. When Add to Database is selected, the program brings up the dialog box shown at Fig. 5.6 in order to enter information about the transmitter if it did not exist in the raw file. If the transmission is truly from an unknown

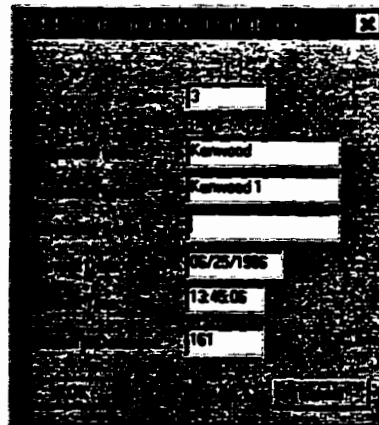


Fig. 5.6. PNN: Classify - Add Transient to Database dialog box.

source, then only the date and time will probably be included in the raw file and the user will have to fill in the other fields if information is available.

Neural Net: Batch Classify

This function is another which has been specifically designed to aid in the role of research tool for TAC-MM. It allows classification of several unknown transients at one time and produces a report listing the results achieved for each transient. When the command is selected, the Windows file dialog box is displayed and a batch list file (.blf) must be chosen. This file, which must be formatted as per Appendix A, specifies the file paths and correct classes of the transients to be classified. After the batch classification process has been completed, a report of the results can be displayed in the lower viewing window by selecting the appropriate viewing mode. The report may also be printed by selecting the Print command while the batch classification results are displayed in the lower viewing window. Appendix E of this thesis contains examples of these printed reports. If a predicted class of -999 is shown in the report for a certain case, this means that there was no transient found in that particular file.

Neural Net: Mode Parameters

Since there currently exists some inconsistencies with the segmentation process in this system, a feature referred to as *multimodal segmentation* has been added as an option to the classification scheme. In multimodal segmentation, several possible transient start points are flagged in the raw file of an unknown transient. Then, separate transient models are constructed starting at each of these points and each one is classified individually by the PNN. The predicted class of the model classified with the highest winning neuron activation level is chosen as the correct one. This feature can be enabled by selecting the

Mode Parameters menu item and choosing multimodal segmentation in the dialog box shown in Fig. 5.7. Then, the four parameters listed in the dialog box must be set. The

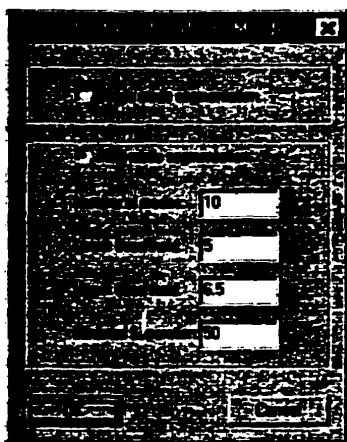


Fig. 5.7. PNN: Mode Parameters - Select Classification Mode dialog box.

number of modes is simply the number of different fractal models to be constructed for each unknown transient. This value should remain somewhat conservative as execution time may become a factor. The lower and upper thresholds represent the value τ used in triggering the start of the transient as per Eq. 3.1. The successive segmentation positions are triggered by a linearly increasing threshold bounded by these two values. For instance, if the lower threshold is set to 5, the upper threshold is set to 6, and there is to be 6 segmentation modes, then the successive transient models will be segmented using thresholds of 5.0, 5.2, 5.4, 5.6, 5.8, and 6.0. The final parameter in the dialog box specifies the minimum separation, in number of samples, between successive transient startpoints. This provides a means to overcome the potential of a relatively extreme, yet false, change in the variance dimension trajectory causing identical transient models all triggered at the same spot. Thus, by setting the minimum separation to a significant value, the chances of

finding the correct transient start are significantly improved, and furthermore, it can be ensured that each model is substantially different.

Neural Net: Initialize Sigmas

This function begins the training process described in Subsection 4.3.2 and sets all of the PNN scaling parameters to a common value. Before proceeding with training though, the user is prompted to provide a search range and the number of search points for the initial global search. The program automatically switches over to the golden section minimization after completing the global search as defined by the user. This will continue until successive iterations yield insignificant improvement or until it is halted by the user. Training to this stage results in the basic PNN described in Sections 4.1 to 4.3, except that the continuous error function is used instead of the simple counting error function.

Neural Net: Optimize Sigmas


Selecting this command implements the multiple-sigma PNN model discussed in Section 4.4. A simple dialog box is brought up that allows the user to start and stop conjugate gradients training or to return to other program operations. Prior to invoking the multivariate optimization, though, the PNN scaling parameters must be initialized using the previously described menu function. Once this minimization procedure begins, it will continue until three successive iterations fail to yield significant improvement or until the user selects stop. During this optimization, regular progress updates are displayed to aid in determining when sufficient training has been completed.

Neural Net: Set Rejection Threshold

This command allows the user to set a threshold for rejecting transients that do not belong to any of the classes represented by the training set. This feature is critical in any

practical application as a means to avoid misclassifying completely new transients. It is implemented during the classification process by comparing the largest summation neuron in the network to the rejection threshold before normalization. In this way, a strict comparison can be made between the two values and if the activation of the winning neuron is not large enough, then the case is rejected. In a typical PNN implementation, the rejection mechanism involves thresholding the Bayesian confidence, which provides the relative activation of the winning neuron compared to the losing neurons. However, the mechanism implemented in TAC-MM performs an absolute comparison instead of a relative comparison. This is much more useful in the typical situation where the training set is not mutually exclusive and exhaustive. Since the default setting for this value is zero, it must be initialized to some higher value before this feature becomes functional.

5.1.6 The Help Menu

The typical functions found in a Windows Help Menu are not implemented in this version of TAC-MM. Future revisions will likely include information as it is presented in this chapter. However, there is one function, **About TAC-MM**  , which can be selected to show the current version of the software.

5.2 Verification of the System Software Modules

There are a total of 47 source code files for TAC-MM as listed at Appendix B to this thesis. A brief overview of each file, in terms of its content and purpose, is given in Table B-1 at the start of the Appendix. To verify that the numerical algorithms coded in these files actually perform as expected, several tests have been conducted. Details of these tests will now follow.

5.2.1 *Multifractal Segmentation and Feature Extraction*

Since the segmentation and feature extraction stages of TAC-MM are both implemented using many of the same routines, they could be verified concurrently. First, the variance fractal dimension trajectory calculation was checked for accuracy using signals with known fractal dimensions. This started with a simple verification using a sine wave and white Gaussian noise generated by the Park-Miller implementation of a linear congruential random number generator [PTVF92]. These signals were measured at the expected fractal dimension measurements of 1.0 and 2.0 respectively. However, these basic tests only cover the most straightforward of the possible variance fractal dimensions and therefore do not provide solid evidence of functionality. To accomplish this, several signals with intermediate fractal dimensions were generated using a technique referred to as *direct spectral filtering* [Kins94c]. Specifically, this technique provides an efficient means, using spectral analysis, for synthesis of signals which exhibit fractional Brownian motion (fBm). In its implementation, a spectral exponent, β , is specified to generate fBm with a certain characteristic degree of persistence. The fractal dimension of the fBm can be calculated using

$$D = E + \frac{3-\beta}{2}, (1 \leq \beta \leq 3) \quad (5.1)$$

which was first introduced in Chapter 2 for the closely related spectrum-based fractal dimensions.

The direct spectral filtering algorithm is implemented using a four step process as follows [Kins94c]:

1. Generate Gaussian noise on N points.

2. Calculate the discrete Fourier transform of the Gaussian noise.
3. Filter the spectrum of the Gaussian noise in accordance with the desired β .
4. Calculate the inverse Fourier transform to obtain the fBm.

Further details about this process can be interpreted from the source code listed at the end of Appendix C to this thesis.

To verify the segmentation and feature extraction stages of TAC-MM, 9 different fBm signals, spanning the range between 1.1 and 1.9, were generated using the technique described above. Then, a new document was created in TAC-MM with the following parameter settings:

Raw File Size	16384
Transient Size	4096
Segmentation Window Size	2048
Segmentation Variance Pairs	25
Segmentation Threshold	1
Feature Extraction Window Size	512
Feature Extraction Variance Pairs	5
Feature Extraction Window Shift	16

Fig. 5.8. Parameters for verification of TAC-MM variance dimension calculations.

At the expense of processing time, the parameter settings for segmentation have been set to the maximum values allowed by the program so that fractal dimension calculations are as accurate as possible. The segmentation threshold is not important in this test and has therefore been set to its lowest possible value to ensure that at least some part of the raw signal is tagged for multifractal modelling. The parameters for the feature extraction stage are set as typical modelling parameters to emphasize the contrast between the two stages.

The 9 fBm signals were then added to the new document database so that segmentation and feature extraction could be performed on each one individually. The calculated fractal trajectory of each test signal is shown at Appendix C for both the segmentation and the feature extraction stages. The variance dimension trajectories were then averaged to obtain a single fractal dimension, \bar{D}_σ , for each signal. These results are presented in Table 5.1. The standard deviation of the variance dimensions calculated on successive windows is also shown in order to provide a quantitative representation of the level of variation exhibited along the trajectory of the different signals.

Table 5.1: Verification of fractal dimension trajectory calculations.

β	$\bar{D}_\sigma, \text{ expected}$	$\bar{D}_\sigma, \text{ segmentation, (Std Dev)}$	$\bar{D}_\sigma, \text{ feature extraction, (Std Dev)}$
2.8	1.1	1.164 ,(0.028)	1.197 ,(0.041)
2.6	1.2	1.241 ,(0.027)	1.259 ,(0.043)
2.4	1.3	1.326 ,(0.025)	1.336 ,(0.042)
2.2	1.4	1.415 ,(0.023)	1.422 ,(0.040)
2.0	1.5	1.505 ,(0.022)	1.508 ,(0.039)
1.8	1.6	1.592 ,(0.022)	1.593 ,(0.037)
1.6	1.7	1.676 ,(0.021)	1.677 ,(0.035)
1.4	1.8	1.753 ,(0.020)	1.756 ,(0.033)
1.2	1.9	1.822 ,(0.018)	1.826 ,(0.030)

Analysis of the results shows that the calculated fractal dimensions are much closer to the expected fractal dimensions near the middle of the interval and tend to skew slightly towards the middle at the extremes. Figure 5.9 depicts this skewing effect more clearly. This result seems somewhat suspicious, especially since accurate fractal dimension calculations were previously achieved at the extreme edges of the interval for a

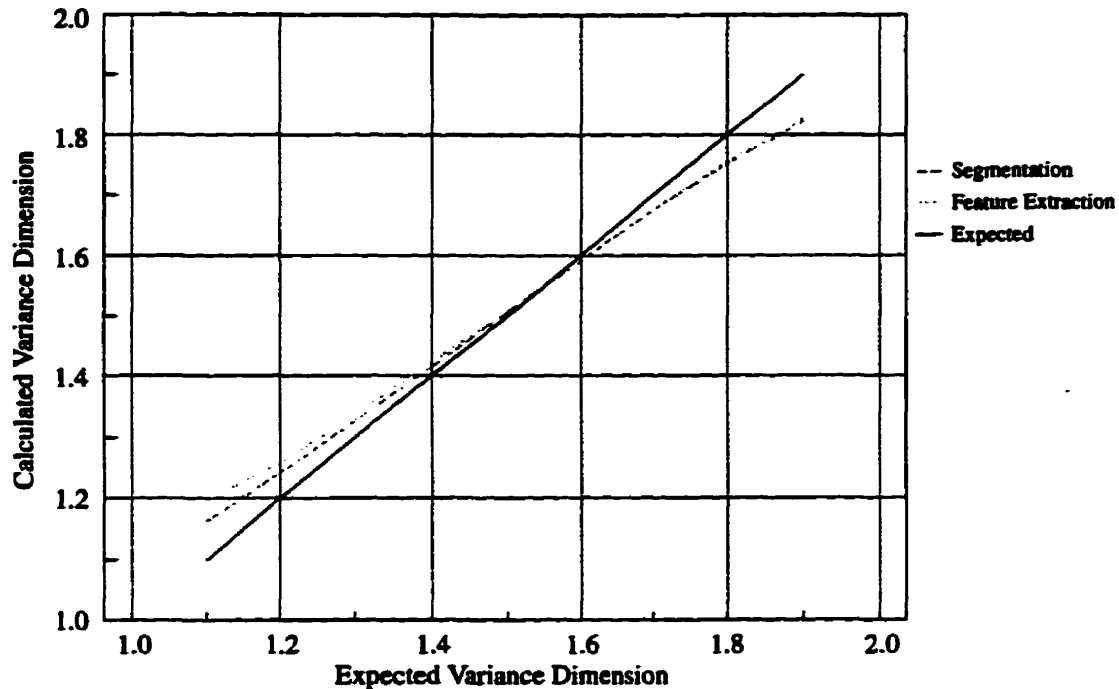


Fig. 5.9. Comparison of expected and calculated fractal dimensions.

sine wave and white Gaussian noise. The irregularities encountered in this situation are possibly caused by incorrect variance dimension calculations, fBm signals which are not completely accurate in terms of their noise characteristic, or a combination of both of these factors. Since excellent results were achieved for most of the variance dimension calculations, especially near the centre and at the extreme ends of the interval, it can be concluded that the majority of the problem is with the actual noise characteristic of the fBm signals in the test set. This deduction can be supported by a simple test where a fBm signal with $\beta = 1.0$ is generated using the same routine as the other signals. Theoretically, a signal consisting entirely of white Gaussian noise, with $D_G = 2.0$, should be produced. Figure 5.10 shows that the signal obtained from this procedure is obviously correlated to some extent and is therefore not white Gaussian noise. Adding this signal to the current

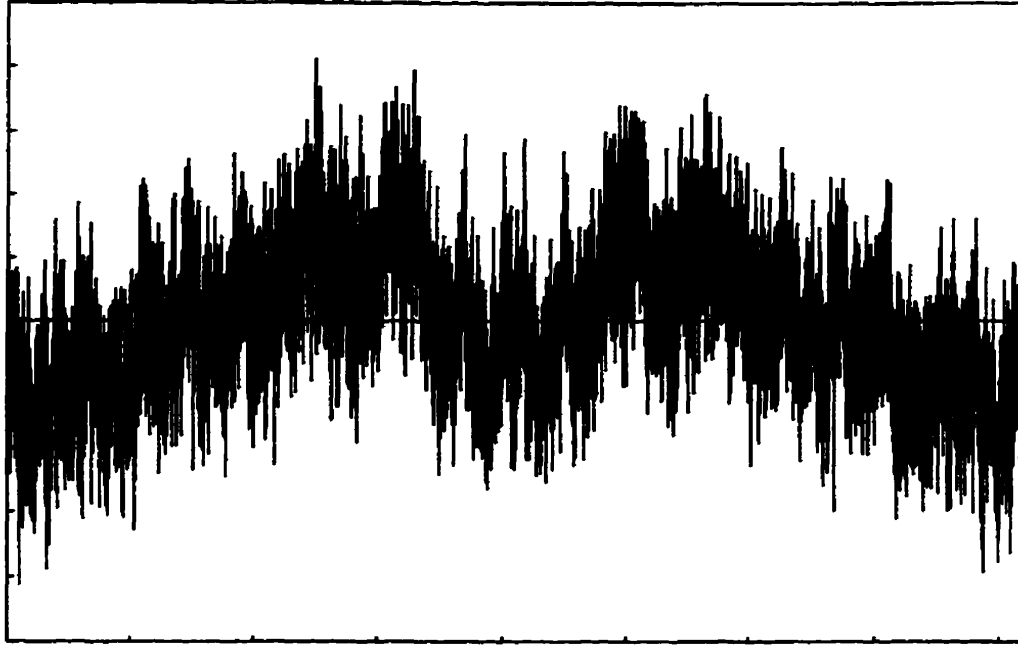


Fig. 5.10. fBm signal generated using direct spectral filtering with $\beta = 1.0$.

TAC-MM document yields $D_G = 1.88$ in the segmentation stage and 1.89 in the feature extraction stage, which both seem reasonable from observation of the signal.

Despite the previous conclusion that the testing discrepancies were primarily resulting from inadequate test data, it would be somewhat presumptuous to state that the results obtained perfectly characterize the variance dimensions of the fBm signals in the test set. There are simply too many possible parameter settings in the implementation of these variance dimension measurements to make such a statement. It will be shown later that variations in the fractal parameters can have a significant impact on the results achieved for dimension calculations. Therefore, it is likely that a small portion of the observed error in the previous test was caused by a lack of range depth in the variance dimension calculation. To increase the depth of the variance dimension calculation would require increasing the window size and/or the number of variance pairs. This would,

however, be at substantial cost in terms of processing time and would not yield proportionately better results.

Given the imperfect results achieved during the verification of the variance dimension calculations, it is worthwhile to digress somewhat and discuss the impact of this apparent problem. For the set of test signals, it is significant that the measured variance dimension increased as the value of β decreased, or as the signals became more uncorrelated. Thus, we know that the measured results do reflect the relative degree of correlation for a given signal, despite the possibility that they may not indicate the exact fractal dimension. For fractal segmentation, this is sufficient since we are concerned only with finding relative changes in a signal's correlation; the actual variance dimension is of no concern. For the feature extraction stage of our analysis, this is equally true as long as the parameters for the dimension calculations remain consistent for all of the models in a given database. In fact, for the fractal modelling, it is beneficial to work with a depth of dimension calculation which is less than ideal in order to emphasize the varying multifractal characteristics of a signal.

The last test performed to verify the operation of the multifractal analysis in this system was to prove that the dyadic sequence of time increments for the feature extraction is indeed more appropriate than the linear sequence used in segmentation. Referring back to Table 5.1, it is clear that the standard deviation of the fractal trajectories obtained using feature extraction parameters are significantly higher than the trajectories obtained using segmentation parameters. Since feature extraction is intended to emphasize the important characteristics of the signal, this is the desired effect. It shows that these parameters allow a mechanism which is capable of representing various features using variation within a

broader scale. However, the data shown in Table 5.1 is considerably biased to this effect because the depth of the fractal analysis is much lower for the feature extraction than for the segmentation. In order to remove this bias, a new document was created such that all of the fractal parameters, except the sequence of time increments, were set identically as follows:

Raw File Size	16384
Transient Size	4096
Segmentation Window Size	1024
Segmentation Variance Pairs	8
Segmentation Threshold	1
Feature Extraction Window Size	1024
Feature Extraction Variance Pairs	8
Feature Extraction Window Shift	1

Fig. 5.11. Parameters to show contrast between dyadic and linear time increments.

The same fBm signals were then added to the new document and the results obtained are shown in Table 5.2. Again, sharp contrast in the standard deviation of the trajectories

Table 5.2: Contrasting dyadic and linear time increments for D_G calculations.

β	\bar{D}_G , expected	\bar{D}_G , segmentation ,(Std Dev)	\bar{D}_G , feature extraction ,(Std Dev)
2.8	1.1	1.159 ,(0.021)	1.213 ,(0.044)
2.6	1.2	1.232 ,(0.021)	1.280 ,(0.046)
2.4	1.3	1.313 ,(0.022)	1.358 ,(0.047)
2.2	1.4	1.398 ,(0.024)	1.443 ,(0.046)
2.0	1.5	1.483 ,(0.025)	1.531 ,(0.044)
1.8	1.6	1.565 ,(0.025)	1.620 ,(0.042)
1.6	1.7	1.645 ,(0.024)	1.706 ,(0.036)
1.4	1.8	1.719 ,(0.022)	1.786 ,(0.028)
1.2	1.9	1.786 ,(0.019)	1.855 ,(0.019)

obtained for the segmentation and the feature extraction stages is observed. It can therefore be concluded that the dyadic sequence of time increments is indeed more suitable than a linear sequence in this context. Notice also, as previously alluded to, the significant change in some of the dimension measurements resulting from the different parameter settings.

5.2.2 The PNN Classifier

Verification of the basic PNN classifier used in TAC-MM was a relatively straightforward process. A set of 16 sinusoidal signals was generated for training the PNN. The signals were identical except for a phase shift such that the entire range of 0 to 2π radians was spanned by the signals in increments of $\pi/8$ radians. Each signal could then be treated as an individual class because of its unique phase characteristic. Since the holdout method of PNN training requires a minimum of two cases from each class, a second identical set of signals was generated and white Gaussian noise was added to the new signals such that each one had a signal to noise ratio (SNR) of 10 dB. Combining these 32 signals, which represented 16 different classes, a sufficient training set was developed to verify the operation of the PNN.

A new document was then created in TAC-MM with the following parameters:

Raw File Size	8192
Transient Size	2048
Segmentation Window Size	*512
Segmentation Variance Pairs	*3
Segmentation Threshold	0
Feature Extraction Window Size	*512
Feature Extraction Variance Pairs	-1
Feature Extraction Window Shift	4

Fig. 5.12. Parameters for verification of PNN module in TAC-MM.

It is important to discuss a few of the parameters selected above. First, setting the segmentation threshold to zero activates a feature of the software which allows it to bypass the segmentation stage of this system. Instead, each “transient” is modelled starting at a position $1/4$ the number of samples into the raw signal. This feature is critical in the verification of the PNN module so that it can be isolated from the other modules and tested separately. Equally important for isolation of the PNN module is setting the feature extraction variance pairs to -1 which, as previously indicated, replaces the fractal modelling process with a simple moving average. Since the feature extraction window shift parameter has been set to four, the 2048 samples in the modelled part of the signal are reduced to 512 samples by the averaging scheme. The parameters indicated by an asterisk in the above chart are insignificant due to the exclusion of the fractal segmentation and modelling stages.

After the 32 training signals were added to the document, the PNN scaling parameters were initialized using the default search range. Within a few seconds, the initialization routine had ended at an average error of 0.0, as calculated by Eq. 4.19 for each training case. Also, it showed a total of zero misclassifications for the training set. This is especially significant since, using the holdout method, the neural network is classifying the noisy signals based only on knowledge of the clean signals and vice-versa. Such results could definitely not have been achieved by a multi-layer feedforward neural network given so little training time!

To perform the actual verification of the PNN, it was necessary to produce a set of test signals which would not be used for training of the network. This time, however, the 16 sinusoidal signals were generated such that each one was virtually buried in white

Gaussian noise at a SNR of -10 dB. Before discussing the results obtained, it is important to point out that the white Gaussian noise added to each signal within this data was synthesized from a unique sequence of pseudorandom numbers. The sequence of pseudorandom numbers was obtained using the Park-Miller implementation of a linear congruential random number generator [PTVF92]. Great care was taken to ensure that each signal, both in the training set and the test set, had a different sequence so that the details of the noise would remain completely meaningless during classification. Testing the network with this set of signals yielded perfect results whereby each of the corrupted signals was classified with 100% confidence level. A similar test set was then generated such that each signal had a SNR of -15 dB. The massive deviation of the signals in this test set from any of the original signals finally caused the PNN to fail. With all test signals in this set, the distance summation became too large and its negative exponent was driven to zero as discussed at the end of Subsection 4.2.3. Appropriately, the software yielded a meaningless classification result and displayed zero confidence level after each prediction.

Due to the periodic and well-behaved nature of the previous set of test signals, there was no opportunity to verify the training of the advanced PNN features implemented in TAC-MM. Specifically, none of the features within the signals were any more important than others for classification purposes. Therefore, no benefit could be realized by varying individual scaling parameters for each of the discrete elements of the input signal. To verify the advanced PNN structure and training routines, yet another set of test signals was developed as shown in Fig 5.13. This set of test signals was especially designed to cause classification failure using the basic PNN model with only one scaling parameter. The signal begins with unique white Gaussian noise followed by a relatively low power sine

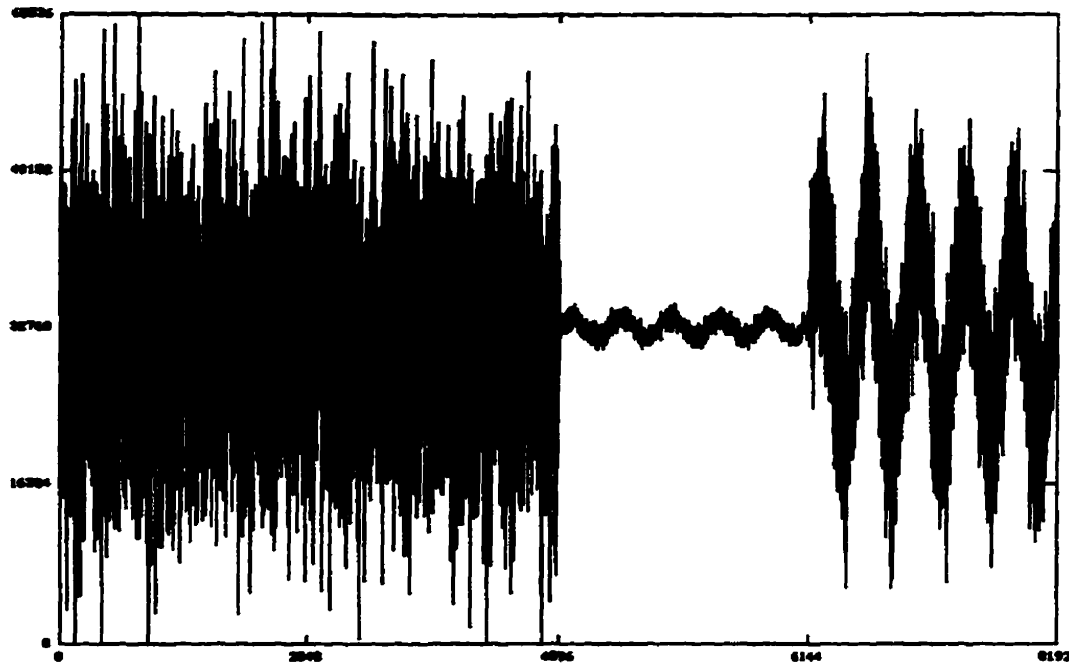


Fig. 5.13. Test signal generated to cause the basic PNN to fail.

wave which is characterized by its phase only. Finally, the signal ends with another phase characterized sine wave which has 10 times the power of the previous portion of the signal. The sinusoidal portions of the signal shown, when considered separately, are contaminated with additive white Gaussian noise at 5 dB SNR. Examination of this signal suggests that, during the distance summation, irrelevant variations in the high-power noise at the start of the signal would completely bury any characteristic phase differences in the low-power sinusoidal portion of the signal. Furthermore, the problem is compounded by the phase differences and noise contamination of the relatively high-power sinusoid at the end of the signal.

The set designed to test the advanced PNN structure consisted of 64 signals representing 16 different classes. Each class was a permutation of the 0 , $\pi/8$, $\pi/4$, $3\pi/8$ phase shifts for the low-power sine wave and the same four phase shifts for the high-power

sine wave. Again, special care was taken to ensure that all additive white Gaussian noise, as well as the pure noise at the start of the signal, was unique. For each of the 16 classes, there was four signals with different magnitudes of noise added to the sinusoidal sections. One of the four signals had no noise added to the sinusoidal sections and the other three had SNRs of 20 dB, 15 dB, and 10 dB.

A new document was then created in TAC-MM with the following parameters:

Raw File Size	8192
Transient Size	6000
Segmentation Window Size	*256
Segmentation Variance Pairs	*3
Segmentation Threshold	0
Feature Extraction Window Size	*256
Feature Extraction Variance Pairs	-1
Feature Extraction Window Shift	30

Fig. 5.14. Parameters for verification of enhanced PNN structure in TAC-MM.

It is important to review some of the parameters selected above. First, the segmentation threshold has been set to zero to bypass the segmentation stage of this system and model the “transients” starting at a position 1/4 the number of samples into the raw signal. The transient size has been set to 6000, which means that the portion of the signal modelled as the transient will have 2048 samples of pure white Gaussian noise, 2048 samples of the low-power sine wave, and 1904 samples of the high-power sine wave. Also, the feature extraction variance pairs parameter is set to -1 to replace the fractal modelling process with a moving average. Since the feature extraction window shift parameter is set to 30, the 6000 samples in the modelled part of the signal are reduced to 200 samples by the

averaging scheme. The parameters indicated by an asterisk in the above chart are again insignificant due to the exclusion of the fractal segmentation and modelling stages.

The 64 signals in the training set were then added to the document described above. The scaling parameters were initialized using a comprehensive search range at 1000 points spread logarithmically over the interval from 0.0001 to 1000. After the global search and the single variable golden section minimization, the basic PNN model was able to correctly classify, using the holdout method, only 18 of the 64 training signals. The average error across all training signals was 0.7415. At this point, no attempts were made to test the network with unknown signals as classification results would be certainly unfavourable. Instead, training was continued using the conjugate gradients multivariate optimization scheme. Within five minutes (4:31), the routine had “learned” appropriate scaling parameters and the PNN could correctly classify each signal in the training set using the holdout method.

To verify that this network could correctly classify unknowns, a similar set of 16 signals was generated such that each of the two sinusoidal parts had a SNR of -10 dB. Once more, the leading and additive white Gaussian noise sequences were unique for all cases. The network correctly classified each signal in the set with 100% confidence. This example clearly demonstrates the power of the enhanced PNN with separate scaling parameters for each input.

One final set of test signals, identical to the previous except with a SNR of -15 dB, was then generated for verifying this network. In this test, only 6 out of 16 signals were classified correctly, signifying that -15 dB is again near the level of noise where the classifier fails. Another deliberate failure can be induced by attempting to classify a

completely different signal from any of those in the training set. This was accomplished by using one of the sine waves generated for the verification of the basic PNN model. Attempting to classify with one of these completely unknown signals caused the negative exponent of all distance summations to become zero; producing a meaningless result with 0% confidence level.

5.3 Summary of Chapter 5

This chapter has demonstrated the various features and limitations of the TAC-MM software package for analysing transmitter transients. Details of the user interface, menu structure, and software implementation have been provided. Then, a comprehensive series of tests was conducted in order to verify that the various computational routines were functioning correctly. To accomplish this, specific stages within the transient analysis and classification scheme were isolated and tested for predetermined results. In doing so, it was shown that the PNN classifier is a very powerful and much faster alternative to the popular MLFN classifier.

Chapter 6 will begin with an introduction to the data used for testing this thesis. Several tests are conducted using this data and results are presented in the form of confusion matrices. Finally, conclusions are made as to the success of this scheme for classifying radio transmitter transients.

CHAPTER VI

CLASSIFYING TRANSMITTER TRANSIENTS

Since the fundamental purpose of this thesis is to develop a system which can be used to predict accurately the source of a radio transmission based on analysis of its transient signature, the TAC-MM software package must be tested using more than just artificially generated data sets. In this chapter, details of testing with a set of actual transmitter transients will be presented. First, Section 6.1 provides a description of the capturing system, the file format, and the composition of the set of transients used to test this thesis. Then, in Section 6.2, particulars of the testing process are discussed along with a presentation of the results achieved for the various tests. This chapter closes with an analysis of these results in Section 6.3, focusing specifically on the various confidence measures available for this system.

6.1 The Thesis Test Set

6.1.1 *Transient Capturing System*

The set of transients used for testing this thesis was captured by the Communications Research Centre (CRC), Ottawa, using a system specially implemented for this purpose by Toonstra and Kinsner at the University of Manitoba [Toon97]. It uses an *Icom IC-R7000* communications receiver and a *SoundBlaster 16* sound card running on a PC. The *SoundBlaster* continuously samples the output of the *IC-R7000*'s discriminator at 44,100 kHz to 16 bits accuracy on its left channel. The signal is stored in a roughly 32 kB (16,348 samples) circular buffer until the system receives a marker trigger signal

through the right channel of the *SoundBlaster*. The marker is derived from the speaker output of the *IC-R7000* and activates when the software detects a break in the squelch level beyond a certain threshold. At this point, the buffer collects a final 8,192 samples and disengages from the collection process so that the data can be written to disk for further processing. Each transient file is named uniquely according to a scheme developed by Kinsner.

Since the trigger is very much dependant upon the squelch level, it cannot be relied upon to capture a transient beginning at a consistent point in time. The continuous recording on the left channel, however, provides a means to minimize the impact of the unreliable marker. Specifically, it provides sufficient data to enable the TAC-MM system to perform segmentation of the transients from the ambient channel noise before beginning the modelling and classification processes. Notice that it would be interesting to develop a transient capturing system based on a real-time multifractal analysis as a triggering mechanism. This might locate the transient at a relatively consistent point along its transmission and eliminate the requirement for off-line segmentation as it is done in TAC-MM.

6.1.2 The Transient File Structure

After the transients were collected using the system described above, they were stored on disk in an unresolved circular buffer. While a transient did exist in all of the files, its start point could be found anywhere within the 16,348 samples. Figure 6.1 gives an example of this situation. This type of file format would be very difficult for the TAC-MM software to analyze since there exists two incidents of very sharp change in the variance fractal dimension. The problem occurs when the first significant dimension transition is

from signal to noise, as shown in the example, rather than the expected noise to signal transition.

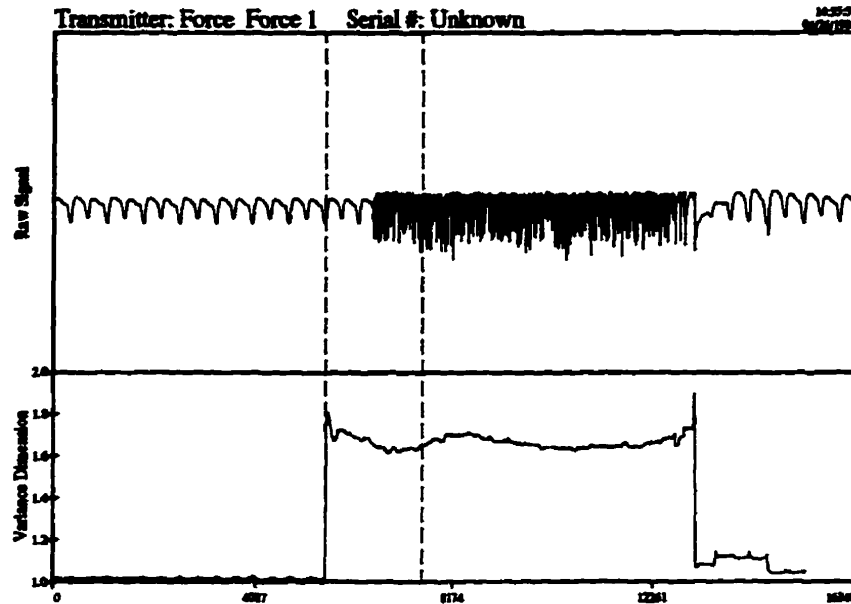


Fig. 6.1. Transient in unresolved circular buffer.

With the signal shown in Fig. 6.1, the start of the transient can be located roughly using visual inspection. Since TAC-MM only requires that the transient be preceded by channel noise for a minimum of 1/4 the duration of the file, realignment of each signal was performed visually so that the transition from noise to signal occurred near the centre of the file. Then, equal portions were truncated from the start and the end of each file to obtain a signal 8,192 samples in size, with the transient beginning approximately in the centre. This procedure was performed deliberately to obtain a smaller size signal, while ensuring that adequate noise was included in the file. Examples of these truncated files can be found at Appendix D to this thesis.

Note that the TAC-MM software would work fine using the larger signal, but since manual adjustment was being conducted anyway, the size of the file was decreased to

improve the speed performance of the fractal segmentation process. It should further be noted that this manual realignment of the signal would have been completely unnecessary if the circular buffer had been resolved at the time of data collection by the acquisition software. This could have been accomplished trivially by writing the buffer to disk starting at the location of the data pointer where the next sample was to have been written as shown in Fig. 6.2.

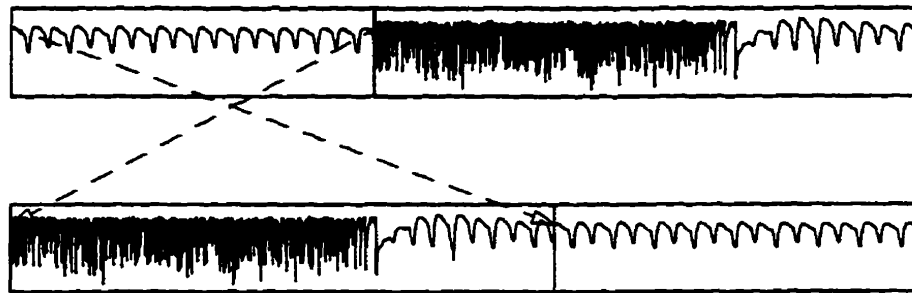


Fig. 6.2. Resolution of circular buffer containing noise and a transient.

6.1.3 Composition of the Test Set

The set of transients used to test this thesis were collected by the CRC in Ottawa, Ontario. They used the acquisition system described in Section 6.1, and the files were provided for this research in their original, unresolved format as shown in Fig. 6.1. The set consisted of 415 transients, distributed approximately evenly among eight different transmitters. Specifically, there were three Kenwood models, three Force models, and two Yaesu models as shown in Table 6.1. Also shown in Table 6.1 is the class identification number assigned to each transmitter model. This number is for use by the classification stage of the system. A single example of segmentation and feature extraction from each transmitter can be found at Appendix D in this thesis. The images shown have been

processed by TAC-MM, using the parameters for the first experiment discussed in the next section.

Table 6.1: Transmitters used for testing the thesis.

Transmitter	Number of Transients	Class Identifier
Force 1	50	0
Force 2	52	1
Force 3	52	2
Kenwood 1	50	3
Kenwood 2	51	4
Kenwood 3	52	5
Yaesu 1	58	6
Yaesu 2	50	7

6.2 Testing and Results

There was a total of three different training sets constructed using the set of transients described above. The three training sets differ in the number and selection of transients used for training the PNN as opposed to the transients held back for validation purposes. After conducting classification tests with the three different training sets, separate experiments were performed to test the PNN's rejection ability, the multimodal segmentation feature, and a new transformation of the fractal modelling process. All tests were conducted using TAC-MM on a 133 MHz Pentium PC, with 48 MB RAM, under the Microsoft Windows 95 operating system.

6.2.1 Training Set # 1 (First 20 Transients)

In this experiment, the first 20 transients collected for each transmitter were used as the training set. This selection of transients represents less than half of the available

transients and has a composition which, being the first 20 transients collected, reflects a situation as it would likely occur in a practical application of this system. A new document was created in TAC-MM using the default parameters for the software. This sets the raw file size to 8192 samples and the length of the transient to 2048 samples. The other parameters, for the fractal segmentation and modelling processes, are insignificant at this time as they will be modified after some initial testing is performed. Then, the 160 transients for this training set were added to the document individually along with the class identification number for each one. This left a total of 255 transients for the validation set.

Before results of this experiment could be assessed in terms of the PNN's classification ability, a sufficient set of fractal parameters for both the segmentation and feature extraction stages had to be found. Since there are six separate parameters to set, a methodology for isolation of some or one of the variables was necessary. Using a systematic approach, a set of good, yet perhaps not optimal, parameters was found. The approach used in these experiments starts with isolation of the segmentation stage by eliminating the fractal modelling of the transient and replacing it with the simple moving average. The amount of data reduction selected using this technique was equal for all trials, at 128 supersamples, to ensure sufficient transient characteristics were retained. In this manner, suitable parameters for fractal segmentation could be found before attempting to find parameters for the feature extraction.

As with any optimization or selection of parameters, an objective measure of performance must be utilized. For the segmentation stage of this experiment, the performance measure was a combination of visual inspection and PNN trials. It should be

noted, despite what may be implied by a visual performance assessment, that this approach is by no means subjective. Instead, the ability to inspect the data visually provides a rapid means for discarding clearly unfavourable results. For instance, if the depth of the fractal analysis is too low, such that significant variations are noticed before transition points, then the user is able to discard this trial without going through the neural network testing process. Another example would be if the segmentation threshold is set too low, the user can clearly see that false transients are being triggered at different points along the ambient channel noise. When results appear to be satisfactory, the performance measure becomes the average error of the PNN when it is tasked to learn the training set. After suitable segmentation parameters have been found, the focus could be shifted over to selection of parameters for the feature extraction stage. This process would also select parameters with respect to the PNN's average training error, but additionally, batch classification results for transients in the validation set were also considered before ending the search.

Conducting the parametric search described above, the configuration found to yield the best results is as follows:

Raw File Size	8192
Transient Size	2048
Segmentation Window Size	512
Segmentation Variance Pairs	10
Segmentation Threshold	8
Feature Extraction Window Size	768
Feature Extraction Variance Pairs	8
Feature Extraction Window Shift	64

Fig. 6.3. TAC-MM parameters for testing Training Set #1.

Since an exhaustive search was not conducted, no claim of optimality will be made for these parameters. However, they do provide a suitable balance between speed, data reduction, and favourable classification results. Notice that the transient size was reduced from 2048 samples down to only 32. After selecting these parameters, the PNN was trained up to the initialization stage only, with all scaling parameters set to the same value. This training took 16 seconds to complete. At this level of training, the PNN could correctly classify 147/160 transients in the training set, using the holdout method. More importantly though, it was able to correctly classify 243/255 transients in the validation set. This represents a success rate of 95.3%. It is also notable that the 255 transients in the validation set were segmented, modelled, and classified in 131 seconds or in about a half second each. Details of the classification results can be found at Appendix E of this thesis. For the present time, the confidence measures and winning neuron activations shown for each validation case should be disregarded as they will be discussed in Subsection 6.2.4 and in Section 6.3.

A more descriptive analysis of the results of this experiment can be derived from a confusion matrix, which is a standard tool used for testing any type of classifier. The matrix shows the various patterns of misclassification that are obtained from a validation set. Table 6.2 is the confusion matrix with the results achieved from this experiment.

Notice that the confusion matrix has one row and one column for each class. Interpretation of the matrix is straightforward. For instance, the number in the Class 2 row and the Class 1 column is the number of cases that are truly members of Class 2 but have been classified into Class 1. Ideally then, the confusion matrix for a perfect classification experiment would be strictly diagonal. In this confusion matrix, the area within the thicker

Table 6.2: Confusion matrix with results from Validation Set # 1.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Class 0	27	1					2	
Class 1	1	30	1					
Class 2		2	30					
Class 3				29		1		
Class 4					31			
Class 5				3		29		
Class 6							38	
Class 7							1	29

lines represents all of the transmitters from the same manufacturer. A misclassification into any of the classes from the same manufacturer is certainly more creditable than a completely inaccurate result.

From this experiment, it can be seen that the transients that were most often classified incorrectly were from the Force 1 and the Kenwood 3 transmitters. However, it is significant that all of the misclassifications in the Kenwood 3 category were mistaken for transmitters built by the same manufacturer. In fact, out of the total 12 misclassifications, only two of them were classified outside of the correct manufacturer. It is also significant that the transmitters represented by Class 4 and Class 6 were classified perfectly every time. Furthermore, results favouring Class 3 and Class 7 are also noteworthy, with only one misclassification each. A conclusion that can be drawn from this analysis is that decisions favouring these classes can be accepted with high confidence, regardless of the Bayesian confidence level calculated by TAC-MM.

A final note will be made about the training process before presenting results for the other experiments. It was previously noted that, in this experiment, the PNN was initialized only, and that all of the scaling parameters were set equal. It was found that optimization of individual scaling parameters, while greatly reducing the classification error for the training set, did not significantly improve the classification ability of the network when dealing with the validation set. In fact, optimization of the scaling parameters caused the network to produce confidence levels at or near 100%, even for incorrect classifications. This is obviously an undesirable side-effect as it renders the already invalid confidence measures useless.

6.2.2 Training Set # 2 (Random Selection of 10)

In this experiment, 10 transients from each transmitter were selected at random to make up the training set. This left a total of 335 transients for the validation set. A similar parametric search was conducted as described in the previous experiment. Since the same transients are being used, just in different set combinations, the parameters selected for segmentation in the previous experiment were again found to be successful. However, a slight change was made in the number of variance pairs in the feature extraction stage. A value of nine was used instead of eight as for the previous experiment. Initially, this different setting seems somewhat suspicious because the same set of transients have been modelled and compared, with the only difference being the composition of the training set. However, further analysis shows that a slightly different PNN scaling parameter was selected because of the different training set. This, in turn, caused the PNN to react differently to the different model parameters and justifies the slight variation.

As with the previous experiment, training was only conducted to the initialization stage to prevent overtraining of the network and because further training did not improve performance. The results obtained for the validation set are listed at Appendix E of this thesis, and the confusion matrix is shown in Table 6.3. As expected, the classification ability using this training set was substantially lower with only 306/335 or 91.3% of transients classified correctly. This is, of course, attributable to the smaller size of the training set and the commensurately lower likelihood of each validation case finding a match in the training set.

Table 6.3: Confusion matrix with results from Validation Set # 2.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Class 0	38						2	
Class 1	1	37	4					
Class 2		10	32					
Class 3	5			33		2		
Class 4					41			
Class 5				4		38		
Class 6							47	1
Class 7								40

Examination of the results from this experiment shows that out of the total 29 misclassifications, only seven of them were completely wrong in terms of the manufacturer of the transmitter. Additionally, 14 of the errors were caused by confusion between the Force 2 and Force 3 transmitters. This is significant because the classification rate of the entire test set is heavily biased by a common confusion between only these two

transmitters. Equally significant is that the classification rate for two of the transmitters is perfect. Therefore, from this analysis, it can be concluded that results in favour of Class 4 or Class 7 can be accepted with much more certainty than those for Class 1 or Class 2. Transients from Class 6 also produced excellent results.

6.2.3 Training Set # 3 (Random Selection of 30)

This training set consisted of 30 transients from each class, selected randomly, leaving 175 transients in the validation set. Again, the same parameters for the segmentation were found to be most effective. Also the parameters used for feature extraction were identical to the ones selected for Training Set # 2. In this experiment, the network was initialized using the single scaling parameter as in the other two experiments. The results obtained for the validation set are listed at Appendix E to this thesis and the confusion matrix is shown in Table 6.4.

Table 6.4: Confusion matrix with results from Validation Set # 3.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Class 0	18						2	
Class 1	2	19	1					
Class 2		1	21					
Class 3				18		2		
Class 4					20			1
Class 5				1		21		
Class 6							28	
Class 7								20

The classification ability, using this validation set, was slightly lower than that of the first set with 165/175 or 94.3% of transients classified correctly. This is somewhat surprising considering that a larger training set was used. These results demonstrate that, in addition to the number of cases in the training set, the quality of the training set is also important. Ideally, the training set would be structured such that all possible patterns for each class are represented. However, despite that the best way to ensure this is to increase the size of the training set, this does not guarantee absolute success, especially in an environment where the signals contain a significant amount of noise. Therefore, since this training set is substantially different from, rather than a simple addition to the first training set, it is possible that the mixture of transients is not as comprehensive as it could be. The results from this experiment, when compared to those achieved in the first experiment, do reflect similar error patterns, though. Specifically, analysis of the validation set results at Appendix E shows that five transients misclassified in this experiment were equivalently misclassified in the first experiment. Since these same transients are consistently misclassified, it is likely that this is caused by segmentation faults where either too much noise precedes the transient or significant characteristics have been truncated. It is noteworthy that Classes 4, 6, and 7 continue to produce excellent results.

For this experiment, training the individual scaling parameters did yield better classification results from the validation set. After the network was initialized, two iterations of the conjugate gradients algorithm were performed in about 12 seconds. This brought the training set error from 16 misclassifications down to seven. However, more importantly, it improved the classification rate of the validation set to 168/175 or 96%. Specific results for this test can also be found at Appendix E. The significantly increased,

and perhaps even more invalid, confidence levels should be noted in this data for most of the test signals.

6.2.4 Rejection of Unknown Transients

An important measure of a classifier's merit, in addition to its ability to classify, is its ability to recognize when it cannot classify. Ideally, a classifier should reject any case which does not belong to any of the classes that it has trained with. Given the architecture of the PNN, where an unknown case is compared to all known cases and the closest one is automatically selected, it is impossible to have an output neuron represent the reject category. However, there are two simple mechanisms which are commonly used for dealing with this situation. The first technique has already been discussed as an automatic rejection when the distance summations of Eq. 4.11 become so high that the negative exponent goes to zero for all cases. This would only handle extreme cases though.

Alternatively, the computed Bayesian confidence measure might be used, despite that it is invalid because the training set is obviously not exhaustive. A simple threshold can be set where if the confidence is below that value, that case would be put in the reject category. Since the present implementation of TAC-MM has no built in Bayesian confidence thresholding mechanism, the user would be responsible for establishing and flagging these cases immediately after classification.

To test the Bayesian confidence thresholding mechanism, the first training set was modified such that all cases from two of the transmitters were removed. Specifically, the signals from the Yaesu 1 and Yaesu 2 transmitters were removed from the training set, but not from the validation set. The original fractal parameters established for this training set were retained and the confidence threshold for flagging a rejected case was to be set after

analysis of a significant amount of data. However, a quick scan of the classification results showed that this rejection mechanism failed miserably in this test. The PNN's winner-take-all design falsely classified every Yaesu 1 and Yaesu 2 transient with daunting conviction.

In response to the problem encountered above, a new rejection mechanism was specially designed as described in Subsection 5.1.5 of this thesis. Briefly, instead of thresholding the Bayesian confidence measure, the winning summation neuron is thresholded before normalization. Using this approach, a transient can be rejected if the activation of the winning neuron is too low, regardless of the activation level of the other neurons. To test this new technique, the same training set and parameter settings were used as described above. After the first trial, examination of the winning neuron activations in the batch classification report showed that the rejection threshold should be set at 10^{-20} . The results for the subsequent experiment are shown in Table 6.5, and detailed at Appendix E.

From Table 6.5, results for the six transients in the training set show that 176/187 or 94.1% were classified correctly. This is comparable to results achieved in the first experiment, except that three of the transients misclassified in the first experiment have now been rejected. This is especially significant because both of the transients classified in the wrong manufacturer in the first experiment class are rejected. Therefore, with the exception of the rejected cases, these results reflect perfect classification of the transients' transmitter manufacturers. More relevant to this discussion, though, is that every Yaesu transient has been appropriately banished to the reject category.

Table 6.5: Confusion matrix with results from test for rejection ability.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Reject
Class 0	27	1					2
Class 1	1	30	1				
Class 2		2	30				
Class 3				29		1	
Class 4					31		
Class 5				2		29	1
Class 6							38
Class 7							30

6.2.5 Multimodal Segmentation

In this set of experiments, the multimodal segmentation option of TAC-MM was tested. For comparison purposes, the initial experiments using the three different training sets were repeated with identical parameter settings, except that multimodal segmentation was used. The mode parameters were set for five different segmentation points, spaced linearly between thresholds of seven and eight, with a minimum separation of 35 samples between successive transient start points. In all experiments, the PNN was trained to the initialization state only.

For the first training and validation sets, results are shown in Table 6.6 and are detailed at Appendix E. Overall, significant improvement is seen with 247/255 or 96.9% correct classifications within the validation set. The confusion matrix shows that only one transient was classified outside of the correct manufacturer category. Additionally, three classes of transients were classified correctly every time.

Table 6.6: Confusion matrix from Validation Set # 1 using multimodal segmentation.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Class 0	29						1	
Class 1		31	1					
Class 2		1	31					
Class 3				28	1	1		
Class 4					31			
Class 5				3		29		
Class 6							38	
Class 7								30

For the second set, results are shown in Table 6.7 and are detailed at Appendix E. Again, significant improvement is seen over the first experiment with this training set, with correct classification of 311/335 or 92.8% of the transients in this validation set. Referring to the confusion matrix and the classification reports at Appendix E, only the single Force 1 transient is in the wrong manufacturer category again. Otherwise all transients are at least classified within the correct manufacturer. It is also significant that 16/24 misclassifications are between the Force 2 and the Force 3 transmitters.

Classification results for the third set are shown in Table 6.8 and are detailed at Appendix E. Using this training set, the PNN correctly classified 167/175 or 95.4% of the transients in Validation Set # 3. Again, the same Force 1 transient is the only one that is classified in the wrong manufacturer category. Figure 6.4 shows that this particular transient has two abnormal spikes which cause its dimension trajectory to be affected significantly. However, despite the high confidence levels shown for the incorrect classifications of this transient, individual classification trials show very low winning

Table 6.7: Confusion matrix from Validation Set # 2 using multimodal segmentation.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Class 0	38	1					1	
Class 1		31	11					
Class 2		5	37					
Class 3				37	1	2		
Class 4					41			
Class 5				2		40		
Class 6							47	1
Class 7								40

Table 6.8: Confusion matrix from Validation Set # 3 using multimodal segmentation.

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Class 0	19						1	
Class 1		21	1					
Class 2	1		21					
Class 3				16	1	3		
Class 4					21			
Class 5						22		
Class 6							27	1
Class 7								20

neuron activations in all experiments. Therefore, as discussed in the previous section, judicious selection of the rejection threshold could lessen the impact of this misclassification. Referring back to Table 6.8, it is also significant that all of the transients from three of the transmitters have been classified correctly.

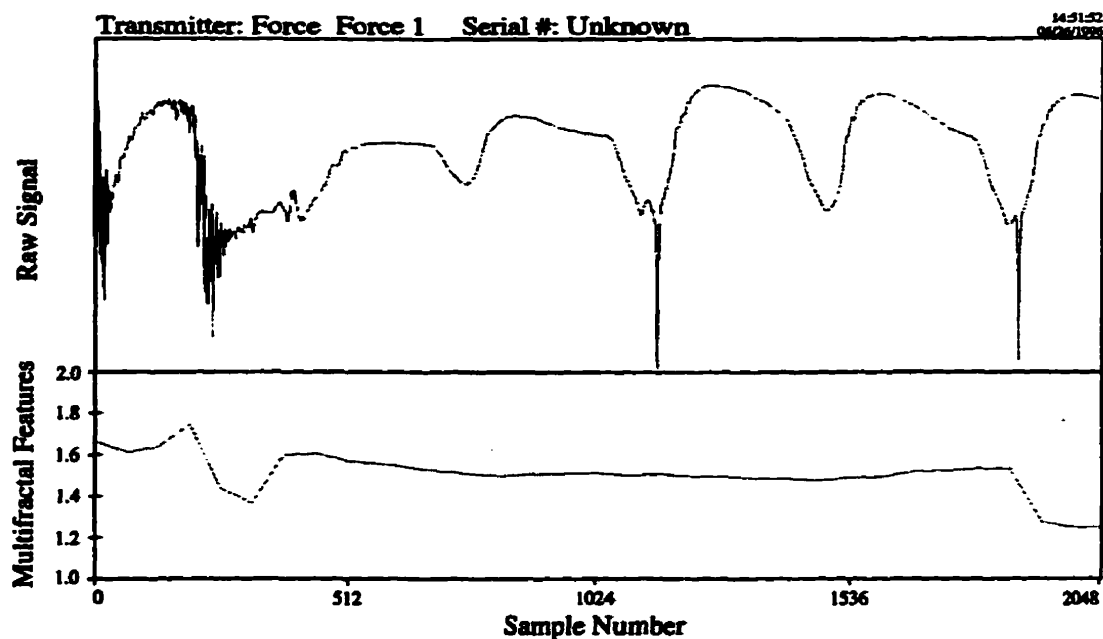


Fig. 6.4. Consistently misclassified Force 1 transient.

6.2.6 Transformation of Fractal Trajectory Model

During the testing process, an implementation error caused a transformation of the multifractal model which produced significantly improved results. Specifically the number of variance pairs was accidentally set, contrary to the limitation of Eq. 2.22, such that the spacing between compared samples could exceed the limits of the window size. In this case, the routine inadvertently set $Var(\Delta B)_k$ to zero for all values of k that caused the dyadic spacing (2^k) to exceed the window size. Then, a feature initially intended to avoid unnecessary calculations when $Var(\Delta B)_k$ equals zero, caused the program to decrement the index for the summations in the LSR algorithm. However, the multiplicative K in two terms of the LSR algorithm was not decremented. Therefore, if the number of variance pairs is set one higher than it should theoretically be, the least squares regression is transformed into

$$2H = \frac{(K+1) \sum_{i=1}^K x_i \psi_i - \left(\sum_{i=1}^K x_i \right) \left(\sum_{i=1}^K \psi_i \right)}{(K+1) \sum_{i=1}^K x_i^2 - \left(\sum_{i=1}^K x_i \right)^2} \quad (6.1)$$

which produces significantly different results than its original form (Eq 2.26). Figure 6.5 shows the difference in the calculated trajectories using this transformation. The most

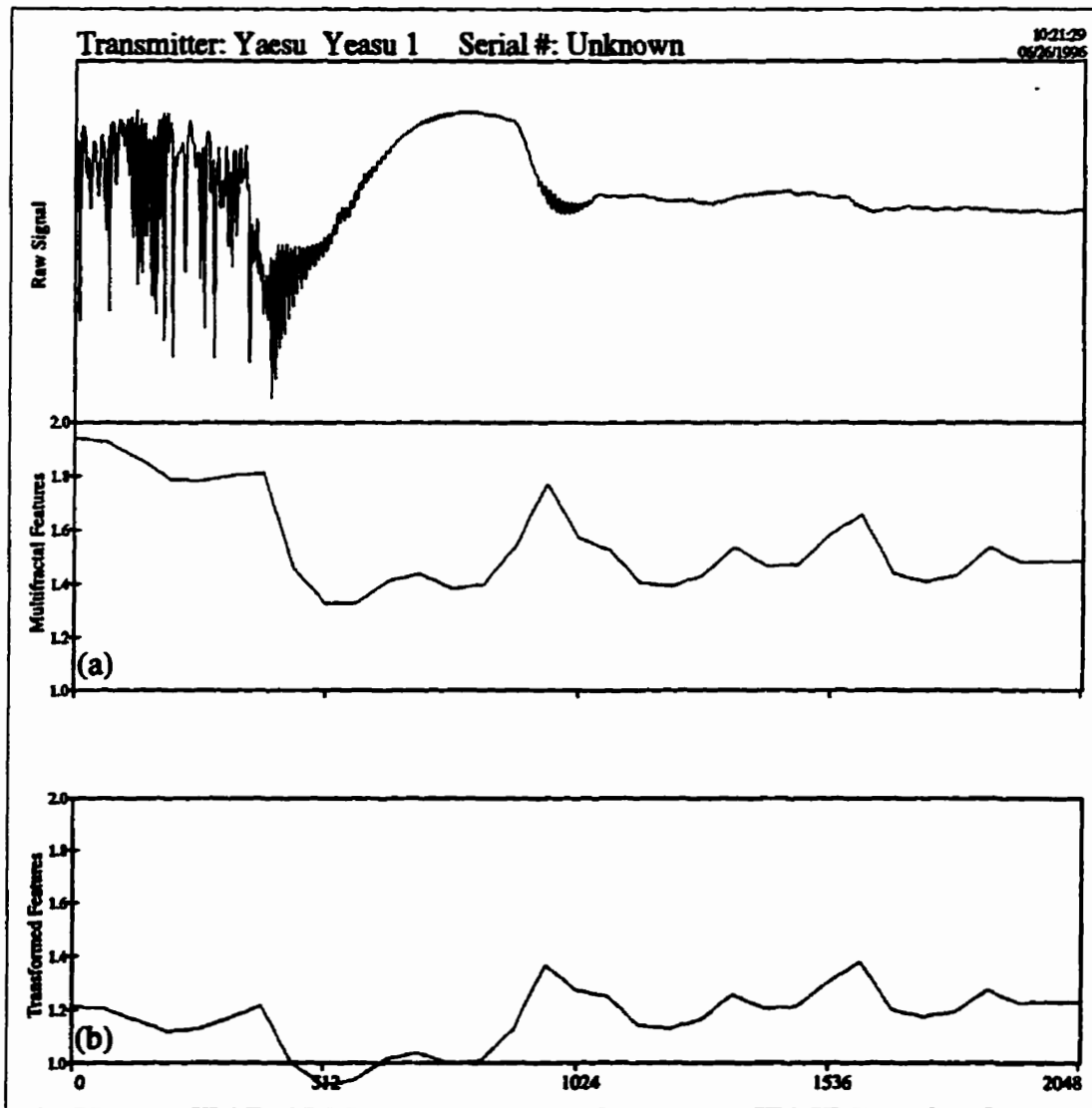


Fig. 6.5. Comparison of (a) multifractal model and (b) transformed model.

obvious difference is that the transformed dimensions are lower than the fractal dimensions. For this set of signals, the first summation in the numerator of Eq. 6.1 is much larger than the first term of the denominator. This increases the magnitude of H and, referring back to Eq. 2.20, decreases the value obtained for D_G .

The overall reduced values in the transformed model do not, however, constitute an explanation for improved classification results. Further analysis of Eq. 6.1 and Fig. 6.5 shows that the transformed model attenuates the effect of highly uncorrelated portions of the signal. Since a slight segmentation error could include irrelevant channel noise at the beginning of the transient, deliberate attenuation of this noise during modelling could definitely assist in the classification process. At the same time, while reducing the impact of irrelevant noise, it is clear that the transformed model retains the important fractal characteristics of the signal.

The actual results achieved using the transformed fractal model will now be presented. For these experiments, rather than setting the number of variance pairs to an unreasonably high number, the TAC-MM source code was modified to deliberately implement Eq. 6.1. This allows objective comparison of the modelling methods using identical parameter settings. Each of the first three experiments was repeated using identical fractal parameters and the transformed fractal model. Multimodal segmentation was not used and, again, training was only performed up to the initialization state for each experiment. As shown in Table 6.9, significantly improved results can be seen for each of the training sets.

Further testing with the transformed fractal model yielded some surprising results. It was found that a further reduction in data size could be achieved while significantly

Table 6.9: Classification results using transformed fractal model.

	Previous Results	New Results
Validation Set # 1	243/255 (95.3%)	244/255 (95.7%)
Validation Set # 2	306/335 (91.3%)	314/335 (93.7%)
Validation Set # 3	165/175 (94.3%)	170/175 (97.1%)

improving classification rates. Specifically, the three training sets were tested using the following parameter settings:

Raw File Size	8192
Transient Size	2048
Segmentation Window Size	512
Segmentation Variance Pairs	10
Segmentation Threshold	8
Feature Extraction Window Size	512
Feature Extraction Variance Pairs	4
Feature Extraction Window Shift	256

Fig. 6.6. TAC-MM parameters for testing eight element transformed model.

Notice that with the window shift parameter set to 256, the fractal model has been reduced from its original 2048 samples to just eight elements. This is a significant achievement in terms of storage requirements and PNN processing time, allowing for a much larger database of transients. The classification results achieved for these experiments are summarized in Table 6.10.

Table 6.10: Classification results using eight element model size.

	Transformed Fractal Model	Original Fractal Model
Validation Set # 1	247/255 (96.9%)	219/255 (85.9%)
Validation Set # 2	318/335 (94.9%)	267/335 (79.7%)
Validation Set # 3	169/175 (96.6%)	157/175 (89.7%)

First, it can be seen that with these parameters, the transformed fractal models perform comparably to or better than their similar, yet less compact, counterparts. This is likely attributable to the less detailed modelling of noise at the start of each transient, thus further reducing the effect of slight inconsistencies in segmentation. Therefore, these better results are achieved by masking details which are prominent, yet insignificant for classification. For comparison purposes, Table 6.10 also shows the classification results using these parameters for the original fractal modelling process. From this comparison, it is clear that the original fractal models, which tend to place more emphasis on noise, perform considerably worse than the transformed models at these parameter settings and this level of data reduction.

6.3 Confidence Measures and the PNN

Observation of the classification reports generated in Appendix E show that the confidence measures calculated by TAC-MM for this database of transients are virtually meaningless. As previously discussed, there are several reasons for this, including the theoretical limitations that none of the training set permutations are mutually exclusive or exhaustive. From a computational point of view, the contemptuously high confidence levels seen for some of the misclassifications are caused by a combination of the PNN's winner-take-all architecture and the Gaussian weighting function used in its Parzen density estimation. During training of the PNN, the scaling parameters are inclined toward becoming as small as possible, while still being able to classify similar cases, such that the total error is reduced. However, this causes the Gaussian weight functions to become very narrow and, if a case is encountered that is not close to any of the training cases, then the distance summations tend to approach zero for all possible classes. In this situation, the

summation neurons can easily adopt variations between 10^{-10} and 10^{-80} with only small differences in the Euclidean distance summations. Then, when it comes to selecting the neuron with the highest activation, the one that is selected tends to be disproportionately larger than the other neurons and a very high confidence level is produced.

The computational problem described above is an inherent limitation of the PNN architecture. It is especially prevalent when the PNN encounters cases which are unlike any of the cases in the training set. However, this is not a well-documented problem and certainly warrants further research so that a convincing solution can be found. One possible solution is to select a different weighting function than the Gaussian that is similar in shape, but does not exponentially go to zero at its extremes. This would give a better relative measure of activation between possible classes and would likely produce lower confidence levels for all decisions.

Another potential solution might be found with an approach similar to that of Subsection 6.2.4, where the absolute value of only the winning neuron was used to trigger rejection of a decision. It seems logical that this idea could be extended, for a given training set and PNN at a certain trained state, such that some sort of exponential mapping function can be determined between the activation level of the winning neuron and the coveted realm of percent confidence. This type of exponential mapping seems more logical than what is currently used and would not require changing the Gaussian weight function which otherwise produces such favourable results.

Despite that the computed confidence measures for this training set are not relevant for whatever reason, the experimental classification results themselves can be used to make a statement of confidence. A reasonable definition of confidence in this context is

the PNN's *probability of making a correct decision* [Mast93]. With this broad definition, the relative or absolute activation level of winning neurons cannot be regarded as the only possible means for measuring confidence levels. For example, consider a set of data with which, for several different permutations of training sets and validation sets, a classifier consistently performs at 95% accuracy. Thus, assuming that a similarly representative training set is always used, the probability of the PNN making a correct decision should be 0.95, with no other information given. Then, for an individual classification trial, if the activation level of the winning neuron is considered, the 95% confidence level can be further adjusted accordingly. In this context, the results achieved in this thesis show that this system is very accurate. It produces results with confidence levels between 90% and 97%, depending on the training set, the winning neuron activation, the segmentation options, and modelling technique.

A final point should be made about the transients analysed in this thesis. Observation of the ambient channel noise in the raw signals at Appendix D shows that the capturing system exhibits a nonlinear characteristic. It is possible that this nonlinearity would have an effect on the results obtained using this classification system. For this reason, every effort should be made to capture unknown transients using the same system that was used for capturing a given training set. Alternatively, and perhaps more favourably, a linear capturing system could be developed and implemented.

6.4 Summary of Chapter 6

This chapter began with a brief description of the transmitter transients used for testing this thesis. Some details of the acquisition, preprocessing, and composition of the training set were provided. Then, several tests were conducted on the data and relevant

results were presented in confusion matrices. It is significant that in every test, the system implemented by TAC-MM correctly classified over 90% of all transients in the validation sets! In a separate test, it was shown that, rather than resorting to misclassification, the PNN is highly capable of rejecting cases which are not part of the training set. Also, it was shown that the multimodal segmentation scheme can improve classification results. The final experiment showed that an interesting transformation of the fractal modelling process could simultaneously improve both the data reduction and the classification results. The chapter ended with a discussion on PNN confidence measures. Some suggestions were made for improving the standard technique and specific information was given for determining confidence measures for the results achieved in this thesis.

The next chapter presents a concise summary of conclusions, contributions, and recommendations resulting from this thesis.

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

In this thesis, a system for fast and accurate identification of radio transmitter transients was developed. Specifically, a three stage system performing segmentation, feature extraction, and classification was implemented and tested using a set of 415 recorded transmitter transients. The average processing time for completing all three stages and rendering a decision was about half of a second per transient. Experimental results showed classification rates between 90% and 97%.

The results were achieved using transients which were segmented from ambient channel noise using an effective fractal analysis technique. This analysis is able to flag transient start points earlier than conventional methods and, therefore, retains critical details which would be otherwise discarded as noise. Then, a multifractal modelling technique is effectively used to extract significant features from the transients and store them in a substantially reduced size array. The transients have been effectively reduced from 2048 samples down to 32 elements using this technique. The technique also normalizes the transients without any additional effort. Furthermore, using an interesting new transformed fractal model, data reduction down to eight elements was achieved with no loss in relevant detail.

At the backbone of this system, a PNN is used to classify the transients with a high degree of accuracy and confidence. The PNN trains with significant sized training sets in

only seconds and classifies individual cases in fractions of a second. Furthermore, for rejection of transients which are not represented in the training set, an effective mechanism has been developed by thresholding the activation level of the winning neuron. This ability contributes significantly to the total value of the classifier's decisions.

Overall, the experimental results have demonstrated that the objectives of this thesis, as listed in Section 1.2, have been achieved.

7.2 Contributions

This thesis has made the following contributions:

- a technique for segmentation of a signal from noise has been studied and refined to the point where it produces relatively consistent results;
- a highly representative, yet compact, multifractal modelling technique has been successfully implemented on a set of nonstationary transient signals;
- a PNN has been successfully implemented for accurate classification of modelled transmitter transients;
- a practical exploration of the PNN's Bayesian confidence measures has been conducted and an alternative technique, using thresholding of the winning neuron's activation, has been introduced for rejecting low confidence decisions; and
- a comprehensive and user friendly software package, TAC-MM, has been developed to implement the transient classification system on a standard desktop PC under Microsoft Windows 95.

7.3 Recommendations

Based on the research conducted in this thesis, recommendations are as follows:

- **the parameter settings and thresholding mechanism used in the segmentation stage of this system require further optimization to enable more consistent separation of a transient from ambient channel noise;**
- **a hardware or software transient acquisition system should be developed with more linear characteristics and such that the onset of a transmission is triggered directly in real time by the multifractal analysis described in this thesis;**
- **a multifractal characterization of the transients using the Rényi dimension and Mandelbrot spectrum should be explored as alternative modelling schemes;**
- **an in depth analysis and mathematical proof of the transformed fractal model discovered in this thesis should be conducted;**
- **further modifications should be made to the PNN in order to achieve more useful confidence measures as discussed in Section 6.3;**
- **a larger database of transients is required for further testing and to provide a suitable training set before this system can be put into practical use; and**
- **the software developed in this thesis is flexible enough that it can and should be tested in the processing and classification of other nonstationary signals such as speech.**

REFERENCES

- [Ande95] Darryl Anderson, "Transient signal classification using wavelet packet bases," *B. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (vi+120) pp., 1995.
- [Caco66] T. Cacoullou, "Estimation of a multivariate density," *Annals of the Institute of Statistical Mathematics* (Tokyo), Vol. 18, No. 2, pp 179-189, 1966.
- [CRC92] CRC Report, "Transmitter Signature Analysis," Ottawa, Ontario, 9 pp., 1992.
- [Diet94] James Dietrich, "A wavelet analysis of transients in phase-locked loops," *B. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (v+89) pp., 1994.
- [FeKi95] K. Ferens and W. Kinsner, "Multifractal texture classification of images," *Proceedings IEEE Wescanex '95*, (Winnipeg, MB), May 15-16, 1995; IEEE Cat No. 95CH3581-6; pp. 438-444.
- [Grie96] W. S. Grieder, "Variance fractal dimension for signal feature enhancement and segmentation from noise," *M.Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (ix+ 84) pp., 1996.
- [GrKi94] W. Grieder and W. Kinsner, "Speech segmentation by variance fractal dimension," *Proc. IEEE Can. Conf. Electrical & Computer Engineering, CCE&CE'94*; (Halifax, NS), September 25-28, 1994; IEEE Cat. No. 94TH8023; pp. 481-485.
- [Khan95] Imran Khan, "Transient analysis in frequency synthesizers," *B. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (vii+93) pp., 1995.
- [KiAr92] M. W. Kim and M. Arozullah, "Generalized probabilistic neural network-based classifiers," *International Joint Conference on Neural Networks*, Baltimore, MD, 1992.
- [Kins94a] W. Kinsner, "Fractal dimensions: Morphological , entropy, spectrum, and variance classes," *Technical Report, DEL94-4*. Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, Manitoba, Canada, (ix+137) 146 pp., May 31, 1994.

- [Kins94b] W. Kinsner, "Batch and real-time computation of a fractal dimension based on variance of a time series," *Technical Report*, DEL94-6, Department of Electrical and Computer Engineering, University of Manitoba, 22 pp., June 15, 1994.
- [Kins94c] W. Kinsner, "Fractional Brownian noise and the variance dimension," *Technical Report*, DEL94-3, Department of Electrical and Computer Engineering, University of Manitoba, (viii+78) 86 pp., April 15, 1994.
- [KiGr95] W. Kinsner and W. Grieder, "Fractal amplification of signal features using variance fractal dimension," *10th International Conference on Mathematical and Computer Modelling Record*. ICMCM'95 (Boston, MA), July 5-8, 1995.
- [Kwok95] Raymond Kwok, "High-speed capture of turn-on transients in transmitters," *B. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (x+173) pp., 1995.
- [Lang96] A. Langi, "Wavelet and Fractal Processing and Compression of Nonstationary Signals," *Ph.D. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (xviii+248) pp., 1996.
- [Marc92] M. J. Marcus, "Progress in VHF/UHF Mobile Transmitter Identification," *FCC Report*, Washington, D.C., 10 pp., 1992.
- [Mast93] T. Masters, *Practical Neural Network Recipes in C++*. San Diego, CA: Academic Press, Inc, 1993, 493 pp. ISBN 0-12-479040-2.
- [Mast95] T. Masters, *Advanced Algorithms for Neural Networks: A C++ Sourcebook*. New York, NY: John Wiley & Sons, Inc, 1995, 431 pp. ISBN 0-471-10588-0.
- [Meis72] W. S. Meisel. *Computer-oriented approaches to pattern recognition*. New York, NY: Academic Press, 1972.
- [NIST5163] U.S. National Institute of Standards and Technology, *FBI Fingerprint Study*.
- [NIST5209] U.S. National Institute of Standards and Technology, *Handprinted Digit Classifier Study*.
- [Parz62] E. Parzen, "On estimation of a probability density function and mode." *Annals of Mathematical Statistics*, 33, pp. 1065-1076, 1962.
- [Pola71] E. Polak, *Computational Methods in Optimization*, New York, NY: Academic Press, 1971.

- [PTVF92] W. Press, S Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C, Second Edition*. New York, NY: Cambridge University Press, 994 pp., 1992.
- [Rény55] A. Rényi, "On a new axiomatic theory of probability," *Acta Mathematica Hungarica* 6, pp. 285-335, 1955.
- [Ruda94] Tena Rudachek, "Phase-locked loops and their transients," *B. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (vii+57) pp., 1994.
- [RuMc86] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, vol. 1, *Foundations*. Cambridge, MA: MIT Press, 1986.
- [SCHA91] H. Schioler and U. Hartmann, "Mapping neural networks derived from the parzen window estimator," *Neural Networks*, Vol. 5, pp. 903-909, 1992.
- [Shaw94] D. B. Shaw, "Identification of Transients in Phase-Locked Loops Using Neural Networks," *B.Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, 56 pp., 1994.
- [Spec88] D. F. Specht, "Probabilistic neural networks for classification, mapping, or associative memory," *Proc. IEEE Int. Conf. Neural Networks*, Vol. 1 (San Diego, CA), pp. 525-532, July 1988.
- [Spec90a] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, Vol. 3, pp. 109-118, Jan. 1990.
- [Spec90b] D. F. Specht, "Probabilistic neural networks and the polynomial adaline as complementary techniques for classification," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp 111-121, Mar. 1990.
- [Spec91] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, Vol. 2, No. 6, pp 568-576, 1991.
- [Toon95] Jason Toonstra, "Wavelet analysis and genetic modelling of transients in radio transmitters," *B. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, (vi+147) pp., 1995.
- [Toon97] Jason Toonstra, "A radio transmitter fingerprinting system," *M. Sc. Thesis*, Department of Electrical and Computer Engineering, University of Manitoba, 178 pp., 1997.
- [Vics92] T. Vicsek. *Fractal Growth Phenoma*. Signapore: World Scientific, 1992 (2nd ed.), 488 pp. ISBN 9-810206-69-0 pbk.

APPENDIX A

TAC-MM FILE STRUCTURES

Transient Raw File (.raw) Structure

The TAC-MM software works only with files in ASCII format. The file should be arranged so that it begins with the 16-bit positive integer samples, each on its own line, from the test signal. Following that, a file trailer may contain some or all of the information shown below to identify the transmission.

30234

12453

45098

⋮

53254

27982

15274

Manufacturer

Model

Serial number

Frequency of transmission

Date

Time

Location

Coordinates

Operator

Receiver Manufacturer

Receiver Model

Receiver Serial number

Naming convention (Manual/Automatic)

Comments

Acquisition system version

Batch List File (.blf) Structure

The TAC-MM software works only with files in ASCII format. The file should be arranged so that it begins with a positive integer stating the number of files in the list. Then, the full file path for each of the transients in the list, along with the correct predicted class, should follow as shown below.

```
25
d:\Tran\force1\26fwo415.raw 0
d:\Tran\force1\26fwo422.raw 0
d:\Tran\force1\26fwo434.raw 0
d:\Tran\force1\26fwo440.raw 0
d:\Tran\force1\26fwo443.raw 0
d:\Tran\force2\26fwq123.raw 1
d:\Tran\force2\26fwq125.raw 1
d:\Tran\force2\26fwq132.raw 1
d:\Tran\force2\26fwq134.raw 1
d:\Tran\force2\26fwq142.raw 1
d:\Tran\force3\27fwi461.raw 2
d:\Tran\force3\27fwi463.raw 2
d:\Tran\force3\27fwi465.raw 2
d:\Tran\force3\27fwi471.raw 2
d:\Tran\force3\27fwi473.raw 2
d:\Tran\ken1\25fwn380.raw 3
d:\Tran\ken1\25fwn382.raw 3
d:\Tran\ken1\25fwn384.raw 3
d:\Tran\ken1\25fwn392.raw 3
d:\Tran\ken1\25fwn393.raw 3
d:\Tran\ken2\25fwn583.raw 4
d:\Tran\ken2\25fwn584.raw 4
d:\Tran\ken2\25fwn593.raw 4
d:\Tran\ken2\25fwn594.raw 4
d:\Tran\ken2\25fwo000.raw 4
```

APPENDIX B

TAC-MM SOURCE CODE

Table B-1: List of TAC-MM source files.

File Name	Page	Description
TAC_MM.h	B-5	main header file for the TAC_MM application
TAC_MM.cpp	B-6	main application source file; contains the application class CTAC_MMApp
MainFrm.h	B-9	header file for the frame class CMainFrame, which controls all SDI frame features
MainFrm.cpp	B-10	implementation file for the frame class CMainFrame
TAC_MMDoc.h	B-13	header file for document class CTAC_MMDoc, which defines and manages the data within the application
TAC_MMDoc.cpp	B-16	implementation file for the frame class CTAC_MMDoc
TAC_MMView.h	B-39	header file for view class CTAC_MMView, which defines and manages the display in the lower viewing window
TAC_MMView.cpp	B-41	implementation file for view class CTAC_MMView
RawView.h	B-50	header file for view class CRawView, which defines and manages the display in the upper viewing window
RawView.cpp	B-52	implementation file for view class CRawView
Trandata.h	B-62	header file for the class CTransientData, which is the storage class for individual transmitter transients
Trandata.cpp	B-63	implementation file for class CTransientData
Extract.h	B-65	header file for the class CExtract, which contains the segmentation and feature extraction routines
Extract.cpp	B-67	implementation file for the class CExtract

Table B-1: List of TAC-MM source files.

File Name	Page	Description
PNN.h	B-73	header file for the class CPNN, which contains all PNN processing routines
PNN.cpp	B-75	implementation file for the class CPNN
AddUnknownDialog.h	B-93	header file for the class CAddUnknownDialog, which defines and controls the dialog box displayed when the user chooses to add a recently classified transient to the database
AddUnknownDialog.cpp	B-94	implementation file for the class CAddUnknownDialog
BatchProgressDialog.h	B-95	header file for the class CBatchProgressDialog, which defines and controls the dialog box displayed to update the user on the progress of a batch classification
BatchProgressDialog.cpp	B-96	implementation file for the class CBatchProgressDialog
ClassifyDialog.h	B-98	header file for the class CClassifyDialog, which defines and controls the dialog box displayed to show the results of a classification
ClassifyDialog.cpp	B-99	implementation file for the class CClassifyDialog
EditFPProgressDialog.h	B-100	header file for the class CEditFPProgressDialog, which defines and controls the dialog box displayed to update the user on the progress of a change in fractal parameters
EditFPProgressDialog.cpp	B-101	implementation file for the class CEditFPProgressDialog
EnterClassDialog.h	B-103	header file for the class CEnterClassDialog, which defines and controls the dialog box displayed to enter the transmitter class integer of a transient being added to the database
EnterClassDialog.cpp	B-104	implementation file for the class CEnterClassDialog
EnterSigmaInitParamsDialog.h	B-105	header file for the class CEnterSigmaInitParamsDialog, which defines and controls the dialog box displayed to enter the parameters required for the sigma initialization

Table B-1: List of TAC-MM source files.

File Name	Page	Description
EnterSigmaInitParams-Dialog.cpp	B-106	implementation file for the class CEnterSigmaInitParamsDialog
FractalParamDialog.h	B-107	header file for the class CFractalParamDialog, which defines and controls the dialog box displayed to change the fractal parameters
FractalParamDialog.cpp	B-108	implementation file for the class CFractalParamDialog
ModeParamsDialog.h	B-111	header file for the class CModeParamsDialog, which defines and controls the dialog box displayed to change the segmentation mode parameters
ModeParamsDialog.cpp	B-112	implementation file for the class CModeParamsDialog
NewDatabase.h	B-114	header file for the class CNewDataBase, which defines and controls the dialog box displayed to select parameters for a new database
NewDatabase.cpp	B-115	implementation file for the class CNewDataBase
SearchDialog.h	B-117	header file for the class CSearchDialog, which defines and controls the dialog box displayed to search for a specific transient in the database
SearchDialog.cpp	B-118	implementation file for the class CSearchDialog
SetRejectionDialog.h	B-121	header file for the class CSetRejectionDialog, which defines and controls the dialog box displayed to set the rejection threshold
SetRejectionDialog.cpp	B-122	implementation file for the class CSetRejectionDialog
SigmaInitDialog.h	B-123	header file for the class CSigmaInitDialog, which defines and controls the dialog box displayed to update the user on the status of sigma initialization
SigmaInitDialog.cpp	B-124	implementation file for the class CSigmaInitDialog
TrainSigmasOptDialog.h	B-127	header file for the class CTrainSigmasOptDialog, which defines and controls the dialog box displayed to update the user on the status of sigma optimization

Table B-1: List of TAC-MM source files.

File Name	Page	Description
TrainSigmasOptDialog.cpp	B-128	implementation file for the class CTrainSigmasOptDialog
TrainingResults.h	B-131	header file for the class CTrainingResults, which is the storage class for the results obtained during training or from batch classification
TrainingResults.cpp	B-132	implementation file for the class CTrainingResults
StdAfx.h	B-133	header file for system include files, that are used frequently, but are changed infrequently
resource.h	B-134	this is a standard header file, which defines new resource IDs for the Microsoft Developer Studio
TAC_MM.rc	B-137	this is a listing of all the Microsoft Windows resources that the program uses

TAC-MM.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// TAC_MM.h : main header file for the TAC_MM application
//
#if !defined(AFX_TAC_MM_H_778DC4E5_CDEC_11D0_BF54_444553540000_INCLUDED_)
#define AFX_TAC_MM_H_778DC4E5_CDEC_11D0_BF54_444553540000_INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#ifdef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h" // main symbols
////////////////////////////////////
// CTAC_MMApp:
// See TAC_MM.cpp for the implementation of this class
//
class CTAC_MMApp : public CWinApp
{
public:
    CTAC_MMApp();
// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CTAC_MMApp)
public:
    virtual BOOL InitInstance();
    }}AFX_VIRTUAL
// Implementation
    {{{AFX_MSG(CTAC_MMApp)
afx_msg void OnAppAbout();
    // NOTE - the ClassWizard will add and remove member functions here.
    // DO NOT EDIT what you see in these blocks of generated code !
    }}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
////////////////////////////////////
```


TAC-MM.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// TAC_MM.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "MainFrm.h"
#include "Trandata.h"
#include "TAC_MMDoc.h"
#include "TAC_MMView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CTAC_MMApp
BEGIN_MESSAGE_MAP(CTAC_MMApp, CWinApp)
    //{{AFX_MSG_MAP(CTAC_MMApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        // DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
    // Standard print setup command
    ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()
////////////////////////////////////
// CTAC_MMApp construction
CTAC_MMApp::CTAC_MMApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}
////////////////////////////////////
// The one and only CTAC_MMApp object
CTAC_MMApp theApp;
////////////////////////////////////
// CTAC_MMApp initialization
BOOL CTAC_MMApp::InitInstance()
{
    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.
#ifdef _AFXDLL
    Enable3dControls(); // Call this when using MFC in a shared DLL
#else
```

```

    Enable3dControlsStatic();    // Call this when linking to MFC statically
#endif
    LoadStdProfileSettings();    // Load standard INI file options (including MRU)
    // Register the application's document templates. Document templates
    // serve as the connection between documents, frame windows and views.
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CTAC_MMDoc),
        RUNTIME_CLASS(CMainFrame),    // main SDI frame window
        RUNTIME_CLASS(CTAC_MMView));
    AddDocTemplate(pDocTemplate);
    // Enable DDE Execute open
    EnableShellOpen();
    RegisterShellFileTypes(TRUE);
    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);
    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;
    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();
    m_pMainWnd->SetWindowText("TAC-MM");

    return TRUE;
}
////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();
    // Dialog Data
    //({AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //})AFX_DATA
    // ClassWizard generated virtual function overrides
    //({AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //})AFX_VIRTUAL

    // Implementation
    protected:
    //({AFX_MSG(CAboutDlg)
    // No message handlers
    //})AFX_MSG
    DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //({AFX_DATA_INIT(CAboutDlg)
    //})AFX_DATA_INIT
}

```

```
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAboutDlg)
   //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
   //{{AFX_MSG_MAP(CAboutDlg)
    // No message handlers
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command to run the dialog
void CTAC_MMAApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
////////////////////////////////////
// CTAC_MMAApp commands
```

MainFrm.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```

// MainFrm.h : interface of the CMainFrame class
//
////////////////////////////////////////////////////////////////////
class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)
// Attributes
protected:
    CSplitterWnd m_wndSplitter;
public:
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
public:
    virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext);
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
    CStatusBar m_wndStatusBar;
protected: // control bar embedded members

    CToolBar m_wndToolBar;
// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnViewSegmentation();
    afx_msg void OnViewFractaltrajectory();
    afx_msg void OnViewRawsignal();
    afx_msg void OnViewTransient();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
//////////////////////////////////////////////////////////////////

```

MainFrm.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```

// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "MainFrm.h"
#include "Trandata.h"
#include "TAC_MMDoc.h"
#include "TAC_MMView.h"
#include "RawView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
   //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    ON_COMMAND(ID_VIEW_SEGMENTATION, OnViewSegmentation)
    ON_COMMAND(ID_VIEW_FRACTALTRAJECTORY, OnViewFractaltrajectory)
    ON_COMMAND(ID_VIEW_RAWSIGNAL, OnViewRawsignal)
    ON_COMMAND(ID_VIEW_TRANSIENT, OnViewTransient)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()
static UINT indicators[] =
{
    ID_SEPARATOR,      // status line indicator
    ID_SEPARATOR,
};
////////////////////////////////////
// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
}
CMainFrame::~CMainFrame()
{
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndToolBar.Create(this) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {

```

```

        TRACE0("Failed to create toolbar\n");
        return -1;    // fail to create
    }
    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
            sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;    // fail to create
    }
    // TODO: Remove this if you don't want tool tips or a resizable toolbar
    m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
    // TODO: Delete these three lines if you don't want the toolbar to
    // be dockable
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);
    return 0;
}
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
    CCreateContext* pContext)
{
    VERIFY(m_wndSplitter.CreateStatic(this, 2, 1));
    VERIFY(m_wndSplitter.CreateView(0, 0, RUNTIME_CLASS(CRawView),
        CSize(500, 500), pContext));
    VERIFY(m_wndSplitter.CreateView(1, 0, RUNTIME_CLASS(CTAC_MMView),
        CSize(100, 100), pContext));
    return TRUE;
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs
    return CFrameWnd::PreCreateWindow(cs);
}
////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif // _DEBUG
////////////////////////////////////
// CMainFrame message handlers
void CMainFrame::OnViewSegmentation()
{
    m_wndSplitter.SetActivePane(0, 0, NULL);
}

```

```
void CMainFrame::OnViewFractaltrajectory()
{
    m_wndSplitter.SetActivePane(0, 0, NULL);
}
void CMainFrame::OnViewRawsignal()
{
    m_wndSplitter.SetActivePane(0, 0, NULL);
}
void CMainFrame::OnViewTransient()
{
    m_wndSplitter.SetActivePane(0, 0, NULL);
}
```

TAC_MMDoc.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// TAC_MMDoc.h : interface of the CTAC_MMDoc class
```

```
//
```

```
////////////////////////////////////
```

```
#include "Extract.h"
```

```
#include "PNN.h"
```

```
#include "EditFPProgressDialog.h"
```

```
#include "TrainingResults.h"
```

```
#include <afx.h> //for CTime class
```

```
class CTAC_MMDoc : public CDocument
```

```
{
```

```
private:
```

```
    CTransientArray m_TransientArray;
```

```
    CExtract m_Extract;
```

```
    CPNN m_PNN;
```

```
    CResultsArray m_ResultsArray;
```

```
    int m_nCurrentTransient;
```

```
    int *m_nRawSignal;
```

```
    int m_nHighestValue;
```

```
    int m_nLowestValue;
```

```
    int m_nTransientBeginsHere;
```

```
    double m_dRejectThreshold;
```

```
    int m_nNumClassifyModes;
```

```
    int m_nMinSeparation;
```

```
    double *m_dThresholds;
```

```
    double *m_dVarDimTraj;
```

```
    double *m_dTransientModel;
```

```
    BOOL m_bDocCreated;
```

```
    CString m_TransmitterMake;
```

```
    CString m_TransmitterModel;
```

```
    CString m_TransmitterSerial;
```

```
    CString m_Date;//mm/dd/yy
```

```
    CString m_Time;
```

```
    CPageSetupDialog m_PSDdlg;
```

```
protected: // create from serialization only
```

```
    CTAC_MMDoc();
```

```
    DECLARE_DYNCREATE(CTAC_MMDoc)
```

```
// Attributes
```

```
public:
```

```
    BOOL m_bFileInMemory;
```

```
    BOOL m_bViewRawSignalFlag;
```

```
    BOOL m_bViewVarDimTrajFlag;
```

```
    BOOL m_bScrollToTransient;
```

```
    BOOL m_bViewSegmentation;
```

```
    BOOL m_bNotClassifying;
```

```
    BOOL m_bViewTrainingResults;
```

```
    BOOL m_bResultsExist;
```

```
    BOOL m_bTrainingResults;
```

```
    BOOL m_bInBatchProcess;
```



```

int *GetRawSignal() { return m_nRawSignal; }
double *GetVarDimTraj() { return m_dVarDimTraj; }
int GetLowestValue() { return m_nLowestValue; }
int GetHighestValue() { return m_nHighestValue; }
int GetTransientBeginsHere() { return m_nTransientBeginsHere; }
CTransientArray *GetArray() { return &m_TransientArray; }
CExtract *GetExtract() { return &m_Extract; }
CPageSetupDialog *GetPageSetupDlg() { return &m_PSDdlg; }
CResultsArray *GetResultsArray() { return &m_ResultsArray; }
// Operations
public:
    BOOL ReadRawSignal(CString FilePath, BOOL IsAdding); //FALSE if file not good
    int GetCurrentTransient() { return m_nCurrentTransient; }
    void SetCurrentTransient(int Place) { m_nCurrentTransient=Place; }
    void IncrementCurrentTransient() { m_nCurrentTransient+=1;
                                     UpdateStatusBar(); }
    void DecrementCurrentTransient() { m_nCurrentTransient-=1;
                                     if(m_nCurrentTransient!=-1)
                                         UpdateStatusBar();
                                     else
                                         UpdateStatusBarNoTransients();}

    void UpdateStatusBar();
    void UpdateStatusBarNoTransients();
    void UpdateStatusBarUnknown();
    void ViewNeedsUpdating();
    int FindNumClasses();
    void SegmentAllTransients(int NewModelSize,
                              CEditFPProgressDialog *ProgressDlg);
    void ComputeAllModels(int NewModelSize, CEditFPProgressDialog *ProgressDlg);
    int SearchForClass(int Class);
    int SearchForMake(CString Make);
    int SearchForModel(CString Model);
    int SearchForSerial(CString Serial);
    int SearchForDate(CString Date);
    int SearchForTime(CString TTime);
    void DeleteResultsArray();
    void CheckWindowsMessages(int NumChecks);
    double ComputeConfidence(double *SummationNeuron, int PredictedClass);
    int CTAC_MMDoc::MultiModeClassify(double *HighestConfidence,
                                       double *Activation);
// Overrides
    // ClassWizard generated virtual function overrides
    //{ {AFX_VIRTUAL(CTAC_MMDoc)
    public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        virtual void DeleteContents();
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CTAC_MMDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;

```

```

#endif
protected:
// Generated message map functions
protected:
    //{{AFX_MSG(CTAC_MMDoc)
    afx_msg void OnFileNew();
    afx_msg void OnDatabaseAddtransient();
    afx_msg void OnUpdateDatabaseAddtransient(CCmdUI* pCmdUI);
    afx_msg void OnDatabaseDeletetransient();
    afx_msg void OnUpdateDatabaseDeletetransient(CCmdUI* pCmdUI);
    afx_msg void OnViewPrev();
    afx_msg void OnUpdateViewPrev(CCmdUI* pCmdUI);
    afx_msg void OnViewNext();
    afx_msg void OnUpdateViewNext(CCmdUI* pCmdUI);
    afx_msg void OnViewRawsignal();
    afx_msg void OnViewFractaltrajectory();
    afx_msg void OnUpdateViewFractaltrajectory(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewRawsignal(CCmdUI* pCmdUI);
    afx_msg void OnFilePageSetup();
    afx_msg void OnUpdateFilePageSetup(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFilePrint(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFilePrintPreview(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFileSave(CCmdUI* pCmdUI);
    afx_msg void OnNeuralnetClassify();
    afx_msg void OnUpdateNeuralnetClassify(CCmdUI* pCmdUI);
    afx_msg void OnNeuralnetInitializesigmas();
    afx_msg void OnUpdateNeuralnetInitializesigmas(CCmdUI* pCmdUI);
    afx_msg void OnEditFractalparameters();
    afx_msg void OnUpdateEditFractalparameters(CCmdUI* pCmdUI);
    afx_msg void OnViewSearch();
    afx_msg void OnUpdateViewSearch(CCmdUI* pCmdUI);
    afx_msg void OnNeuralnetOptimizesigmas();
    afx_msg void OnUpdateNeuralnetOptimizesigmas(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewZoomedrawsignal(CCmdUI* pCmdUI);
    afx_msg void OnUpdateViewTrainingresults(CCmdUI* pCmdUI);
    afx_msg void OnViewZoomedrawsignal();
    afx_msg void OnViewTrainingresults();
    afx_msg void OnNeuralnetBatchclassify();
    afx_msg void OnUpdateNeuralnetBatchclassify(CCmdUI* pCmdUI);
    afx_msg void OnNeuralnetModeparameters();
    afx_msg void OnUpdateNeuralnetModeparameters(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFileNew(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFileOpen(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFileSaveAs(CCmdUI* pCmdUI);
    afx_msg void OnNeuralnetSetrejectionthreshold();
    afx_msg void OnUpdateNeuralnetSetrejectionthreshold(CCmdUI* pCmdUI);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
////////////////////////////////////

```

TAC_MMDoc.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// TAC_MMDoc.cpp : implementation of the CTAC_MMDoc class
//
#include "stdafx.h"
#include <fstream.h>
#include "TAC_MM.h"
#include "MainFrm.h"
#include "Trandata.h"
// #include "Extract.h" declared in TAC_MMDoc.h
#include "TAC_MMDoc.h"
#include "NewDatabase.h"
#include "EnterClassDialog.h"
#include "ClassifyDialog.h"
#include "AddUnknownDialog.h"
#include "SigmaInitDialog.h"
#include "EnterSigmaInitParamsDialog.h"
#include "TrainSigmasOptDialog.h"
#include "FractalParamDialog.h"
#include "SearchDialog.h"
#include "BatchProgressDialog.h"
#include "ModeParamsDialog.h"
#include "SetRejectionDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CTAC_MMDoc
IMPLEMENT_DYNCREATE(CTAC_MMDoc, CDocument)
BEGIN_MESSAGE_MAP(CTAC_MMDoc, CDocument)
    //{{AFX_MSG_MAP(CTAC_MMDoc)
    ON_COMMAND(ID_FILE_NEW, OnFileNew)
    ON_COMMAND(ID_DATABASE_ADDTRANSIENT, OnDatabaseAddtransient)
    ON_UPDATE_COMMAND_UI(ID_DATABASE_ADDTRANSIENT,
OnUpdateDatabaseAddtransient)
    ON_COMMAND(ID_DATABASE_DELETETRANSIENT, OnDatabaseDeletetransient)
    ON_UPDATE_COMMAND_UI(ID_DATABASE_DELETETRANSIENT,
OnUpdateDatabaseDeletetransient)
    ON_COMMAND(ID_VIEW_PREV, OnViewPrev)
    ON_UPDATE_COMMAND_UI(ID_VIEW_PREV, OnUpdateViewPrev)
    ON_COMMAND(ID_VIEW_NEXT, OnViewNext)
    ON_UPDATE_COMMAND_UI(ID_VIEW_NEXT, OnUpdateViewNext)
    ON_COMMAND(ID_VIEW_RAWSIGNAL, OnViewRawsignal)
    ON_COMMAND(ID_VIEW_FRACTALTRAJECTORY, OnViewFractaltrajectory)
    ON_UPDATE_COMMAND_UI(ID_VIEW_FRACTALTRAJECTORY,
OnUpdateViewFractaltrajectory)
    ON_UPDATE_COMMAND_UI(ID_VIEW_RAWSIGNAL, OnUpdateViewRawsignal)
    ON_COMMAND(ID_FILE_PAGE_SETUP, OnFilePageSetup)
```

```

ON_UPDATE_COMMAND_UI(ID_FILE_PAGE_SETUP, OnUpdateFilePageSetup)
ON_UPDATE_COMMAND_UI(ID_FILE_PRINT, OnUpdateFilePrint)
ON_UPDATE_COMMAND_UI(ID_FILE_PRINT_PREVIEW, OnUpdateFilePrintPreview)
ON_UPDATE_COMMAND_UI(ID_FILE_SAVE, OnUpdateFileSave)
ON_COMMAND(ID_NEURALNET_CLASSIFY, OnNeuralnetClassify)
ON_UPDATE_COMMAND_UI(ID_NEURALNET_CLASSIFY, OnUpdateNeuralnetClassify)
ON_COMMAND(ID_NEURALNET_INITIALIZESIGMAS, OnNeuralnetInitializesigmas)
ON_UPDATE_COMMAND_UI(ID_NEURALNET_INITIALIZESIGMAS,
OnUpdateNeuralnetInitializesigmas)
ON_COMMAND(ID_EDIT_FRACTALPARAMETERS, OnEditFractalparameters)
ON_UPDATE_COMMAND_UI(ID_EDIT_FRACTALPARAMETERS,
OnUpdateEditFractalparameters)
ON_COMMAND(ID_VIEW_SEARCH, OnViewSearch)
ON_UPDATE_COMMAND_UI(ID_VIEW_SEARCH, OnUpdateViewSearch)
ON_COMMAND(ID_NEURALNET_OPTIMIZESIGMAS, OnNeuralnetOptimizesigmas)
ON_UPDATE_COMMAND_UI(ID_NEURALNET_OPTIMIZESIGMAS,
OnUpdateNeuralnetOptimizesigmas)
ON_UPDATE_COMMAND_UI(ID_VIEW_ZOOMEDRAWSIGNAL,
OnUpdateViewZoomedrawsignal)
ON_UPDATE_COMMAND_UI(ID_VIEW_TRAININGRESULTS, OnUpdateViewTrainingresults)
ON_COMMAND(ID_VIEW_ZOOMEDRAWSIGNAL, OnViewZoomedrawsignal)
ON_COMMAND(ID_VIEW_TRAININGRESULTS, OnViewTrainingresults)
ON_COMMAND(ID_NEURALNET_BATCHCLASSIFY, OnNeuralnetBatchclassify)
ON_UPDATE_COMMAND_UI(ID_NEURALNET_BATCHCLASSIFY,
OnUpdateNeuralnetBatchclassify)
ON_COMMAND(ID_NEURALNET_MODEPARAMETERS, OnNeuralnetModeparameters)
ON_UPDATE_COMMAND_UI(ID_NEURALNET_MODEPARAMETERS,
OnUpdateNeuralnetModeparameters)
ON_UPDATE_COMMAND_UI(ID_FILE_NEW, OnUpdateFileNew)
ON_UPDATE_COMMAND_UI(ID_FILE_OPEN, OnUpdateFileOpen)
ON_UPDATE_COMMAND_UI(ID_FILE_SAVE_AS, OnUpdateFileSaveAs)
ON_COMMAND(ID_NEURALNET_SETREJECTIONTHRESHOLD,
OnNeuralnetSetrejectionthreshold)
ON_UPDATE_COMMAND_UI(ID_NEURALNET_SETREJECTIONTHRESHOLD,
OnUpdateNeuralnetSetrejectionthreshold)
//}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CTAC_MMDoc construction/destruction
CTAC_MMDoc::CTAC_MMDoc()
{
    m_bFileInMemory=FALSE;
    m_bDocCreated=FALSE;
    m_nCurrentTransient=-1;
    m_bViewRawSignalFlag=TRUE;
    m_bViewVarDimTrajFlag=TRUE;
    m_bScrollToTransient=FALSE;
    m_bViewSegmentation=TRUE;
    m_bNotClassifying=TRUE;
    m_bViewTrainingResults=FALSE; //View Raw signal by default
    m_bResultsExist=FALSE;
    m_bInBatchProcess=FALSE;
    m_PNN.SetTransientArrayPointer(&m_TransientArray);
    m_nNumClassifyModes=1;
}

```

```

m_dRejectThreshold=0.000000;
//set default printer margins, paper size, orientation
m_PSDdlg.m_psd.Flags=m_PSDdlg.m_psd.Flags|PSD_RETURNDEFAULT;
m_PSDdlg.m_psd.rMargin.top=1000;
m_PSDdlg.m_psd.rMargin.bottom=1000;
m_PSDdlg.m_psd.rMargin.left=1000;
m_PSDdlg.m_psd.rMargin.right=1000;
m_PSDdlg.DoModal();           //dlg not displayed cuz of PSD_RETURNDEFAULT

(m_PSDdlg.GetDevMode()->dmOrientation=DMORIENT_LANDSCAPE;
(m_PSDdlg.GetDevMode()->dmPaperSize=DMPAPER_LETTER;
m_PSDdlg.DoModal();
m_PSDdlg.m_psd.Flags=m_PSDdlg.m_psd.Flags^PSD_RETURNDEFAULT; //^is XOR
AfxGetApp()->SelectPrinter(m_PSDdlg.m_psd.hDevNames,
    m_PSDdlg.m_psd.hDevMode, TRUE);
}
CTAC_MMDoc::~CTAC_MMDoc()
{
    CPageSetupDialog Temp;//select default: printer before exiting
    AfxGetApp()->SelectPrinter(Temp.m_psd.hDevNames,
        Temp.m_psd.hDevMode, FALSE);
}
BOOL CTAC_MMDoc::OnNewDocument()
{
    TRACE("Don, OnNewDocument being executed.\n");
    if (!CDocument::OnNewDocument())
        return FALSE;
    // TODO: add reinitialization code here
    // (SDI documents will reuse this document)

    return TRUE;
}
////////////////////////////////////
// CTAC_MMDoc serialization
void CTAC_MMDoc::Serialize(CArchive& ar)
{
    m_Extract.Serialize(ar);
    m_TransientArray.Serialize(ar);
    m_PNN.Serialize(ar);

    if (ar.IsStoring())
    {
        ar << m_nCurrentTransient;
    }
    else
    {
        ar >> m_nCurrentTransient;//position at transient at last save
        m_nRawSignal=new int[m_Extract.GetRawFileSize()];           //mem for raw signal
        m_dVarDimTraj=new double[m_Extract.GetRawFileSize()];
        m_dTransientModel=new double[m_Extract.GetTransientModelSize()];
        m_dThresholds=new double[1];                               //dummy new so that delete [] m_dThresholds
                                                                //doesn't fail when first called

        m_bDocCreated=TRUE;
        if(m_TransientArray.GetSize(>0) {

```

```

        m_bFileInMemory=TRUE;
//      UpdateStatusBar();
        ViewNeedsUpdating();
    }
    else {
        m_nCurrentTransient=-1;
        UpdateStatusBarNoTransients();
        UpdateAllViews(NULL);
    }
}
}
////////////////////////////////////////////////////////////////////
// CTAC_MMDoc diagnostics
#ifdef _DEBUG
void CTAC_MMDoc::AssertValid() const
{
    CDocument::AssertValid();
}
void CTAC_MMDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif // _DEBUG
////////////////////////////////////////////////////////////////////
// CTAC_MMDoc commands
void CTAC_MMDoc::OnFileNew()
{
    TRACE("Don, OnFileNew being executed.\n");
    CNewDatabase NewDatabaseDlg; //declare dialog class object
    NewDatabaseDlg.m_RawFileSize=8192; //set defaults
    NewDatabaseDlg.m_TransientSize=2048;
    NewDatabaseDlg.m_WindowSize=256; //Segmentation window size
    NewDatabaseDlg.m_VariancePairs=5; //Segmentation variance pairs
    NewDatabaseDlg.m_Threshold=15.0;
    NewDatabaseDlg.m_ModelWindowSize=512;
    NewDatabaseDlg.m_ModelVariancePairs=5;
    NewDatabaseDlg.m_ModelWindowShift=16;

    if (NewDatabaseDlg.DoModal()==IDOK) {

        CTAC_MMDoc::OnNewDocument();
        m_Extract.SetParams(NewDatabaseDlg.m_RawFileSize,
            NewDatabaseDlg.m_TransientSize,
            NewDatabaseDlg.m_WindowSize,
            NewDatabaseDlg.m_VariancePairs,
            NewDatabaseDlg.m_Threshold,
            NewDatabaseDlg.m_ModelWindowSize,
            NewDatabaseDlg.m_ModelVariancePairs,
            NewDatabaseDlg.m_ModelWindowShift);
        m_nRawSignal=new int[NewDatabaseDlg.m_RawFileSize]; //mem for raw signal
        m_dVarDimTraj=new double[NewDatabaseDlg.m_RawFileSize];
        m_dTransientModel=new double[m_Extract.GetTransientModelSize()];
        m_dThresholds=new double[1]; //dummy new so that delete [] m_dThresholds
            //doesn't fail when first called
    }
}

```

```

    m_PNN.Initialize(m_Extract.GetTransientModelSize());
    m_nCurrentTransient=-1;
    m_bDocCreated=TRUE;
    UpdateStatusBarNoTransients();
    UpdateAllViews(NULL);
}
}
void CTAC_MMDoc::DeleteContents()
{
    // TODO: Add your specialized code here and/or call the base class
    TRACE("Don, DeleteContents being executed.\n");

    if(m_bDocCreated) {
        delete [] m_nRawSignal;
        delete [] m_dVarDimTraj;
        delete [] m_dTransientModel;
        delete [] m_dThresholds;
        m_bFileInMemory=FALSE;
        m_bDocCreated=FALSE;
    }
    int ctr=m_TransientArray.GetSize();
    while (ctr--) {
        delete m_TransientArray.GetAt(ctr);
    }
    m_TransientArray.RemoveAll();

    DeleteResultsArray();
    CDocument::DeleteContents();
    UpdateAllViews(NULL);
}
void CTAC_MMDoc::OnDatabaseAddtransient()
{
    CString PathName;
    int TransientModelSize=m_Extract.GetTransientModelSize();
    CFileDialog FileDlg(TRUE, "raw", "*."); //TRUE means file open,
    FileDlg.m_ofn.lpstrTitle="Add Transient: Select a File";

    if (FileDlg.DoModal()==IDOK) {
        HCURSOR Wait=AfxGetApp()->LoadStandardCursor(IDC_WAIT);
        ::SetCursor(Wait);
        PathName=FileDlg.GetPathName();
        if (ReadRawSignal(PathName,TRUE)) { //sees if file exists, is compatible
            m_bInBatchProcess=TRUE; // disables all menu functions
            m_nTransientBeginsHere=m_Extract.FindTransient(m_nRawSignal,
                m_dVarDimTraj, m_dTransientModel);
            m_bInBatchProcess=FALSE;
            if (m_nTransientBeginsHere==1) {
                AfxMessageBox("No transient found. Try adjusting settings."
                    ,MB_OK);
            }
        }
        else {
            CTransientData *TransientData=
                new CTransientData(TransientModelSize);
            TransientData->SetRawFilePath(PathName);
        }
    }
}

```

```

    TransientData->SetTransientBeginsHere(m_nTransientBeginsHere);
    TransientData->SetTransmitterMake(m_TransmitterMake);
    TransientData->SetTransmitterModel(m_TransmitterModel);
    TransientData->SetTransmitterSerial(m_TransmitterSerial);
    TransientData->SetTransmitDate(m_Date);
    TransientData->SetTransmitTime(m_Time);
    TransientData->SetTransientModelSize(TransientModelSize);
    TransientData->SetTransientModel(m_dTransientModel);
    TransientData->SetTransientClass(-1);
    m_TransientArray.InsertAt(m_nCurrentTransient+1, TransientData);
    m_bFileInMemory=TRUE;
    IncrementCurrentTransient();
    m_bScrollToTransient=TRUE;
    m_bViewSegmentation=TRUE;           //show tran Segmentation info when adding
    UpdateAllViews(NULL);
    CEnterClassDialog EnterClassDlg;
    EnterClassDlg.DoModal();
    m_TransientArray.GetAt(m_nCurrentTransient)
        ->SetTransientClass(EnterClassDlg.m_TransientClass);
    SetModifiedFlag(TRUE);
    UpdateStatusBar();
    m_PNN.SetNumCases(m_TransientArray.GetSize());
    m_PNN.SetNumClasses(FindNumClasses());
}
}
HCURSOR Arrow=AfxGetApp()->LoadStandardCursor(IDC_ARROW);
::SetCursor(Arrow);
}
}
BOOL CTAC_MMDoc::ReadRawSignal(CString FilePath, BOOL IsAdding)
{
    int RawFileSize=m_Extract.GetRawFileSize();
    int Temp=-1;

    ifstream tfile(FilePath, ios::nocreate);
    if (tfile.fail() {
        AfxMessageBox("Error opening file.",MB_OK);
        return FALSE;
    }
    m_nHighestValue=0;
    m_nLowestValue=65535;
    for(int ctr=0; ctr<RawFileSize; ctr++) {
        tfile >> Temp;
        if (Temp<0||Temp>65536||tfile.eof()) {
            AfxMessageBox("Incompatibe file format.",MB_OK);
            return FALSE;
        }
        if(Temp>m_nHighestValue) {
            m_nHighestValue=Temp;
        }
        else if(Temp<m_nLowestValue) {
            m_nLowestValue=Temp;
        }
        m_nRawSignal[ctr]=Temp;
    }
}

```



```

        Temp=-1;
    }
    if(IsAdding) {
        char temp[50];
        // Temp=65000;
        // while(Temp>=10000&&Temp<=65536) {
        //     tfile >> Temp;
        // }
        tfile.getline(temp,50,'\n');
        tfile.getline(temp,50,'\n');
        m_TransmitterMake=temp;
        tfile.getline(temp,50,'\n');
        m_TransmitterModel=temp;
        tfile.getline(temp,50,'\n');
        m_TransmitterSerial=temp;
        tfile.getline(temp,50,'\n');
        tfile.getline(temp,50,'\n');
        m_Date=temp;
        tfile.getline(temp,50,'\n');
        m_Time=temp;
    }
    tfile.close;
    return TRUE;
}
void CTAC_MMDoc::UpdateStatusBar()
{
    CString str;
    CMainFrame *pFrame=(CMainFrame*) AfxGetApp()->m_pMainWnd;
    CStatusBar *pStatus=&pFrame->m_wndStatusBar;
    if (pStatus) {
        str.Format("Transient #%d of %d.  Class: %d", (m_nCurrentTransient+1),
            m_TransientArray.GetSize(),
            m_TransientArray.GetAt(m_nCurrentTransient)->GetTransientClass());
        pStatus->SetPaneText(1,str);
    }
}
void CTAC_MMDoc::UpdateStatusBarNoTransients()
{
    CString str;
    CMainFrame *pFrame=(CMainFrame*) AfxGetApp()->m_pMainWnd;
    CStatusBar *pStatus=&pFrame->m_wndStatusBar;
    if (pStatus) {
        str.Format(" No transients in memory.");
        pStatus->SetPaneText(1,str);
    }
}
void CTAC_MMDoc::UpdateStatusBarUnknown()
{
    CString str;
    CMainFrame *pFrame=(CMainFrame*) AfxGetApp()->m_pMainWnd;
    CStatusBar *pStatus=&pFrame->m_wndStatusBar;
    if (pStatus) {
        str.Format(" Unknown Transient.");
        pStatus->SetPaneText(1,str);
    }
}

```

```

    }
}
void CTAC_MMDoc::OnDatabaseDeletetransient()
{
    if (AfxMessageBox("Are you sure you want to delete the current transient?",
        MB_YESNO)==IDYES) {
        delete m_TransientArray.GetAt(m_nCurrentTransient);
        m_TransientArray.RemoveAt(m_nCurrentTransient);
        SetModifiedFlag(TRUE);
        if (m_nCurrentTransient>m_TransientArray.GetUpperBound()) {
            DecrementCurrentTransient();
        }
        else {
            UpdateStatusBar();//status bar is updated in decrement otherwise
        }
        if(m_nCurrentTransient>=1) {
            ViewNeedsUpdating();
        }
        else {
            m_bFileInMemory=FALSE;
            UpdateAllViews(NULL);
        }
        m_PNN.SetNumCases(m_TransientArray.GetSize());
        m_PNN.SetNumClasses(FindNumClasses());
    }
}
void CTAC_MMDoc::ViewNeedsUpdating()
{
    if (m_bViewRawSignalFlag) {
        CString PathName=m_TransientArray.GetAt(m_nCurrentTransient)->
            GetRawFilePath();
        if (!ReadRawSignal(PathName,FALSE)) {
            for(int ctr=0; ctr<m_Extract.GetRawFileSize(); ctr++) {
                m_nRawSignal[ctr]=0;
            }
        }
    }
    if (m_bViewVarDimTrajFlag) {
        m_Extract.FillVarDimArray(m_nRawSignal, m_dVarDimTraj,
            m_bViewSegmentation, m_TransientArray.GetAt(m_nCurrentTransient)->
            GetTransientBeginsHere(),m_TransientArray.
            GetAt(m_nCurrentTransient)->GetTransientModel());
    }
    m_nTransientBeginsHere=m_TransientArray.GetAt(m_nCurrentTransient)->
        GetTransientBeginsHere();
    m_bScrollToTransient=TRUE;
    UpdateAllViews(NULL);
}
void CTAC_MMDoc::OnViewPrev()
{
    if (m_nCurrentTransient>0) {
        DecrementCurrentTransient();
        ViewNeedsUpdating();
    }
}

```

```

}
void CTAC_MMDoc::OnViewNext()
{
    if (m_nCurrentTransient < m_TransientArray.GetUpperBound()) {
        IncrementCurrentTransient();
        ViewNeedsUpdating();
    }
}
void CTAC_MMDoc::OnViewRawsignal()
{
    if(m_bViewRawSignalFlag)
        m_bViewRawSignalFlag=FALSE;
    else
        m_bViewRawSignalFlag=TRUE;
    UpdateAllViews(NULL);
}
void CTAC_MMDoc::OnViewFractaltrajectory()
{
    if(m_bViewVarDimTrajFlag)
        m_bViewVarDimTrajFlag=FALSE;
    else
        m_bViewVarDimTrajFlag=TRUE;
    UpdateAllViews(NULL);
}
void CTAC_MMDoc::OnFilePageSetup()
{
    if(m_PSDdlg.DoModal()==IDOK) {
        AfxGetApp()->SelectPrinter(m_PSDdlg.m_psd.hDevNames,
            m_PSDdlg.GetDevMode(), FALSE);
    }
}
void CTAC_MMDoc::OnNeuralnetClassify()
{
    CString PathName;
    int TransientModelSize=m_Extract.GetTransientModelSize();
    double *SummationNeuron, Activation;
    CFileDialog FileDlg(TRUE, "raw", "*."); //TRUE means file open,
    FileDlg.m_ofn.lpstrTitle="Classify Transient: Select a File";

    if (FileDlg.DoModal()==IDOK) {
        HCURSOR Wait=AfxGetApp()->LoadStandardCursor(IDC_WAIT);
        ::SetCursor(Wait);
        PathName=FileDlg.GetPathName();
        if (ReadRawSignal(PathName,TRUE)) { //sees if file exists, is compatible
            CClassifyDialog ClassifyDlg;
            if(m_nNumClassifyModes==1) {
                m_bInBatchProcess=TRUE; //disables menu functions
                m_nTransientBeginsHere=m_Extract.FindTransient(m_nRawSignal,
                    m_dVarDimTraj, m_dTransientModel);
                m_bInBatchProcess=FALSE;
                if (m_nTransientBeginsHere==1) {
                    AfxMessageBox("No transient found. Try adjusting settings."
                        ,MB_OK);
                    return;
                }
            }
        }
    }
}

```

```

}
SummationNeuron=new double[m_PNN.GetNumClasses()];
ClassifyDlg.m_PredictedClass=m_PNN.Classify(m_dTransientModel,
    SummationNeuron,-1, m_dRejectThreshold, &Activation);
ClassifyDlg.m_Confidence=100.0*ComputeConfidence
    (SummationNeuron, ClassifyDlg.m_PredictedClass);
ClassifyDlg.m_Activation=Activation;
delete [] SummationNeuron;
}
else {
    double Confidence;
    ClassifyDlg.m_PredictedClass=MultiModeClassify
        (&Confidence, &Activation);
    ClassifyDlg.m_Confidence=Confidence;
    ClassifyDlg.m_Activation=Activation;
}
m_bScrollToTransient=TRUE;
m_bViewSegmentation=TRUE;           //show tran Segmentation info
m_bNotClassifying=FALSE;
UpdateAllViews(NULL);
UpdateStatusBarUnknown();
if(ClassifyDlg.DoModal()==IDOK) { //Add to database selected
    CAddUnknownDialog AddUnknownDlg;
    AddUnknownDlg.m_TransmitterMake=m_TransmitterMake;
    AddUnknownDlg.m_TransmitterModel=m_TransmitterModel;
    AddUnknownDlg.m_TransmitterSerial=m_TransmitterSerial;
    AddUnknownDlg.m_Date=m_Date;
    AddUnknownDlg.m_Time=m_Time;
    AddUnknownDlg.m_TransientClass=ClassifyDlg.m_PredictedClass;
    AddUnknownDlg.m_Position=m_TransientArray.GetSize()+1;
    if(AddUnknownDlg.DoModal()==IDOK) {
        CTransientData *UnknownTransient=
            new CTransientData(TransientModelSize);
        UnknownTransient->SetRawFilePath(PathName);
        UnknownTransient->SetTransientBeginsHere
            (m_nTransientBeginsHere);
        UnknownTransient->SetTransmitterMake
            (AddUnknownDlg.m_TransmitterMake);
        UnknownTransient->SetTransmitterModel
            (AddUnknownDlg.m_TransmitterModel);
        UnknownTransient->SetTransmitterSerial
            (AddUnknownDlg.m_TransmitterSerial);
        UnknownTransient->SetTransmitDate(AddUnknownDlg.m_Date);
        UnknownTransient->SetTransmitTime(AddUnknownDlg.m_Time);
        UnknownTransient->SetTransientModelSize
            (TransientModelSize);
        UnknownTransient->SetTransientModel(m_dTransientModel);
        UnknownTransient->SetTransientClass
            (AddUnknownDlg.m_TransientClass);
        m_TransientArray.InsertAt
            ((AddUnknownDlg.m_Position-1), UnknownTransient);

        m_nCurrentTransient=AddUnknownDlg.m_Position-1;
        UpdateStatusBar();
    }
}

```

```

        m_bScrollToTransient=TRUE;
        m_bViewSegmentation=TRUE;           //show tran Segmentation info
        SetModifiedFlag(TRUE);
        m_PNN.SetNumCases(m_TransientArray.GetSize());
        m_PNN.SetNumClasses(FindNumClasses());
    }
    }
    m_bNotClassifying=TRUE;
}
UpdateStatusBar();
ViewNeedsUpdating();
}
HCURSOR Arrow=AfxGetApp()->LoadStandardCursor(IDC_ARROW);
::SetCursor(Arrow);
}
double CTAC_MMDoc::ComputeConfidence(double *SummationNeuron,
                                     int PredictedClass)
{
    double Confidence, Sum=0.0;
    for(int ctr=0; ctr<m_PNN.GetNumClasses(); ctr++) {
        Sum+=SummationNeuron[ctr];
    }
    if(Sum==0.0||PredictedClass== -1)
        return 0.0;

    Confidence=SummationNeuron[PredictedClass]/Sum;
    return Confidence;
}
int CTAC_MMDoc::FindNumClasses()
{
    int Biggest=0;
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetTransientClass()>Biggest) {
            Biggest=m_TransientArray.GetAt(ctr)
                ->GetTransientClass();
        }
    }
    return (Biggest+1);           //account for zero index
}
void CTAC_MMDoc::OnNeuralnetInitializesigmast()
{
    if (m_PNN.GetSigmasOptimized()==TRUE) {
        if (AfxMessageBox("Destroy Previous Sigma Optimization?"
            ,MB_YESNO)==IDNO) {
            return;
        }
    }
    if (m_bViewTrainingResults)
        OnViewZoomedrawsignal();

    CEnterSigmaInitParamsDialog EnterSigmaInitParamsDlg;
    if(m_PNN.GetSigmasInitialized()==TRUE) {
        EnterSigmaInitParamsDlg.m_LowerSearchLimit=0.9*m_PNN.GetFirstSigma();
    }
}

```

```

        EnterSigmaInitParamsDlg.m_UpperSearchLimit=1.1*m_PNN.GetFirstSigma();
    }
    else {
        EnterSigmaInitParamsDlg.m_LowerSearchLimit=0.01;
        EnterSigmaInitParamsDlg.m_UpperSearchLimit=3;
    }
    EnterSigmaInitParamsDlg.m_NumSearchPoints=30;

    if (EnterSigmaInitParamsDlg.DoModal()==IDOK) {
        CSigmaInitDialog SigmaInitDlg;
        SigmaInitDlg.m_TransientArray=&m_TransientArray;
        SigmaInitDlg.m_PNN=&m_PNN;
        m_PNN.SetLowerSearchLimit(EnterSigmaInitParamsDlg.m_LowerSearchLimit);
        m_PNN.SetUpperSearchLimit(EnterSigmaInitParamsDlg.m_UpperSearchLimit);
        SigmaInitDlg.m_nNumSearchPoints=
            EnterSigmaInitParamsDlg.m_NumSearchPoints;
        if(SigmaInitDlg.DoModal()==IDOK) {
            SetModifiedFlag(TRUE);
            DeleteResultsArray();
            m_PNN.FillTrainingResultsArray(&m_ResultsArray);
            m_bResultsExist=TRUE;
            m_bTrainingResults=TRUE;        // false if results are from batch classify
        }
    }
}

void CTAC_MMDoc::OnNeuralnetOptimizesigmas()
{
    if (m_bViewTrainingResults)
        OnViewZoomedrawsignal();
    CTrainSigmasOptDialog SigmasOptDlg;
    SigmasOptDlg.m_TransientArray=&m_TransientArray;
    SigmasOptDlg.m_PNN=&m_PNN;
    SigmasOptDlg.m_dImprovementTolerance=1.0e-250;
    if(SigmasOptDlg.DoModal()==IDOK) {
        SetModifiedFlag(TRUE);
        DeleteResultsArray();
        m_PNN.FillTrainingResultsArray(&m_ResultsArray);
        m_bResultsExist=TRUE;
        m_bTrainingResults=TRUE;        // false if results are from batch classify
    }
}

void CTAC_MMDoc::OnEditFractalparameters()
{
    int OldModelWindowShift;
    int NewModelSize=0;
    CFractalParamDialog FractalParamDlg;
    FractalParamDlg.m_RawFileSize=m_Extract.GetRawFileSize();
    FractalParamDlg.m_TransientSize=m_Extract.GetTransientSize();
    FractalParamDlg.m_WindowSize=m_Extract.GetWindowSize(); //Segmentation window
    FractalParamDlg.m_VariancePairs=m_Extract.GetVariancePairs(); //Segmentation
    FractalParamDlg.m_Threshold=m_Extract.GetThreshold();
    FractalParamDlg.m_ModelWindowSize=m_Extract.GetModelWindowSize();
    FractalParamDlg.m_ModelVariancePairs=m_Extract.GetModelVariancePairs();
    OldModelWindowShift=FractalParamDlg.m_ModelWindowShift

```

```

    =m_Extract.GetModelWindowShift();
    if(FractalParamDlg.DoModal()==IDOK) {
        CEditFPProgressDialog EditFPProgressDlg;
        EditFPProgressDlg.m_ProgressBarLimit=m_TransientArray.GetSize();
        if (FractalParamDlg.m_SetSegmentationParams) {
            m_Extract.SetWindowSize(FractalParamDlg.m_WindowSize);
            m_Extract.SetVariancePairs(FractalParamDlg.m_VariancePairs);
            m_Extract.SetThreshold(FractalParamDlg.m_Threshold);
            if (FractalParamDlg.m_SetModelParams) {
                m_Extract.SetModelWindowSize(FractalParamDlg.m_ModelWindowSize);
                m_Extract.SetModelVariancePairs
                    (FractalParamDlg.m_ModelVariancePairs);
                m_Extract.SetModelWindowShift(FractalParamDlg.m_ModelWindowShift);
            }
            m_bInBatchProcess=TRUE;           //disables menu functions

            if(OldModelWindowShift!=FractalParamDlg.m_ModelWindowShift) {
                NewModelSize=m_Extract.GetTransientModelSize();
            }
            delete [] m_dTransientModel;
            m_dTransientModel=new double
                [m_Extract.GetTransientModelSize()];
            m_PNN.Initialize(m_Extract.GetTransientModelSize());

            EditFPProgressDlg.Create();
            SegmentAllTransients(NewModelSize, &EditFPProgressDlg);
            EditFPProgressDlg.DestroyWindow();
            m_bInBatchProcess=FALSE;
        }
        else if (FractalParamDlg.m_SetModelParams) {
            m_Extract.SetModelWindowSize(FractalParamDlg.m_ModelWindowSize);
            m_Extract.SetModelVariancePairs(FractalParamDlg.m_ModelVariancePairs);
            m_Extract.SetModelWindowShift(FractalParamDlg.m_ModelWindowShift);
            m_bInBatchProcess=TRUE;           //disables menu functions

            if(OldModelWindowShift!=FractalParamDlg.m_ModelWindowShift) {
                NewModelSize=m_Extract.GetTransientModelSize();
            }

            delete [] m_dTransientModel;
            m_dTransientModel=new double
                [m_Extract.GetTransientModelSize()];
            m_PNN.Initialize(m_Extract.GetTransientModelSize());

            EditFPProgressDlg.Create();
            ComputeAllModels(NewModelSize, &EditFPProgressDlg);
            EditFPProgressDlg.DestroyWindow();
            m_bInBatchProcess=FALSE;
        }
    }
}

void CTAC_MMDoc::SegmentAllTransients(int NewModelSize,
                                     CEditFPProgressDialog *ProgressDlg)
{

```

```

CString PathName;
BOOL FileError=FALSE;
int SegmentationErrors=0;
int TransientBeginsHere;
HCURSOR Wait=AfxGetApp()->LoadStandardCursor(IDC_WAIT);
::SetCursor(Wait);

SetModifiedFlag(TRUE);
for (int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
    CheckWindowsMessages(5);
    ProgressDlg->OnUpdateProgress(ctr+1);
    ProgressDlg->m_UpdateProgress.SetPos(ctr+1);
    CheckWindowsMessages(5);
    PathName=m_TransientArray.GetAt(ctr)->GetRawFilePath();
    if (ReadRawSignal(PathName,FALSE)) { //checks to see if file exists
        TransientBeginsHere=m_Extract.FindTransient(m_nRawSignal,
            m_dVarDimTraj, m_dTransientModel);
        if (TransientBeginsHere==1) {
            SegmentationErrors+=1;
        }
        else {
            m_TransientArray.GetAt(ctr)->
                SetTransientBeginsHere(TransientBeginsHere);
            if(NewModelSize>0) {
                m_TransientArray.GetAt(ctr)->
                    UpdateTransientModelSize(NewModelSize);
            }
            m_TransientArray.GetAt(ctr)->
                SetTransientModel(m_dTransientModel);
        }
    }
    else {
        AfxMessageBox("Raw file not found. Aborting",MB_OK);
        FileError=TRUE;
        break;
    }
}

if (SegmentationErrors&&!FileError) {
    CString strText;
    strText.Format
        ("There were %d segmentation errors.\nOriginal positions retained."
        ,SegmentationErrors);
    AfxMessageBox(strText,MB_OK);
}

if (FileError) {
    SetModifiedFlag(FALSE);
    UpdateStatusBarNoTransients();
    DeleteContents(); //take no chance of user
                    //killing original database
}

else{
    m_bViewSegmentation=TRUE; //show tran Segmentation info
    ViewNeedsUpdating();
}

```



```

        m_PNN.SetSigmasInitialized(FALSE);
        m_PNN.SetSigmasOptimized(FALSE);
    }
    HCURSOR Arrow=AfxGetApp()->LoadStandardCursor(IDC_ARROW);
    ::SetCursor(Arrow);
}
void CTAC_MMDoc::ComputeAllModels(int NewModelSize,
                                  CEditFPProgressDialog *ProgressDlg)
{
    CString PathName;
    BOOL FileError=FALSE;
    int TransientBeginsHere;
    HCURSOR Wait=AfxGetApp()->LoadStandardCursor(IDC_WAIT);
    ::SetCursor(Wait);

    SetModifiedFlag(TRUE);
    for (int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        CheckWindowsMessages(5);
        ProgressDlg->OnUpdateProgress(ctr+1);
        ProgressDlg->m_UpdateProgress.SetPos(ctr+1);
        CheckWindowsMessages(5);
        PathName=m_TransientArray.GetAt(ctr)->GetRawFilePath();
        if (ReadRawSignal(PathName,FALSE)) { //checks to see if file exists
            TransientBeginsHere=
                m_TransientArray.GetAt(ctr)->GetTransientBeginsHere();
            m_Extract.FillTransientModel
                (m_nRawSignal, TransientBeginsHere, m_dTransientModel);
            if(NewModelSize>0) {
                m_TransientArray.GetAt(ctr)->
                    UpdateTransientModelSize(NewModelSize);
            }
            m_TransientArray.GetAt(ctr)->
                SetTransientModel(m_dTransientModel);
        }
        else {
            AfxMessageBox("Raw file not found. Aborting",MB_OK);
            FileError=TRUE;
            break;
        }
    }
    if (FileError) {
        SetModifiedFlag(FALSE);
        UpdateStatusBarNoTransients();
        DeleteContents(); //take no chance of user
                        //killing original database
    }
    else {
        m_bViewSegmentation=TRUE; //show tran Segmentation info
        ViewNeedsUpdating();
        m_PNN.SetSigmasInitialized(FALSE);
        m_PNN.SetSigmasOptimized(FALSE);
    }
    HCURSOR Arrow=AfxGetApp()->LoadStandardCursor(IDC_ARROW);
    ::SetCursor(Arrow);
}

```

```

}
void CTAC_MMDoc::OnViewSearch()
{
    CSearchDialog SearchDlg;
    if(SearchDlg.DoModal()!=IDOK) {
        return;
    }

    int Temp=-1;
    switch (SearchDlg.m_FieldToSearch)
    {
        case(0): {
            Temp=SearchForClass(SearchDlg.m_Class);
            break;
        }
        case(1): {
            Temp=SearchForMake(SearchDlg.m_Make);
            break;
        }
        case(2): {
            Temp=SearchForModel(SearchDlg.m_Model);
            break;
        }
        case(3): {
            Temp=SearchForSerial(SearchDlg.m_Serial);
            break;
        }
        case(4): {
            Temp=SearchForDate(SearchDlg.m_Date);
            break;
        }
        case(5): {
            Temp=SearchForTime(SearchDlg.m_Time);
            break;
        }
    }
    if(Temp!=-1) {
        m_nCurrentTransient=Temp;
        UpdateStatusBar();
        ViewNeedsUpdating();
    }
    else {
        AfxMessageBox("Search field not found.",MB_OK);
    }
}

int CTAC_MMDoc::SearchForClass(int Class)
{
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetTransientClass()==Class) {
            return ctr;
        }
    }
    return -1;          //search unsuccessful
}

```

```

}
int CTAC_MMDoc::SearchForMake(CString Make)
{
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetTransmitterMake()==Make) {
            return ctr;
        }
    }
    return -1;        //search unsuccessful
}
int CTAC_MMDoc::SearchForModel(CString Model)
{
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetTransmitterModel()==Model) {
            return ctr;
        }
    }
    return -1;        //search unsuccessful
}
int CTAC_MMDoc::SearchForSerial(CString Serial)
{
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetTransmitterSerial()==Serial) {
            return ctr;
        }
    }
    return -1;        //search unsuccessful
}
int CTAC_MMDoc::SearchForDate(CString Date)
{
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetDate()==Date) {
            return ctr;
        }
    }
    return -1;        //search unsuccessful
}
int CTAC_MMDoc::SearchForTime(CString TTime)
{
    for(int ctr=0; ctr<m_TransientArray.GetSize(); ctr++) {
        if (m_TransientArray.GetAt(ctr)
            ->GetTime()==TTime) {
            return ctr;
        }
    }
    return -1;        //search unsuccessful
}
void CTAC_MMDoc::On ViewZoomedrawsignal()
{
    m_bViewTrainingResults=FALSE;
}

```

```

    m_bScrollToTransient=TRUE;
    UpdateAllViews(NULL);
}
void CTAC_MMDoc::OnViewTrainingresults()
{
    m_bViewTrainingResults=TRUE;
    UpdateAllViews(NULL);
}
void CTAC_MMDoc::DeleteResultsArray()
{
    int ctr=m_ResultsArray.GetSize();
    while (ctr--) {
        delete m_ResultsArray.GetAt(ctr);
    }
    m_ResultsArray.RemoveAll();
}
void CTAC_MMDoc::OnNeuralnetBatchclassify()
{
    if (m_bViewTrainingResults)
        OnViewZoomedrawsignal();
    CString PathName;
    int TransientModelSize=m_Extract.GetTransientModelSize();
    int NumFiles, NumClasses=m_PNN.GetNumClasses();
    double *SummationNeuron, Confidence, Activation;
    CString FileName;
    char temp[50];
    int CorrectClass, PredictedClass;
    m_bNotClassifying=FALSE; //so view isn't messed up by batch transients
    CFileDialog FileDlg(TRUE, "blf", "*.bf"); //TRUE means file open,
    FileDlg.m_ofn.lpstrTitle="Batch Classify: Select a Batch List File";

    if (FileDlg.DoModal()!=IDOK) {
        m_bResultsExist=FALSE;
        return;
    }
    CCmdTarget::BeginWaitCursor();
    PathName=FileDlg.GetPathName();
    ifstream tfile(PathName, ios::nocreate);
    if (tfile.fail()) {
        AfxMessageBox("Error opening file.",MB_OK);
        return;
    }

    tfile >> NumFiles;
    if (NumFiles<0||NumFiles>10000||tfile.eof()) {
        AfxMessageBox("Incompatibe file format.",MB_OK);
        return;
    }
    DeleteResultsArray();
    m_bTrainingResults=FALSE; // TRUE if results are from training
    m_bResultsExist=TRUE;
    for(int ctr=0; ctr<NumFiles; ctr++) {
        CTrainingResults *TrainingResults=new CTrainingResults(1);
        //only 1 Segmentation mode

```

```

tfile >> temp;
FileName=temp;
TrainingResults->SetRawFilePath(FileName);
tfile >> CorrectClass;
if (CorrectClass<0||CorrectClass>NumClasses!tfile.eof()){
    AfxMessageBox("Incompatibe file format.",MB_OK);
    m_bResultsExist=FALSE;
    return;
}
TrainingResults->SetCorrectClass(CorrectClass);
m_ResultsArray.InsertAt(ctr, TrainingResults);
}
tfile.close;
CBatchProgressDialog BatchProgressDlg;
BatchProgressDlg.m_ProgressBarLimit=NumFiles;
m_bInBatchProcess=TRUE;          // disables all menu functions
BatchProgressDlg.Create();
for(ctr=0; ctr<NumFiles; ctr++) {
    CheckWindowsMessages(20);
    BatchProgressDlg.OnUpdateProgress(ctr+1);
    BatchProgressDlg.m_UpdateProgress.SetPos(ctr+1);
    CheckWindowsMessages(20);
    if (ReadRawSignal(m_ResultsArray.GetAt(ctr)->GetRawFilePath(),FALSE)) {
        if(m_nNumClassifyModes==1) {
            m_nTransientBeginsHere=m_Extract.FindTransient(m_nRawSignal,
                m_dVarDimTraj, m_dTransientModel);
            if (m_nTransientBeginsHere!=1) {
                SummationNeuron=new double[NumClasses];
                PredictedClass=m_PNN.Classify(m_dTransientModel,
                    SummationNeuron,-1,m_dRejectThreshold,&Activation);
                m_ResultsArray.GetAt(ctr)
                    ->SetPredictedClass(PredictedClass,0);
                m_ResultsArray.GetAt(ctr)
                    ->SetActivation(Activation,0);
                Confidence=100.0*ComputeConfidence
                    (SummationNeuron, PredictedClass);
                m_ResultsArray.GetAt(ctr)->SetError(Confidence,0);
                delete [] SummationNeuron;
            }
            else {          // no transient found in raw file
                m_ResultsArray.GetAt(ctr)->SetPredictedClass(-999,0);
                m_ResultsArray.GetAt(ctr)->SetError(0,0,0);
                m_ResultsArray.GetAt(ctr)->SetActivation(0,0,0);
            }
        }
        else {
            double Confidence;
            PredictedClass=MultiModeClassify(&Confidence,&Activation);
            m_ResultsArray.GetAt(ctr)
                ->SetPredictedClass(PredictedClass,0);
            m_ResultsArray.GetAt(ctr)->SetError(Confidence,0);
            m_ResultsArray.GetAt(ctr)->SetActivation(Activation,0);
        }
    }
}
}

```

```

    }
    BatchProgressDlg.DestroyWindow();
    m_bInBatchProcess=FALSE;
    HCURSOR Arrow=AfxGetApp()->LoadStandardCursor(IDC_ARROW);
    ::SetCursor(Arrow);
    m_bNotClassifying=TRUE;
    ViewNeedsUpdating();
}
void CTAC_MMDoc::CheckWindowsMessages(int NumChecks)
{
    MSG message;
    for(int ctr=0; ctr<NumChecks; ctr++) {
        if (::PeekMessage(&message, NULL, 0, 0, PM_REMOVE)) {
            ::TranslateMessage(&message);
            ::DispatchMessage(&message);
        }
    }
}
int CTAC_MMDoc::MultiModeClassify(double *WinningConfidence,
                                   double *Activation)
{
    int PredictedClass, PossibleClass;
    int *TransientBeginsHere;
    double **TransientModel, **SummationNeuron, HighestActivation;
    TransientBeginsHere=new int[m_nNumClassifyModes];
    TransientModel=new double*[m_nNumClassifyModes];
    SummationNeuron=new double*[m_nNumClassifyModes];
    for(int ctr=0; ctr<m_nNumClassifyModes; ctr++) {
        TransientModel[ctr]=new double[m_Extract.GetTransientSize()];
        SummationNeuron[ctr]=new double[m_PNN.GetNumClasses()];
    }
    m_Extract.FindMultiTransients(m_nRawSignal, m_dVarDimTraj,
                                   TransientModel, m_nNumClassifyModes, m_dThresholds,
                                   m_nMinSeparation, TransientBeginsHere);

    HighestActivation=0.0;
    for(ctr=0; ctr<m_nNumClassifyModes; ctr++) {
        if(TransientBeginsHere[ctr]!=-1) {
            PossibleClass=m_PNN.Classify(TransientModel[ctr],
                                           SummationNeuron[ctr],-1, m_dRejectThreshold, Activation);
            if(*Activation>HighestActivation) {
                *WinningConfidence=100.0*ComputeConfidence
                    (SummationNeuron[ctr], PossibleClass);
                HighestActivation=*Activation;
                PredictedClass=PossibleClass;
                m_nTransientBeginsHere=TransientBeginsHere[ctr];
                m_dTransientModel=TransientModel[ctr];
            }
        }
    }
    delete [] TransientBeginsHere;
    for(ctr=0; ctr<m_nNumClassifyModes; ctr++) {
        delete [] TransientModel[ctr];
        delete [] SummationNeuron[ctr];
    }
}

```

```

    }
    delete [] TransientModel;
    delete [] SummationNeuron;
    *Activation=HighestActivation;
    if(*WinningConfidence>0.0)
        return PredictedClass;
    else
        return -999;// no valid transient found at all
}
void CTAC_MMDoc::OnNeuralnetModeparameters()
{
    CModeParamsDialog ModeParamsDlg;
    if(m_nNumClassifyModes==1)
        ModeParamsDlg.m_Mode=0;
    else {
        ModeParamsDlg.m_Mode=1;
        ModeParamsDlg.m_NumModes=m_nNumClassifyModes;
        ModeParamsDlg.m_LowThreshold=m_dThresholds[0];
        ModeParamsDlg.m_UpperThreshold=m_dThresholds[m_nNumClassifyModes-1];
        ModeParamsDlg.m_MinSeparation=m_nMinSeparation;
    }
    if(ModeParamsDlg.DoModal()==IDOK) {
        if(ModeParamsDlg.m_Mode==0||ModeParamsDlg.m_NumModes==1) {
            m_nNumClassifyModes=1;
            return;
        }
        else {
            m_nNumClassifyModes=ModeParamsDlg.m_NumModes;
            m_nMinSeparation=ModeParamsDlg.m_MinSeparation;
            delete [] m_dThresholds;
            m_dThresholds=new double[m_nNumClassifyModes];
            double Increment=(ModeParamsDlg.m_UpperThreshold
                -ModeParamsDlg.m_LowThreshold)/(double)(m_nNumClassifyModes-1);
            for(int ctr=0; ctr<m_nNumClassifyModes; ctr++) {
                m_dThresholds[ctr]=ModeParamsDlg.m_LowThreshold+ctr*Increment;
            }
        }
    }
}
void CTAC_MMDoc::OnNeuralnetSetrejectionthreshold()
{
    CSetRejectionDialog RejectionDialog;
    RejectionDialog.m_dRejectionThreshold=m_dRejectThreshold;
    if (RejectionDialog.DoModal()==IDOK)
        m_dRejectThreshold=RejectionDialog.m_dRejectionThreshold;
}
void CTAC_MMDoc::OnUpdateDatabaseAddtransient(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bDocCreated&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateDatabaseDeletetransient(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&& m_nCurrentTransient!=1
        &&!m_bInBatchProcess);
}

```

```

}
void CTAC_MMDoc::OnUpdateViewPrev(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&& m_nCurrentTransient>0&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateViewNext(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&
        m_nCurrentTransient<m_nTransientArray.GetUpperBound()
        &&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateViewFractaltrajectory(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
    pCmdUI->SetCheck(m_bViewVarDimTrajFlag ? 1 : 0);
}
void CTAC_MMDoc::OnUpdateViewRawsignal(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
    pCmdUI->SetCheck(m_bViewRawSignalFlag ? 1 : 0);
}
void CTAC_MMDoc::OnUpdateFilePageSetup(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateFilePrint(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateFilePrintPreview(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateFileSave(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(IsModified())&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateNeuralnetClassify(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&m_PNN.GetSigmasInitialized()
        &&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateNeuralnetInitializesigmas(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateEditFractalparameters(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateViewSearch(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}

```



```

}
void CTAC_MMDoc::OnUpdateNeuralnetOptimizsigmas(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&& m_PNN.GetSigmasInitialized()
        &&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateViewZoomedrawsignal(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
    pCmdUI->SetCheck(m_bViewTrainingResults ? 0 : 1);
}
void CTAC_MMDoc::OnUpdateViewTrainingresults(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&& m_bResultsExist&&!m_bInBatchProcess);
    pCmdUI->SetCheck(m_bViewTrainingResults ? 1 : 0);
}
void CTAC_MMDoc::OnUpdateNeuralnetBatchclassify(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&& m_PNN.GetSigmasInitialized()
        &&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateNeuralnetModeparameters(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateFileNew(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateFileOpen(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(!m_bInBatchProcess);}
void CTAC_MMDoc::OnUpdateFileSaveAs(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(!m_bInBatchProcess);
}
void CTAC_MMDoc::OnUpdateNeuralnetSetrejectionthreshold(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bFileInMemory&&!m_bInBatchProcess);
}

```

TAC_MMView.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```

// TAC_MMView.h : interface of the CTAC_MMView class
//
////////////////////////////////////////////////////////////////////
class CTAC_MMView : public CScrollView
{
protected: // create from serialization only
    CTAC_MMView();
    DECLARE_DYNCREATE(CTAC_MMView)
// Attributes
public:
    CTAC_MMDoc* GetDocument();
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CTAC_MMView)
public:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
protected:
    virtual void OnInitialUpdate(); // called first time after construct
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CTAC_MMView();
    void PrintTrainingResults(CDC* pDC);
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:
    CRect m_rectClient;
    int m_TopMargin;
    int m_BottomMargin;
    int m_LeftMargin;
    int m_RightMargin;
    int m_FontHeight;
    int m_TextLinesPerPage;
// Generated message map functions
protected:
    //{{AFX_MSG(CTAC_MMView)
    //}}AFX_MSG

```

```
    DECLARE_MESSAGE_MAP()
};
#ifdef _DEBUG // debug version in TAC_MMView.cpp
inline CTAC_MMDoc* CTAC_MMView::GetDocument()
    { return (CTAC_MMDoc*)m_pDocument; }
#endif
////////////////////////////////////////////////////////////////
```

TAC_MMView.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// TAC_MMView.cpp : implementation of the CTAC_MMView class
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "Trandata.h"
#include "TAC_MMDoc.h"
#include "TAC_MMView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CTAC_MMView
IMPLEMENT_DYNCREATE(CTAC_MMView, CScrollView)
BEGIN_MESSAGE_MAP(CTAC_MMView, CScrollView)
    {{{AFX_MSG_MAP(CTAC_MMView)
    //}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()
////////////////////////////////////
// CTAC_MMView construction/destruction
CTAC_MMView::CTAC_MMView()
{
}
CTAC_MMView::~CTAC_MMView()
{
}
BOOL CTAC_MMView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs
    return CScrollView::PreCreateWindow(cs);
}
////////////////////////////////////
// CTAC_MMView drawing
void CTAC_MMView::OnDraw(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    OnPrepareDC(pDC);
    if(!pDoc->m_bFileInMemory) {
        return;
    }
    if (pDoc->m_bViewTrainingResults) {
```

```

    PrintTrainingResults(pDC);
    return;
}

int TransientBeginsHere=pDoc->GetTransientBeginsHere();

int SigHeight=pDoc->GetHighestValue()-pDoc->GetLowestValue();
int ClipYCoord=pDoc->GetLowestValue();

CRect rectClient;
GetClientRect(rectClient);
int *RawSignal=pDoc->GetRawSignal();
int TransientSize=pDoc->GetExtract()->GetTransientSize();
int RawFileSize=pDoc->GetExtract()->GetRawFileSize();
int LeftTextMargin=(int)(rectClient.right/20.0);

CPen *pOldPen;
CPen GreyDashPen(PS_DASH, 1, RGB(192,192,192));
CPen ThickBlackPen(PS_SOLID, 3, RGB(0,0,0));

pOldPen=pDC->SelectObject(&ThickBlackPen);
pDC->MoveTo(LeftTextMargin,0);
pDC->LineTo(LeftTextMargin,rectClient.bottom);
pDC->LineTo(LeftTextMargin+RawFileSize,rectClient.bottom);
pDC->SelectObject(pOldPen);
//Draw Axis and other good stuff on screen
//CSize GetViewportExt( ) const;

CPoint Init(LeftTextMargin,rectClient.bottom-((float)(RawSignal[0]
-ClipYCoord))*((float)((float)rectClient.bottom/(float)SigHeight)));
pDC->MoveTo(Init);
for(int ctr=1; ctr<RawFileSize; ctr++) {
    CPoint Next(LeftTextMargin+ctr,rectClient.bottom-
        ((float)(RawSignal[ctr]-ClipYCoord)*
        (float)((float)rectClient.bottom/(float)SigHeight)));
    pDC->LineTo(Next);
}

pOldPen=pDC->SelectObject(&GreyDashPen);
pDC->MoveTo(LeftTextMargin+TransientBeginsHere,0);
pDC->LineTo(LeftTextMargin+TransientBeginsHere,rectClient.bottom);
pDC->MoveTo(LeftTextMargin+TransientBeginsHere+TransientSize,0);
pDC->LineTo(LeftTextMargin+TransientBeginsHere+TransientSize,
    rectClient.bottom);
pDC->SelectObject(pOldPen);
if(pDoc->m_bScrollToTransient) {
    CPoint Scroll(TransientBeginsHere,0);
    ScrollToPosition(Scroll);
    pDoc->m_bScrollToTransient=FALSE;
}
}
void CTAC_MMView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();
}

```

```

    CSize sizeTotal;
    // TODO: calculate the total size of this view
    sizeTotal.cx = sizeTotal.cy = 100;
    SetScrollSizes(MM_TEXT, sizeTotal);
}
////////////////////////////////////
// CTAC_MMView printing
BOOL CTAC_MMView::OnPreparePrinting(CPrintInfo* pInfo)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    CPageSetupDialog* PSD=pDoc->GetPageSetupDlg();
    int PageWidth=PSD->m_psd.ptPaperSize.x-PSD->m_psd.rtMinMargin.left-
        PSD->m_psd.rtMinMargin.right;
    int PageLength=PSD->m_psd.ptPaperSize.y-PSD->m_psd.rtMinMargin.top
        -PSD->m_psd.rtMinMargin.bottom;
    int UsablePageWidth=PSD->m_psd.ptPaperSize.x-PSD->m_psd.rtMargin.left-
        PSD->m_psd.rtMargin.right;
    int UsablePageLength=PSD->m_psd.ptPaperSize.y-PSD->m_psd.rtMargin.top
        -PSD->m_psd.rtMargin.bottom;
    m_TopMargin=(PSD->m_psd.rtMargin.top-PSD->m_psd.rtMinMargin.top)/10;
    m_BottomMargin=(PSD->m_psd.rtMargin.bottom-PSD->m_psd.rtMinMargin.bottom)/10;
    m_LeftMargin=(PSD->m_psd.rtMargin.left-PSD->m_psd.rtMinMargin.left)/10;
    m_RightMargin=(PSD->m_psd.rtMargin.right-PSD->m_psd.rtMinMargin.right)/10;
    m_rectClient.left=0;
    m_rectClient.right=PageWidth/10;           //convert to MM_LOENGLISH
    m_rectClient.top=0;
    m_rectClient.bottom=PageLength/10;        //convert to MM_LOENGLISH
    m_FontHeight=16;
    m_TextLinesPerPage=UsablePageLength/10/m_FontHeight;
    int NumTextLines=(pDoc->GetResultsArray()->GetSize());
    int LinesPrinted=m_TextLinesPerPage-6, PrintPages=1;           // -6 accts for title
    while (LinesPrinted<(NumTextLines+4)) {                       // +4 accts for totals
        PrintPages++;
        LinesPrinted+=m_TextLinesPerPage-2;                       // -2 accts for row headings
    }
    pInfo->SetMaxPage(PrintPages);
    return DoPreparePrinting(pInfo);
}

void CTAC_MMView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
}

void CTAC_MMView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}
////////////////////////////////////
// CTAC_MMView diagnostics
#ifdef _DEBUG
void CTAC_MMView::AssertValid() const
{
    CScrollView::AssertValid();
}
}

```

```

void CTAC_MMView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}
CTAC_MMDoc* CTAC_MMView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CTAC_MMDoc)));
    return (CTAC_MMDoc*)m_pDocument;
}
#endif // _DEBUG
////////////////////////////////////
// CTAC_MMView message handlers
void CTAC_MMView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
    Invalidate();
}
void CTAC_MMView::PrintTrainingResults(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CString Field1, Field2, Field3, Field4, Field5, Field6;

    int Spacing=m_rectClient.right/8;
    if(pDoc->m_bTrainingResults) {
        Field1.Format(" Training Results ");
    }
    else {
        Field1.Format("Batch Classify Results");
    }
    CFont BigRomanFont;
    int Size=(m_rectClient.right/3)/(Field1.GetLength()-5);
    BigRomanFont.CreateFont(1.5*Size, Size, 0, 0, 600, FALSE, TRUE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    CFont *pOldFont=pDC->SelectObject(&BigRomanFont);
    pDC->TextOut(m_rectClient.right/3, 0, Field1);
    Field1.Format("Transient #");
    Field2.Format("FileName and Path");
    Field3.Format("Class");
    Field4.Format("Predicted");
    if(pDoc->m_bTrainingResults)
        Field5.Format("Error");
    else
        Field5.Format("Conf(%%)");
    Field6.Format(" Activation");
    CFont RomanFontUnder;
    RomanFontUnder.CreateFont(0, 0, 0, 0, 400, FALSE, TRUE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    pDC->SelectObject(&RomanFontUnder);
    pDC->TextOut(0, 3*Size, Field1);
    pDC->TextOut(Spacing, 3*Size, Field2);
}

```

```

pDC->TextOut(4*Spacing, 3*Size, Field3);
pDC->TextOut(5*Spacing, 3*Size, Field4);
pDC->TextOut(6*Spacing, 3*Size, Field5);
pDC->TextOut(7*Spacing, 3*Size, Field6);
CFont RomanFont;
RomanFont.CreateFont(0, 0, 0, 0, 400, FALSE, FALSE, 0,
                    ANSI_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                    DEFAULT_PITCH | FF_ROMAN, NULL);
pDC->SelectObject(&RomanFont);
int NumLines=(pDoc->GetResultsArray()->GetSize());
TEXTMETRIC Metrics;
pDC->GetTextMetrics(&Metrics);
int Height=1.25*(Metrics.tmHeight);
CSize DataSize(m_rectClient.right-14, 10*Size+Height*NumLines);
CSize SizePage=(0,100*Height);
CSize SizeLine=(0,Height);
SetScrollSizes(MM_TEXT,DataSize,SizePage,SizeLine);
double Error, TotalError=0.0;
int PredictedClass, CorrectClass, NumCorrect=0;
for (int ctr=0; ctr<NumLines; ctr++) {
    Field1.Format("%d",ctr+1);
    Field2=pDoc->GetResultsArray()->GetAt(ctr)->GetRawFilePath();
    CorrectClass=pDoc->GetResultsArray()->GetAt(ctr)->GetCorrectClass();
    Field3.Format("%d",CorrectClass);
    PredictedClass=pDoc->GetResultsArray()
        ->GetAt(ctr)->GetPredictedClass(0);
    Field4.Format("%d",PredictedClass);
    if (PredictedClass==CorrectClass)
        NumCorrect++;
    TotalError+=Error=pDoc->GetResultsArray()->GetAt(ctr)->GetError(0);
    Field5.Format("%.4lf",Error);
    if(pDoc->m_bTrainingResults)
        Field6.Format("N/A");
    else
        Field6.Format("%.3e",pDoc->GetResultsArray()
            ->GetAt(ctr)->GetActivation(0));
    pDC->TextOut(0, 5*Size+ctr*Height, Field1);
    pDC->TextOut(Spacing, 5*Size+ctr*Height, Field2);
    pDC->TextOut(4*Spacing, 5*Size+ctr*Height, Field3);
    pDC->TextOut(5*Spacing, 5*Size+ctr*Height, Field4);
    pDC->TextOut(6*Spacing, 5*Size+ctr*Height, Field5);
    pDC->TextOut(7*Spacing, 5*Size+ctr*Height, Field6);
}
Field1.Format("Number of Correct Classifications: %d/%d.",
    NumCorrect, NumLines);
pDC->TextOut(0, 6*Size+NumLines*Height, Field1);
if(pDoc->m_bTrainingResults) {
    Field1.Format("Average Error: %.12lf.", TotalError/NumLines);
    pDC->TextOut(0, 6*Size+(NumLines+1)*Height, Field1);
}
pDC->SelectObject(pOldFont);
return;
}

```



```

void CTAC_MMView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
    //this function is called immediately before OnDraw for screen
    //and immediately before OnPrint for the printer
    CScrollView::OnPrepareDC(pDC, pInfo);
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if(pDoc->m_bFileInMemory) {
        if(!pDoc->m_bViewTrainingResults) {
            GetClientRect(m_rectClient);
            int RawFileSize=pDoc->GetExtract()->GetRawFileSize();
            int LeftTextMargin=(int)(m_rectClient.right/20.0);
            CSize DataSize(RawFileSize+LeftTextMargin,m_rectClient.bottom-14);
            //compensate for bottom scrollbar(14) which is about to appear
            int TransientSize=pDoc->GetExtract()->GetTransientSize();
            CSize SizePage=(TransientSize,0);//page scroll moves by transient size
            SetScrollSizes(MM_TEXT,DataSize,SizePage);
        }
        else {
            if (!pDC->IsPrinting()) {
                pDC->SetMapMode(MM_TEXT);
                GetClientRect(m_rectClient);
            }
            else
                pDC->SetMapMode(MM_LOENGLISH);
        }
    }
}

static double TotalError;
static int NumCorrect;
void CTAC_MMView::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc->m_bViewTrainingResults) {
        CScrollView::OnPrint(pDC, pInfo);
        return;
    }

    static
    CString Field1, Field2, Field3, Field4, Field5, Field6;
    int Spacing=(m_rectClient.right-m_LeftMargin-m_RightMargin)/8;
    int HeaderOffset;
    int Start, AvailableLines, Page=pInfo->m_nCurPage;
    if (Page==1) {
        TotalError=0.0;
        NumCorrect=0;
        Start=0;
        if(pDoc->m_bTrainingResults) {
            Field1.Format("Training Results");
        }
        else {
            Field1.Format("Batch Classify Results");
        }
    }
}

```

```

    }
    CFont BigRomanFont;
    BigRomanFont.CreateFont(2*m_FontHeight, 0, 0, 0, 600, FALSE, TRUE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    CFont *pOldFont=pDC->SelectObject(&BigRomanFont);
    int Length=Field1.GetLength();
    TEXTMETRIC Metrics;
    pDC->GetTextMetrics(&Metrics);
    Length*=Metrics.tmAveCharWidth;
    pDC->TextOut(m_LeftMargin+(m_rectClient.right-m_LeftMargin
        -m_RightMargin-Length)/2, -m_TopMargin, Field1);

    pDC->SelectObject(pOldFont);
    AvailableLines=m_TextLinesPerPage-6;
    HeaderOffset=4;
}
else {
    Start=(m_TextLinesPerPage-6)+(Page-2)*(m_TextLinesPerPage-2);
    AvailableLines=m_TextLinesPerPage-2;
    HeaderOffset=0;
}

Field1.Format("Transient #");
Field2.Format("FileName and Path");
Field3.Format("Class");
Field4.Format("Predicted");
if(pDoc->m_bTrainingResults)
    Field5.Format("Error");
else
    Field5.Format("Conf(%%)");
Field6.Format("Activation");

CFont RomanFontUnder;
RomanFontUnder.CreateFont(m_FontHeight, 0, 0, 0, 400, FALSE, TRUE, 0,
    ANSI_CHARSET, OUT_DEFAULT_PRECIS,
    CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, NULL);
CFont *pOldFont=pDC->SelectObject(&RomanFontUnder);
pDC->TextOut(m_LeftMargin, -HeaderOffset*m_FontHeight-m_TopMargin, Field1);
pDC->TextOut(m_LeftMargin+Spacing, -HeaderOffset*m_FontHeight-m_TopMargin,
    Field2);
pDC->TextOut(m_LeftMargin+4*Spacing, -HeaderOffset*m_FontHeight-m_TopMargin,
    Field3);
pDC->TextOut(m_LeftMargin+5*Spacing, -HeaderOffset*m_FontHeight-m_TopMargin,
    Field4);
pDC->TextOut(m_LeftMargin+6*Spacing, -HeaderOffset*m_FontHeight-m_TopMargin,
    Field5);
pDC->TextOut(m_LeftMargin+7*Spacing, -HeaderOffset*m_FontHeight-m_TopMargin,
    Field6);
HeaderOffset+=2;
CFont RomanFont;
RomanFont.CreateFont(m_FontHeight, 0, 0, 0, 400, FALSE, FALSE, 0,

```

```

        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
pDC->SelectObject(&RomanFont);
double Error;
int PredictedClass, CorrectClass;
for (int ctr=Start; (ctr<Start+AvailableLines)
    &&ctr<pDoc->GetResultsArray()->GetSize(); ctr++) {
    Field1.Format("%d",ctr+1);
    Field2=pDoc->GetResultsArray()->GetAt(ctr)->GetRawFilePath();
    CorrectClass=pDoc->GetResultsArray()->GetAt(ctr)->GetCorrectClass();
    Field3.Format("%d",CorrectClass);
    PredictedClass=pDoc->GetResultsArray()
        ->GetAt(ctr)->GetPredictedClass();
    Field4.Format("%d",PredictedClass);
    if (PredictedClass==CorrectClass)
        NumCorrect++;
    TotalError+=Error=pDoc->GetResultsArray()->GetAt(ctr)->GetError();
    if(Error>99.9999)
        Field5.Format("%.3lf",Error);
    else if (Error==0.0)
        Field5.Format("%.5lf",Error);
    else
        Field5.Format("%.4lf",Error);
    if(pDoc->m_bTrainingResults)
        Field6.Format("N/A");
    else
        Field6.Format("%.3e",pDoc->GetResultsArray()
            ->GetAt(ctr)->GetActivation());
    pDC->TextOut(m_LeftMargin,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field1);
    pDC->TextOut(m_LeftMargin+Spacing,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field2);
    pDC->TextOut(m_LeftMargin+4*Spacing,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field3);
    pDC->TextOut(m_LeftMargin+5*Spacing,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field4);
    pDC->TextOut(m_LeftMargin+6*Spacing,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field5);
    pDC->TextOut(m_LeftMargin+7*Spacing,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field6);
}

if(Page==pInfo->GetMaxPage()) {
    int NumLines=(pDoc->GetResultsArray()->GetSize());
    HeaderOffset+=2;
    Field1.Format("Number of Correct Classifications: %d/%d.",
        NumCorrect, NumLines);
    pDC->TextOut(m_LeftMargin,
        (-HeaderOffset-(ctr-Start))*m_FontHeight-m_TopMargin, Field1);
    if(pDoc->m_bTrainingResults) {
        Field1.Format("Average Error: %.12lf.", TotalError/NumLines);
        pDC->TextOut(m_LeftMargin, (-HeaderOffset-(ctr-Start-1))
            *m_FontHeight-m_TopMargin, Field1);
    }
}

```

Appendix B: TAC-MM Source Code

```
        }  
    }  
    pDC->SelectObject(pOldFont);  
    return;  
}  
}
```

RawView.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// RawView.h : interface of the CRawView class
//
///////////////////////////////////////////////////////////////////
class CRawView : public CScrollView
{
private:
    int m_ViewFileSize;
    int m_VarDimHeight;
    int m_RawSignalHeight;
    int m_LeftTextMargin;
    int m_BottomTextMargin;
    int m_TopTextMargin;
    int m_WindowXDim;
    int m_WindowYDim;
    CRect m_rectClient;
    CPoint m_Origin;
    int m_ViewStart;//0 for entire raw signal, TransientBeginsHere for tran
    int m_ZoomYCoord;
    int m_ClipYCoord;
protected: // create from serialization only
    CRawView();
    DECLARE_DYNCREATE(CRawView)
// Attributes
public:
    CTAC_MMDoc* GetDocument();
    void PrintTransmitterInfo(CDC* pDC);
    void PlotCommonBox(CDC* pDC);
    void PlotVarDimTraj(CDC* pDC);
    void PlotRawSignal(CDC* pDC);
    void AdjustForMargins(CDC* pDC);
    void SetViewFileSize();
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CRawView)
public:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
protected:
    virtual void OnInitialUpdate(); // called first time after construct
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
    //}AFX_VIRTUAL

```

```

// Implementation
public:
    virtual ~CRawView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
    //{{AFX_MSG(CRawView)
    afx_msg void OnViewSegmentation();
    afx_msg void OnUpdateViewSegmentation(CCmdUI* pCmdUI);
    afx_msg void OnViewTransient();
    afx_msg void OnUpdateViewTransient(CCmdUI* pCmdUI);
    afx_msg void OnEditCopy();
    afx_msg void OnUpdateEditCopy(CCmdUI* pCmdUI);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#ifdef _DEBUG // debug version in RawView.cpp
inline CTAC_MMDoc* CRawView::GetDocument()
    { return (CTAC_MMDoc*)m_pDocument; }
#endif
////////////////////////////////////

```

RawView.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// RawView.cpp : implementation of the CRawView class
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "Trandata.h"
#include "TAC_MMDoc.h"
#include "RawView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CRawView
IMPLEMENT_DYNCREATE(CRawView, CScrollView)
BEGIN_MESSAGE_MAP(CRawView, CScrollView)
    //{ AFX_MSG_MAP(CRawView)
    ON_COMMAND(ID_VIEW_SEGMENTATION, OnViewSegmentation)
    ON_UPDATE_COMMAND_UI(ID_VIEW_SEGMENTATION, OnUpdateViewSegmentation)
    ON_COMMAND(ID_VIEW_TRANSIENT, OnViewTransient)
    ON_UPDATE_COMMAND_UI(ID_VIEW_TRANSIENT, OnUpdateViewTransient)
    ON_COMMAND(ID_EDIT_COPY, OnEditCopy)
    ON_UPDATE_COMMAND_UI(ID_EDIT_COPY, OnUpdateEditCopy)
    //} AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()
////////////////////////////////////
// CRawView construction/destruction
CRawView::CRawView()
{
}
CRawView::~CRawView()
{
}
BOOL CRawView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs
    return CScrollView::PreCreateWindow(cs);
}
////////////////////////////////////
// CRawView drawing
void CRawView::OnDraw(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
```

```

ASSERT_VALID(pDoc);

if(!pDoc->m_bFileInMemory) {
    return;
}
pDC->SetBkMode(TRANSPARENT);
if (pDoc->m_bNotClassifying) {
    PrintTransmitterInfo(pDC);
}
PlotCommonBox(pDC);

if (pDoc->m_bViewVarDimTrajFlag) {
    PlotVarDimTraj(pDC);
}
if (pDoc->m_bViewRawSignalFlag) {
    PlotRawSignal(pDC);
}
}
void CRawView::OnInitialUpdate()
{
    TRACE("Don, CRawView::OnInitialUpdate being executed.\n");
    CScrollView::OnInitialUpdate();

    CSize sizeTotal;
    // TODO: calculate the total size of this view
    sizeTotal.cx = sizeTotal.cy = 100;
    SetScrollSizes(MM_TEXT, sizeTotal);
}
////////////////////////////////////
// CRawView printing
BOOL CRawView::OnPreparePrinting(CPrintInfo* pInfo)
{
    return DoPreparePrinting(pInfo); //displays the Print dialog box and creates
    //a printer device context. If you want to
    //initialize the Print dialog box with
    //values other than the defaults, assign
    //values to the members of pInfo before calling.
}
void CRawView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}
void CRawView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}
////////////////////////////////////
// CRawView diagnostics
#ifdef _DEBUG
void CRawView::AssertValid() const
{
    CScrollView::AssertValid();
}

```



```

void CRawView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}
CTAC_MMDoc* CRawView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CTAC_MMDoc)));
    return (CTAC_MMDoc*)m_pDocument;
}
#endif // _DEBUG
////////////////////////////////////
// CRawView message handlers
void CRawView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if(!pDoc->m_bFileInMemory) {
        Invalidate();
        return;
    }

    m_VarDimHeight=0;
    m_RawSignalHeight=0;
    if (pDoc->m_bViewVarDimTrajFlag) {
        m_VarDimHeight=5000;
    }
    if (pDoc->m_bViewRawSignalFlag) {
        m_RawSignalHeight=8191;
    }

    Invalidate();
}
void CRawView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
    CScrollView::OnPrepareDC(pDC, pInfo);
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if(!pDoc->m_bFileInMemory) {
        return;
    }
    SetViewFileSize();
    m_LeftTextMargin=(int)(m_ViewFileSize/20.0);
    m_WindowXDim=m_ViewFileSize+m_LeftTextMargin;
    m_BottomTextMargin=(int)((m_RawSignalHeight+m_VarDimHeight)/20.0);
    m_TopTextMargin=(int)((m_RawSignalHeight+m_VarDimHeight)/20.0);
    m_WindowYDim=m_RawSignalHeight+m_BottomTextMargin+
        m_VarDimHeight+m_TopTextMargin;

    CSize DataSize(m_WindowXDim,m_WindowYDim);
    if(!pDC->IsPrinting()) {
        GetClientRect(m_rectClient);
        pDC->SetMapMode(MM_ANISOTROPIC);
    }
}

```

```

    pDC->SetWindowExt(DataSize);
    pDC->SetViewportExt(m_rectClient.right, -m_rectClient.bottom);
    m_Origin.x=(double)m_LeftTextMargin/
        (double)m_WindowXDim*(double)m_rectClient.right;
    m_Origin.y=m_rectClient.bottom-
        (double)m_BottomTextMargin/(double)m_WindowYDim*(double)m_rectClient.bottom;
    pDC->SetViewportOrg(m_Origin);
}
}
void CRawView::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if(!pDoc->m_bFileInMemory) {
        return;
    }
    m_rectClient=pInfo->m_rectDraw;
    AdjustForMargins(pDC);

    CPen *pOldPen;
    CPen ThickBlackPen(PS_SOLID, m_WindowXDim/400, RGB(0,0,0));

    pOldPen=pDC->SelectObject(&ThickBlackPen);
    pDC->Rectangle(-m_LeftTextMargin,m_WindowYDim-m_BottomTextMargin,
        m_WindowXDim-m_LeftTextMargin,-m_BottomTextMargin);
    pDC->SelectObject(pOldPen);

    CScrollView::OnPrint(pDC, pInfo);
}
void CRawView::PrintTransmitterInfo(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CString TitleLine=pDoc->GetArray()
        ->GetAt(pDoc->GetCurrentTransient()->GetTitleLine());
    int TextWidth;
    if(TitleLine.GetLength()>(3*m_WindowXDim/4)) {
        TextWidth=(3*m_WindowXDim/4)/TitleLine.GetLength();
    }
    else {
        TextWidth=0;
    }
    CFont RomanFont;
    RomanFont.CreateFont(m_TopTextMargin, TextWidth, 0, 0, 400, FALSE, FALSE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    CFont *pOldFont=pDC->SelectObject(&RomanFont);
    pDC->TextOut(0, (m_WindowYDim-m_TopTextMargin),TitleLine);
    CString Time=pDoc->GetArray()
        ->GetAt(pDoc->GetCurrentTransient()->GetTime());
    CString Date=pDoc->GetArray()
        ->GetAt(pDoc->GetCurrentTransient()->GetDate());
}

```

```

CFont HalfRomanFont;
HalfRomanFont.CreateFont(m_TopTextMargin/2, 0, 0, 0, 400, FALSE, FALSE, 0,
    ANSI_CHARSET, OUT_DEFAULT_PRECIS,
    CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, NULL);
pDC->SelectObject(&HalfRomanFont);
CSize TimeSize=pDC->GetTextExtent(Time);
int TimeX=m_WindowXDim-TimeSize.cx-m_LeftTextMargin;
CSize DateSize=pDC->GetTextExtent(Date);
int DateX=m_WindowXDim-DateSize.cx-m_LeftTextMargin;
pDC->TextOut(TimeX,(m_WindowYDim-m_TopTextMargin),Time);
pDC->TextOut(DateX,(m_WindowYDim-m_TopTextMargin*1.5),Date);
pDC->SelectObject(pOldFont);
return;
}
void CRawView::PlotCommonBox(CDC* pDC)
{
    CPen *pOldPen;
    CFont *pOldFont;
    CPen ThickBlackPen(PS_SOLID, m_WindowXDim/500, RGB(0,0,0));

    pOldPen=pDC->SelectObject(&ThickBlackPen);
    pDC->MoveTo(m_WindowXDim-m_LeftTextMargin,
        (m_WindowYDim-(m_BottomTextMargin+m_TopTextMargin)));
    pDC->LineTo(0,(m_WindowYDim-(m_BottomTextMargin+m_TopTextMargin)));
    pDC->LineTo(0,-m_BottomTextMargin/5);
    pDC->MoveTo(0,0);
    pDC->LineTo(m_WindowXDim-m_LeftTextMargin,0);
    pDC->MoveTo(m_WindowXDim-m_LeftTextMargin-m_WindowXDim/400,0);
    pDC->LineTo(m_WindowXDim-m_LeftTextMargin-m_WindowXDim/400,-m_BottomTextMargin/5);
    pDC->MoveTo((m_WindowXDim-m_LeftTextMargin)/4,0);
    pDC->LineTo((m_WindowXDim-m_LeftTextMargin)/4,-m_BottomTextMargin/5);
    pDC->MoveTo((m_WindowXDim-m_LeftTextMargin)/2,0);
    pDC->LineTo((m_WindowXDim-m_LeftTextMargin)/2,-m_BottomTextMargin/5);
    pDC->MoveTo(3*(m_WindowXDim-m_LeftTextMargin)/4,0);
    pDC->LineTo(3*(m_WindowXDim-m_LeftTextMargin)/4,-m_BottomTextMargin/5);

    CFont SmallRomanFont;
    SmallRomanFont.CreateFont(m_BottomTextMargin/2, 0, 0, 0, 400, FALSE, FALSE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    pOldFont=pDC->SelectObject(&SmallRomanFont);
    CString str;
    CSize NumberSize;
    str.Format("%d",0);
    int XShift=m_WindowXDim/500;
    pDC->TextOut(XShift,-m_BottomTextMargin/3,str);
    for(int ctr=m_ViewFileSize/4; ctr<m_ViewFileSize; ctr+=m_ViewFileSize/4) {
        str.Format("%d",ctr);
        NumberSize=pDC->GetTextExtent(str);
        XShift=ctr-NumberSize.cx/2;
        pDC->TextOut(XShift,-m_BottomTextMargin/3,str);
    }
}

```

```

    str.Format("%d",m_ViewFileSize);
    NumberSize=pDC->GetTextExtent(str);
    XShift=m_WindowXDim-m_LeftTextMargin-NumberSize.cx;
    pDC->TextOut(XShift,-m_BottomTextMargin/3,str);
    pDC->SelectObject(pOldFont);
    pDC->SelectObject(pOldPen);
}
void CRawView::PlotVarDimTraj(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CPen *pOldPen;
    CPen CyanSolidPen(PS_SOLID, 1, RGB(0,128,128));
    CPen ThickBlackPen(PS_SOLID, m_WindowXDim/500, RGB(0,0,0));
    CString YAxisTitle;
    if (pDoc->m_bViewSegmentation) {
        YAxisTitle="Variance Dimension";
    }
    else {
        YAxisTitle="Multifractal Features";
    }
    int DivFactor=2;
    if (pDoc->m_bViewRawSignalFlag) {
        DivFactor=3;
    }
    CFont RomanFont;
    RomanFont.CreateFont(2*m_TopTextMargin/DivFactor, 0, 900, 900, 400, FALSE, FALSE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    CFont* pOldFont=pDC->SelectObject(&RomanFont);
    CSize TitleSize=pDC->GetTextExtent(YAxisTitle);
    int TitleY=(m_VarDimHeight-TitleSize.cy)/2;
    pDC->TextOut(-m_LeftTextMargin,TitleY,YAxisTitle);
    pDC->SelectObject(pOldFont);
    pOldPen=pDC->SelectObject(&ThickBlackPen);
    pDC->MoveTo(m_WindowXDim-m_LeftTextMargin,m_VarDimHeight);
    pDC->LineTo(0,m_VarDimHeight);
    CFont SmallRomanFont;
    SmallRomanFont.CreateFont(m_TopTextMargin/2, 0, 0, 0, 400, FALSE, FALSE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    pOldFont=pDC->SelectObject(&SmallRomanFont);
    CString str;
    int YShift;
    for(int ctr=0;ctr<=100; ctr+=20) {
        str.Format("% .1f", (1.0+(double)(ctr/100.0)));
        TitleSize=pDC->GetTextExtent(str);
        YShift=TitleSize.cy/2;
        pDC->TextOut(-m_LeftTextMargin/2,
            ((double)ctr/100.0)*(double)m_VarDimHeight+YShift,str);
        pDC->MoveTo(-m_LeftTextMargin/10,
            ((double)ctr/100.0)*(double)m_VarDimHeight);
    }
}

```

```

        pDC->LineTo(m_LeftTextMargin/10,
            ((double)ctr/100.0)*(double)m_VarDimHeight);
    }
    pDC->SelectObject(pOldFont);

    pDC->SelectObject(&CyanSolidPen);
    double *VarDimTraj=pDoc->GetVarDimTraj();
    double hold;
    if (pDoc->m_bViewSegmentation) {
        hold=(VarDimTraj[0]-1.0)*m_VarDimHeight;
        pDC->MoveTo(0,hold);
        for(ctr=1;ctr<(m_ViewFileSize-pDoc->GetExtract()->GetWindowShift()-
            pDoc->GetExtract()->GetWindowSize());ctr++) {
            hold=(VarDimTraj[ctr]-1.0)*m_VarDimHeight;
            pDC->LineTo(ctr,hold);
        }
    }
    else {
        hold=(VarDimTraj[m_ViewStart]-1.0)*m_VarDimHeight;
        pDC->MoveTo(0,hold);
        for(ctr=1; ctr<m_ViewFileSize; ctr++) {
            if (ctr==2010)
            {
                hold=1;
            }
            hold=(VarDimTraj[ctr+m_ViewStart]-1.0)*m_VarDimHeight;
            pDC->LineTo(ctr,hold);
        }
    }
    pDC->SelectObject(pOldPen);
    return;
}
void CRawView::PlotRawSignal(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CPen *pOldPen;
    CPen RedSolidPen(PS_SOLID, 1, RGB(255,0,0));
    CPen ThickBlackPen(PS_SOLID, m_WindowXDim/500, RGB(0,0,0));
    int *RawSignal=pDoc->GetRawSignal();

    CString YAxisTitle="Raw Signal";
    int DivFactor=2;
    if (pDoc->m_bViewVarDimTrajFlag) {
        DivFactor=3;
    }
    CFont RomanFont;
    RomanFont.CreateFont(2*m_TopTextMargin/DivFactor, 0, 900, 900, 400, FALSE, FALSE, 0,
        ANSI_CHARSET, OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN, NULL);
    CFont* pOldFont=pDC->SelectObject(&RomanFont);
    CSize TitleSize=pDC->GetTextExtent(YAxisTitle);
    int TitleY=(m_RawSignalHeight-TitleSize.cy)/2;

```

```

pDC->TextOut(-m_LeftTextMargin, TitleY+m_VarDimHeight, YAxisTitle);
pDC->SelectObject(pOldFont);
pOldPen=pDC->SelectObject(&RedSolidPen);
pDC->MoveTo(0,m_VarDimHeight+RawSignal[m_ViewStart]/m_ZoomYCoord-m_ClipYCoord);
for(int ctr=1; ctr<m_ViewFileSize; ctr++) {
    pDC->LineTo(ctr,m_VarDimHeight+
        RawSignal[ctr+m_ViewStart]/m_ZoomYCoord-m_ClipYCoord);
}

if (pDoc->m_bViewSegmentation) {
    CPen BlueDashPen(PS_DASH, 1, RGB(0,0,128));
    int TransientBeginsHere=pDoc->GetTransientBeginsHere();
    int TransientSize=pDoc->GetExtract()->GetTransientSize();
    pDC->SelectObject(&BlueDashPen);

    pDC->MoveTo(TransientBeginsHere,m_RawSignalHeight+m_VarDimHeight);
    pDC->LineTo(TransientBeginsHere,0);
    pDC->MoveTo(TransientBeginsHere+TransientSize,
        m_RawSignalHeight+m_VarDimHeight);
    pDC->LineTo(TransientBeginsHere+TransientSize,0);
}
pDC->SelectObject(pOldPen);
return;
}
}

void CRawView::AdjustForMargins(CDC* pDC)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CPageSetupDialog* PSD=pDoc->GetPageSetupDlg();

    int PageWidth=PSD->m_psd.ptPaperSize.x-PSD->m_psd.rtMinMargin.left-
        PSD->m_psd.rtMinMargin.right;
    int PageLength=PSD->m_psd.ptPaperSize.y-PSD->m_psd.rtMinMargin.top
        -PSD->m_psd.rtMinMargin.bottom;
    int UsablePageWidth=PSD->m_psd.ptPaperSize.x-PSD->m_psd.rtMargin.left-
        PSD->m_psd.rtMargin.right;
    int UsablePageLength=PSD->m_psd.ptPaperSize.y-PSD->m_psd.rtMargin.top
        -PSD->m_psd.rtMargin.bottom;
    int NewWindowXDim=(double)m_WindowXDim*(double)PageWidth/
        (double)UsablePageWidth;
    int NewWindowYDim=(double)m_WindowYDim*(double)PageLength/
        (double)UsablePageLength;

    int LeftMarginDiff=PSD->m_psd.rtMargin.left-PSD->m_psd.rtMinMargin.left;
    int RightMarginDiff=PSD->m_psd.rtMargin.right-PSD->m_psd.rtMinMargin.right;
    int TopMarginDiff=PSD->m_psd.rtMargin.top-PSD->m_psd.rtMinMargin.top;
    int BottomMarginDiff=PSD->m_psd.rtMargin.bottom-
        PSD->m_psd.rtMinMargin.bottom;
    int LeftMarginLP=(double)(NewWindowXDim-m_WindowXDim)*(double)
        ((double)LeftMarginDiff/(double)(LeftMarginDiff+RightMarginDiff));
    int BottomMarginLP=(double)(NewWindowYDim-m_WindowYDim)*(double)
        ((double)BottomMarginDiff/(double)(BottomMarginDiff+TopMarginDiff));

    m-Origin.x=(double)(m_LeftTextMargin+LeftMarginLP)/

```

```

        (double)NewWindowXDim*(double)m_rectClient.right;
m_Origin.y=m_rectClient.bottom-(double)(m_BottomTextMargin+BottomMarginLP)/
        (double)NewWindowYDim*(double)m_rectClient.bottom;

CSize DataSize(NewWindowXDim,NewWindowYDim);
pDC->SetMapMode(MM_ANISOTROPIC);
pDC->SetWindowExt(DataSize);
pDC->SetViewportExt(m_rectClient.right, -m_rectClient.bottom);
pDC->SetViewportOrg(m_Origin);
}
void CRawView::SetViewFileSize()
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if(pDoc->m_bViewSegmentation) {
        m_ViewFileSize=pDoc->GetExtract()->GetRawFileSize();
        m_ViewStart=0;
        m_ZoomYCoord=8;
        m_ClipYCoord=0;
    }
    else {
        m_ViewFileSize=pDoc->GetExtract()->GetTransientSize();
        m_ViewStart=pDoc->GetTransientBeginsHere();
        m_ZoomYCoord=1+(pDoc->GetHighestValue()-pDoc->GetLowestValue())/8191;
        m_ClipYCoord=pDoc->GetLowestValue()/m_ZoomYCoord;
    }
}
void CRawView::OnViewSegmentation()
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->m_bViewSegmentation=TRUE;
    pDoc->ViewNeedsUpdating();
}
void CRawView::OnUpdateViewSegmentation(CCmdUI* pCmdUI)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    pCmdUI->Enable(pDoc->m_bFileInMemory&&!pDoc->m_bInBatchProcess);
    pCmdUI->SetCheck(pDoc->m_bViewSegmentation ? 1 : 0);
}
void CRawView::OnViewTransient()
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->m_bViewSegmentation=FALSE;
    pDoc->ViewNeedsUpdating();
}
void CRawView::OnUpdateViewTransient(CCmdUI* pCmdUI)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

```

```

    pCmdUI->Enable(pDoc->m_bFileInMemory&&!pDoc->m_bInBatchProcess);
    pCmdUI->SetCheck(pDoc->m_bViewSegmentation ? 0 : 1);
}
void CRawView::OnEditCopy()
{
    CBitmap BitmapClip;
    CClientDC ClientDC (this);
    CDC MemDC;
    RECT Rect;
    GetClientRect (&Rect);
    BitmapClip.CreateCompatibleBitmap (&ClientDC, Rect.right-Rect.left,
                                       Rect.bottom-Rect.top);
    MemDC.CreateCompatibleDC (&ClientDC);
    MemDC.SelectObject (&BitmapClip);
    MemDC.BitBlt(0, 0, Rect.right-Rect.left, Rect.bottom-Rect.top,
                &ClientDC, 0, 0, SRCCOPY);
    if(!OpenClipboard())
        return;
    ::EmptyClipboard();
    ::SetClipboardData(CF_BITMAP, BitmapClip.m_hObject);
    BitmapClip.Detach();
    ::CloseClipboard();
}
void CRawView::OnUpdateEditCopy(CCmdUI* pCmdUI)
{
    CTAC_MMDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    pCmdUI->Enable(pDoc->m_bFileInMemory);
}

```


Trandata.h

```

class CTransientData:public CObject
{
    DECLARE_SERIAL(CTransientData)

protected:
    CString m_strTransmitterMake;
    CString m_strTransmitterModel;
    CString m_strTransmitterSerial;
    CString m_Date;
    CString m_Time;
    CString m_strRawFilePath;
    int m_nTransientBeginsHere;
    int m_nTransientClass;
    int m_nTransientModelSize;
    double *m_TransientModel;
public:
    CTransientData() { }
    CTransientData(int TransientModelSize)
        { m_TransientModel=new double[TransientModelSize]; }
    ~CTransientData() { delete [] m_TransientModel; }
    void UpdateTransientModelSize(int NewModelSize)
        { delete [] m_TransientModel;
          m_TransientModel=new double[NewModelSize];
          m_nTransientModelSize=NewModelSize; }
    void Serialize(CArchive& ar);
    void SetRawFilePath(CString FilePath) { m_strRawFilePath=FilePath; }
    void SetTransientBeginsHere(int Here) { m_nTransientBeginsHere=Here; }
    void SetTransmitterMake(CString Make) { m_strTransmitterMake=Make; }
    void SetTransmitterModel(CString Model) { m_strTransmitterModel=Model; }
    void SetTransmitterSerial(CString Serial) { m_strTransmitterSerial=Serial; }
    void SetTransmitDate(CString Date) { m_Date=Date; }
    void SetTransmitTime(CString Time) { m_Time=Time; }
    void SetTransientModelSize(int TransientModelSize) { m_nTransientModelSize=
        TransientModelSize; }
    void SetTransientModel(double *TransientModel);
    void SetTransientClass(int TranClass) { m_nTransientClass=TranClass; }
    int GetTransientBeginsHere() { return m_nTransientBeginsHere; }
    CString GetRawFilePath() { return m_strRawFilePath; }
    CString GetTransmitterMake() { return m_strTransmitterMake; }
    CString GetTransmitterModel() { return m_strTransmitterModel; }
    CString GetTransmitterSerial() { return m_strTransmitterSerial; }
    CString GetTime() { return m_Time; }
    CString GetDate() { return m_Date; }
    double* GetTransientModel() { return m_TransientModel; }
    CString GetTitleLine();
    int GetTransientClass() { return m_nTransientClass; }
#ifdef _DEBUG
    void Dump(CDumpContext& dc) const;
#endif
};
typedef CTypedPtrArray<COBArray, CTransientData*> CTransientArray;

```

Trandata.cpp

```

#include "StdAfx.h"
#include "Trandata.h"

IMPLEMENT_SERIAL(CTransientData, CObject,0)
#ifdef _DEBUG
void CTransientData::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);
    dc << "\nm_strTransmitterMake = " << m_strTransmitterMake;
}
#endif

void CTransientData::Serialize(CArchive& ar)
{
    int ctr;

    if(ar.IsStoring()) {
        ar << m_Date << (LONG)m_nTransientBeginsHere << (LONG)m_nTransientClass
            << m_strRawFilePath << m_strTransmitterMake << m_strTransmitterModel
            << m_strTransmitterSerial << m_Time << m_nTransientModelSize;
        for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
            ar << m_TransientModel[ctr];
        }
    }
    else {
        ar >> m_Date >> (LONG&)m_nTransientBeginsHere >> (LONG&)m_nTransientClass
            >> m_strRawFilePath >> m_strTransmitterMake >> m_strTransmitterModel
            >> m_strTransmitterSerial >> m_Time >> m_nTransientModelSize;
        m_TransientModel=new double[m_nTransientModelSize];
        for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
            ar >> m_TransientModel[ctr];
        }
    }
}

CString CTransientData::GetTitleLine()
{
    CString TitleLine;

    TitleLine="Transmitter: "+m_strTransmitterMake;
    TitleLine+=" "+m_strTransmitterModel;
    if(m_strTransmitterSerial.IsEmpty()) {
        TitleLine+=" Serial #: Unknown";
    }
    else {
        TitleLine+=" Serial #: "+m_strTransmitterSerial;
    }
    return TitleLine;
}

void CTransientData::SetTransientModel(double *TransientModel)
{

```

Appendix B: TAC-MM Source Code

```
int ctr;

for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
    m_TransientModel[ctr]=TransientModel[ctr];
}
}
```

Extract.h

```

class CExtract:public CObject
{
    DECLARE_SERIAL(CExtract)
protected:
    int m_nRawFileSize;           //size of file with raw data.
    int m_nTransientSize;        //size of transient.
    int m_nSegmentationWindowSize;
    int m_nSegmentationVariancePairs;
    double m_Threshold; //abs diff from avg channel noise to trigger
                          //the start of transient

    int m_nModelWindowSize;
    int m_nModelVariancePairs;
    int m_nWindowShift;         //used for display purposes and data reduction

    int m_nVariancePairs;       //number of pairs to send to lsr routine.
    int m_nDyadic;              //set to 0 if unit intervals used, else dyadic.
    int m_nWindowSize;         //size of window for variance dimension calc.
    double *m_dSum1;
    double *m_dSum2;
public:
    CExtract() { }              //constructor can only be called once
    void Serialize(CArchive& ar);
    void SetParams(int RawFileSize, int TransientSize,
                  int SegmentationWindowSize,int SegmentationVariancePairs,
                  double Threshold, int ModelWindowSize,int ModelVariancePairs,
                  int WindowShift) {
        m_nRawFileSize=RawFileSize;
        m_nTransientSize=TransientSize;
        m_nSegmentationWindowSize=SegmentationWindowSize;
        m_nSegmentationVariancePairs=SegmentationVariancePairs;
        m_Threshold=Threshold;
        m_nModelWindowSize=ModelWindowSize;
        m_nModelVariancePairs=ModelVariancePairs;
        m_nWindowShift=WindowShift;
    }
    void SetWindowSize(int WindowSize) { m_nSegmentationWindowSize=WindowSize; }
    void SetVariancePairs(int VarPairs) { m_nSegmentationVariancePairs=VarPairs; }
    void SetThreshold(double Threshold) { m_Threshold=Threshold; }
    void SetModelWindowSize(int WindowSize) { m_nModelWindowSize=WindowSize; }
    void SetModelWindowShift(int WindowShift) { m_nWindowShift=WindowShift; }
    void SetModelVariancePairs(int VarPairs) { m_nModelVariancePairs=VarPairs; }
    int GetRawFileSize() { return m_nRawFileSize; }
    int GetTransientSize() { return m_nTransientSize; }
    int GetVariancePairs() { return m_nSegmentationVariancePairs; }
    int GetWindowSize() { return m_nSegmentationWindowSize; }
    int GetWindowShift() { return m_nWindowShift; }
    int GetModelWindowSize() { return m_nModelWindowSize; }
    int GetModelVariancePairs() { return m_nModelVariancePairs; }
    int GetModelWindowShift() { return m_nWindowShift; }
    double GetThreshold() { return m_Threshold; }
    int GetTransientModelSize() { return m_nTransientSize/m_nWindowShift; }

```

```
int FindTransient(int *RawSignal, double *VarDimTraj,
                 double *TransientModel);
void FillTransientModel(int *RawSignal, int TransientBeginsHere,
                       double *TransientModel);
void FillVarDimArray(int *RawSignal, double *VarDimTraj, bool ViewSegmentation,
                    int TransientBeginsHere, double *TransientModel);
double CalcVarDim(int *RawSignal, int StartLoc);
void Interpolate(double *VarDimTraj, int StartPoint);
void CExtract::FindMultiTransients(int *RawSignal, double *VarDimTraj,
                                   double **TransientModel, int NumModes,
                                   double *Thresholds, int MinSeperation,
                                   int *TransientBeginsHere);

#ifdef _DEBUG
    void Dump(CDumpContext& dc) const;
#endif
};
```

Extract.cpp

```

#include "StdAfx.h"
#include "Extract.h"
#include <math.h>
IMPLEMENT_SERIAL(CExtract, CObject,0)
#ifdef _DEBUG
void CExtract::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);
    dc << "nm_nRawFileSize = " << m_nRawFileSize;
}
#endif
void CExtract::Serialize(CArchive& ar)
{
    if(ar.IsStoring()) {
        ar<< (LONG)m_nRawFileSize << (LONG)m_nTransientSize
            << (LONG)m_nSegmentationWindowSize
            << (LONG)m_nSegmentationVariancePairs
            << m_Threshold << (LONG)m_nModelWindowSize
            << (LONG) m_nModelVariancePairs << (LONG)m_nWindowShift;
    }
    else {
        ar>> (LONG&)m_nRawFileSize >> (LONG&)m_nTransientSize
            >> (LONG&)m_nSegmentationWindowSize
            >> (LONG&)m_nSegmentationVariancePairs
            >> m_Threshold >> (LONG&)m_nModelWindowSize
            >> (LONG&) m_nModelVariancePairs >> (LONG&)m_nWindowShift;
    }
}
int CExtract::FindTransient(int *RawSignal,double *VarDimTraj,
                           double *TransientModel)
{
    m_nDyadic=0;
    m_nWindowSize=m_nSegmentationWindowSize;
    m_nVariancePairs=m_nSegmentationVariancePairs;
    double Mean=0.0, StdDev=0.0;
    int TransientBeginsHere= 1;
    int GoodFileSize=m_nRawFileSize-m_nWindowSize; //don't perform calcs
                                                    //past the end of file
    m_dSum1=new double[m_nSegmentationVariancePairs+1];
    m_dSum2=new double[m_nSegmentationVariancePairs+1];
    // Calculate the mean and the Standard Deviation of the VarDimTraj from
    // the start of the file up to 1/4 of the file size
    for (int ctr=0; ctr<m_nRawFileSize/4; ctr++) { //note windwoshift=1
        VarDimTraj[ctr]=CalcVarDim(RawSignal, ctr);
        Mean+=VarDimTraj[ctr];
        StdDev+=VarDimTraj[ctr]*VarDimTraj[ctr];
    }
    Mean/=(double)(m_nRawFileSize/4);
    StdDev=sqrt((StdDev/(double)(m_nRawFileSize/4))-(Mean*Mean));
    // Adjust the threshold based on the value of the standard deviation
    double Deviation=(m_Threshold/100.0)*Mean+StdDev;
}

```

```

for (ctr=m_nRawFileSize/4; ctr<GoodFileSize; ctr++) {
    VarDimTraj[ctr]=CalcVarDim(RawSignal, ctr);
    if(fabs(VarDimTraj[ctr]-Mean)>Deviation) {
        TRACE("Don, Transient is at %d\n",ctr);
        TransientBeginsHere=ctr;
        break;
    }
}
if (TransientBeginsHere>(m_nRawFileSize-m_nTransientSize-m_nModelWindowSize)) {
    TransientBeginsHere=-1;
}

if (m_Threshold==0.0)//used to circumvent segmentation procedure
    TransientBeginsHere=m_nRawFileSize/4;
if (TransientBeginsHere!=1) {
    for (ctr=TransientBeginsHere+1; ctr<GoodFileSize; ctr++) {
        VarDimTraj[ctr]=CalcVarDim(RawSignal, ctr);
    }
    FillTransientModel(RawSignal, TransientBeginsHere,
        TransientModel);
}

delete [] m_dSum1;
delete [] m_dSum2;
return TransientBeginsHere;
}
void CExtract::FillTransientModel(int *RawSignal, int TransientBeginsHere,
    double *TransientModel)
{
    if(m_nModelVariancePairs>0){           //transient model is fractal
        m_nDyadic=1;
        m_nWindowSize=m_nModelWindowSize;
        m_nVariancePairs=m_nModelVariancePairs;
        int ctr3=-1;
        for (int ctr=TransientBeginsHere;
            ctr<TransientBeginsHere+m_nTransientSize; ctr+=m_nWindowShift) {
            ctr3++;
            TransientModel[ctr3]=CalcVarDim(RawSignal, ctr);
        }
    }
    else {                                   //transient model is simple moving average
        int ctr2, ctr3=-1;
        double Average;
        for (int ctr=TransientBeginsHere;
            ctr<TransientBeginsHere+m_nTransientSize; ctr+=m_nWindowShift) {
            Average=0.0;
            for (ctr2=0; ctr2<m_nWindowShift; ctr2++) {
                Average+=RawSignal[ctr+ctr2];
            }
            ctr3++;
            TransientModel[ctr3]=Average/m_nWindowShift/65536.0+1.0;
        }
    }
}
}

```

```

void CExtract::FillVarDimArray(int *RawSignal,double *VarDimTraj,
                             bool ViewSegmentation, int TransientBeginsHere,
                             double *TransientModel)
{
    if(ViewSegmentation) {
        m_nDyadic=0;
        m_nWindowSize=m_nSegmentationWindowSize;
        m_nVariancePairs=m_nSegmentationVariancePairs;
        int GoodFileSize=m_nRawFileSize-m_nWindowSize;    //don't perform calcs
                                                           //past the end of file
        m_dSum1=new double[m_nSegmentationVariancePairs+1];
        m_dSum2=new double[m_nSegmentationVariancePairs+1];
        for (int ctr=0; ctr<GoodFileSize; ctr+=1) {
            VarDimTraj[ctr]=CalcVarDim(RawSignal, ctr);
        }
        delete [] m_dSum1;
        delete [] m_dSum2;
    }
    else {
        //pull VarDim from record... interpolate
        int ctr3=-1;
        for (int ctr=TransientBeginsHere;
            ctr<TransientBeginsHere+m_nTransientSize; ctr+=m_nWindowShift) {
            ctr3++;
            VarDimTraj[ctr]=TransientModel[ctr3];
        }
        VarDimTraj[ctr]=TransientModel[ctr3];    //fill last spot with same
                                                //vardim for interpolation
        Interpolate(VarDimTraj, TransientBeginsHere);
    }
}

double CExtract::CalcVarDim(int *RawSignal, int StartLoc)
{
    double *LogOfVariance, *LogOfDeltaT;
    double Sum1,Sum2,DeltaB;
    double DimensionPower;
    double Var, VarDim;
    int k,t,NoOfDeltaBs,NoOfZeroes;
    LogOfVariance=new double[m_nVariancePairs+1];
    LogOfDeltaT=new double[m_nVariancePairs+1];

    /* MSG message;
    if (::PeekMessage(&message, NULL, 0, 0, PM_REMOVE)) {
        ::TranslateMessage(&message);
        ::DispatchMessage(&message);
    }
    AfxGetApp()->DoWaitCursor(1);
    */

    NoOfZeroes=0;
    if (m_nDyadic==0) {
        for (t=1; t<=m_nVariancePairs; t++) {
            NoOfDeltaBs=m_nWindowSize-t;
            if (StartLoc==0) {    //must perform all calcs for first window
                m_dSum1[t]=0.0;
                m_dSum2[t]=0.0;
            }
        }
    }
}

```



```

        for(k=StartLoc; k<(StartLoc+NoOfDeltaBs); k++) {
            DeltaB=RawSignal[k+t]-RawSignal[k];
            m_dSum1[t]+=DeltaB*DeltaB;
            m_dSum2[t]+=DeltaB;
        }
    }
    else {
        DeltaB=RawSignal[StartLoc-1+t]-RawSignal[StartLoc-1];
        m_dSum1[t]-=DeltaB*DeltaB;
        m_dSum2[t]-=DeltaB;
        DeltaB=RawSignal[StartLoc+NoOfDeltaBs-1+t]
            -RawSignal[StartLoc+NoOfDeltaBs-1];
        m_dSum1[t]+=DeltaB*DeltaB;
        m_dSum2[t]+=DeltaB;
    }
    Var=(m_dSum1[t]-(m_dSum2[t]*m_dSum2[t])/((double)NoOfDeltaBs))
        /((double)NoOfDeltaBs-1.0);
    if (Var==0.0) {
        NoOfZeroes++;
    }
    else {
        LogOfVariance[t-NoOfZeroes]=log(Var);
        LogOfDeltaT[t-NoOfZeroes]=log((double)t);
    }
}
}
else {
    for (t=1; t<=m_nVariancePairs; t++) {
        Sum1=0.0;
        Sum2=0.0;
        DimensionPower=pow(2.0,(double)(t));
        NoOfDeltaBs=m_nWindowSize-(int)DimensionPower;
        for(k=StartLoc; k<(StartLoc+NoOfDeltaBs); k++) {
            DeltaB=RawSignal[k+(int)DimensionPower]-RawSignal[k];
            Sum1+=DeltaB*DeltaB;
            Sum2+=DeltaB;
        }
        if(NoOfDeltaBs!=0)//protect against divide by zero
            Var=(Sum1-(Sum2*Sum2)/((double)NoOfDeltaBs))
                /((double)NoOfDeltaBs-1.0);
        else
            Var=0.0;
        if (Var==0.0) {
            NoOfZeroes++;
        }
        else {
            LogOfVariance[t-NoOfZeroes]=log(Var);
            LogOfDeltaT[t-NoOfZeroes]=log(DimensionPower);
        }
    }
}
double sum1=0.0, sum2=0.0, sum3=0.0, sum4=0.0, sum5=0.0;

for (int ctr=1; ctr<=(m_nVariancePairs-NoOfZeroes); ctr++)

```

```

    {
        sum1+=LogOfDeltaT[ctr]*LogOfVariance[ctr];
        sum2+=LogOfDeltaT[ctr];
        sum3+=LogOfVariance[ctr];
        sum4+=LogOfDeltaT[ctr]*LogOfDeltaT[ctr];
        sum5+=LogOfDeltaT[ctr];
    }
//if (m_nDyadic==1)           //these modifications (4 lines) were made to implement the fractal
//m_nVariancePairs++;        //transformation discussed in Subsection 6.2.6
    double Slope=(m_nVariancePairs*sum1-sum2*sum3)
        /(m_nVariancePairs*sum4-sum5*sum5);
    VarDim=2.0-0.5*Slope;
//if (m_nDyadic==1)
//m_nVariancePairs--;
    delete [] LogOfVariance;
    delete [] LogOfDeltaT;
    return VarDim;
}
void CExtract::Interpolate(double *VarDimTraj, int StartPoint)    //linear interpolation
                                                                //for display purposes only
{
    int VarDimFileSize;
#ifdef _DEBUG
    int Count=0;
    double Mean=0.0, StdDev=0.0;
#endif
    if (StartPoint==0)
        VarDimFileSize=(m_nRawFileSize-m_nWindowSize);
    else
        VarDimFileSize=StartPoint+m_nTransientSize;
    for (int ctr=StartPoint;ctr<VarDimFileSize; ctr+=m_nWindowShift) {
#ifdef _DEBUG
        Count++;
        Mean+=VarDimTraj[ctr];
        StdDev+=VarDimTraj[ctr]*VarDimTraj[ctr];
#endif
        for (int ctr2=1; ctr2<m_nWindowShift; ctr2++) {
            VarDimTraj[ctr+ctr2]=VarDimTraj[ctr]+(float)((float)ctr2/(float)
                m_nWindowShift)*(VarDimTraj[ctr+m_nWindowShift]-VarDimTraj[ctr]);
        }
    }
#ifdef _DEBUG
    Mean/=(double)Count;
    StdDev=sqrt((StdDev/(double)(Count))-(Mean*Mean));
    TRACE("\nDon...Average from sample %d is %f", StartPoint, Mean);
    TRACE("\nDon...Standard Deviation is %f", StdDev);
#endif
}
void CExtract::FindMultiTransients(int *RawSignal, double *VarDimTraj,
    double **TransientModel, int NumModes,
    double *Thresholds, int MinSeperation,
    int *TransientBeginsHere)
{
    m_nDyadic=0;
    m_nWindowSize=m_nSegmentationWindowSize;
    m_nVariancePairs=m_nSegmentationVariancePairs;
}

```

```

double Mean=0.0, StdDev=0.0;
int ctr, ctr1, Start=m_nRawFileSize/4;
int GoodFileSize=m_nRawFileSize-m_nWindowSize; //don't perform calcs
                                                //past the end of file

m_dSum1=new double[m_nSegmentationVariancePairs+1];
m_dSum2=new double[m_nSegmentationVariancePairs+1];
// Calculate the mean and the Standard Deviation of the VarDimTraj from
// the start of the file to 1/4 of the file size
for (ctr=0; ctr<m_nRawFileSize/4; ctr++) { //note windwoshift=1
    VarDimTraj[ctr]=CalcVarDim(RawSignal, ctr);
    Mean+=VarDimTraj[ctr];
    StdDev+=VarDimTraj[ctr]*VarDimTraj[ctr];
}
Mean/=(double)(m_nRawFileSize/4);
StdDev=sqrt((StdDev/(double)(m_nRawFileSize/4))-(Mean*Mean));

for(ctr1=0; ctr1<NumModes; ctr1++) {
    TransientBeginsHere[ctr1]=-1;

// Adjust the threshold based on the value of the standard deviation
double Deviation=(Thresholds[ctr1]/100.0)*Mean+StdDev;
for (ctr=Start; ctr<GoodFileSize; ctr++) {
    VarDimTraj[ctr]=CalcVarDim(RawSignal, ctr);
    if(fabs(VarDimTraj[ctr]-Mean)>Deviation) {
        TRACE("Don, Transient is at %d\n",ctr);
        TransientBeginsHere[ctr1]=ctr;
        break;
    }
}
if (TransientBeginsHere[ctr1]
    >(m_nRawFileSize-m_nTransientSize-m_nModelWindowSize)) {
    TransientBeginsHere[ctr1]=-1;
}
if (TransientBeginsHere[ctr1]!=-1) {
    FillTransientModel(RawSignal, TransientBeginsHere[ctr1],
        TransientModel[ctr1]);
    Start=TransientBeginsHere[ctr1]+MinSeperation;
}
}
delete [] m_dSum1;
delete [] m_dSum2;
return;
}

```

PNN.h

```

#include "TrainingResults.h"
class CPNN:public CObject
{
    DECLARE_SERIAL(CPNN)
protected:
    int m_nTransientModelSize;        //init in constructor
    int m_nNumCases;
    int m_nNumClasses;
    double *m_Sigma;
    BOOL m_bSigmasInitialized;
    BOOL m_bSigmasOptimized;
    double m_dBestError;
    double m_LowerSearchLimit;
    double m_UpperSearchLimit;
    CTransientArray *m_TransientArray;
public:
    BOOL m_bStopNow;//these 4 vars must be accessible by
    double m_nUserDisplaySigma;//the dialog class that updates the user
    double m_dUserDisplayError;
    int m_nUserDisplayDiscreteError;
    double m_dUserDisplayImprovement;
    CPNN() { m_Sigma=new double[1]; } //in case never init
    ~CPNN() { delete [] m_Sigma; }
    void Initialize(int TransientModelSize);
    void Serialize(CArchive& ar);
    int Classify(double *Unknown, double *SummationNeuron, int Skip,
                double RejectThreshold, double *Activation);
    BOOL TrainSigmasInit(int NumSearchPoints, BOOL GlobalDone,
                        double *LowerSigma, double *LowerError, double *MiddleSigma,
                        double *MiddleError, double *UpperSigma, double *UpperError);
    void SetNumCases(int NumCases) { m_nNumCases=NumCases; }
    void SetNumClasses(int NumClasses) { m_nNumClasses=NumClasses; }
    BOOL GetSigmasInitialized() { return m_bSigmasInitialized; }
    BOOL GetSigmasOptimized() { return m_bSigmasOptimized; }
    int GetNumClasses() { return m_nNumClasses; }
    double GetFirstSigma() { return m_Sigma[0]; }
    void SetLowerSearchLimit(double LowerSearchLimit)
        { m_LowerSearchLimit=LowerSearchLimit; }
    void SetUpperSearchLimit(double UpperSearchLimit)
        { m_UpperSearchLimit=UpperSearchLimit; }
    BOOL GlobMin(int NumSearchPoints, double *LowerSigma, double *LowerError,
                double *MiddleSigma, double *MiddleError, double *UpperSigma,
                double *UpperError);
    double FindError(double Sigma, int *DiscreteError);
    void SetTransientArrayPointer(CTransientArray *TransientArray)
        { m_TransientArray=TransientArray; }
    void SetSigmasInitialized(BOOL Init) { m_bSigmasInitialized=Init; }
    void SetSigmasOptimized(BOOL Opt) { m_bSigmasOptimized=Opt; }
    CString m_strUserMessage;
    void GoldMin(double *LowerSigma, double *LowerError, double *MiddleSigma,
                double *MiddleError, double *UpperSigma, double *UpperError);

```

```

int FindDerivs(double *Unknown, double *SummationNeuron,
    int Skip, double *Deriv, double *Deriv2);
double CumulateErrorAndDerivs(double *Sigma, double *Deriv,
    double *Deriv2, BOOL CumulateDerivs, int *DiscreteError);
BOOL ConjGradientsMin(double ConvergenceTolerance);
double FindGamma(double *g, double *Grad);
void FindNewDir(double Gamma, double *g, double *h, double *Grad);
double UniVarError(double Point, double *Sigma, double *Base, double *Deriv);
BOOL CGGlobMin(int NumSearchPoints, double *LowerPoint, double *LowerError,
    double *MiddlePoint, double *MiddleError, double *UpperPoint,
    double *UpperError, double *Sigma, double *Base,
    double *Direc);
double BrentMin(int ItMax, double *LowerPoint, double *MiddlePoint,
    double *UpperPoint, double *MiddleError, double *Sigma,
    double *Base, double *Direc);
BOOL TrainSigmasOpt(double Tolerance);
void FillTrainingResultsArray(CResultsArray *ResultsArray);
#ifdef _DEBUG
    void Dump(CDumpContext& dc) const;
#endif
};

```

PNN.cpp

Some of the routines in this file have been adapted from code in the book *Advanced Algorithms for Neural Networks: A C++ Sourcebook* [Mast95]; Copyright (c) 1995 John Wiley & Sons, Inc.

```

#include "StdAfx.h"
#include "Trandata.h"
#include "PNN.h"
#include <math.h>
IMPLEMENT_SERIAL(CPNN, CObject,0)
#ifdef _DEBUG
void CPNN::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);
    dc << "Hi Don... CPNN Dump.";
}
#endif
void CPNN::Initialize(int TransientModelSize)
{
    m_nTransientModelSize=TransientModelSize;

    delete [] m_Sigma; //had to do dummy init in constructor...undo it now
    m_Sigma=new double[m_nTransientModelSize];
    m_Sigma[0]=1.0;
    m_dBestError=-999.0;
    m_dUserDisplayError=m_dBestError;
    m_nUserDisplayDiscreteError=-999;
    m_bSigmasInitialized=FALSE;
    m_bSigmasOptimized=FALSE;
}
void CPNN::Serialize(CArchive& ar)
{
    int ctr;
    if(ar.IsStoring()) {
        ar << m_bSigmasInitialized << m_bSigmasOptimized << m_dBestError
            << m_nTransientModelSize << m_nNumCases << m_nNumClasses;
        for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
            ar << m_Sigma[ctr];
        }
    }
    else {
        ar >> m_bSigmasInitialized >> m_bSigmasOptimized >> m_dBestError
            >> m_nTransientModelSize >> m_nNumCases >> m_nNumClasses;
        delete [] m_Sigma; //had to do dummy init in constructor...undo it now
        m_dUserDisplayError=m_dBestError;
        m_Sigma=new double[m_nTransientModelSize];
        for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
            ar >> m_Sigma[ctr];
        }
    }
}
int CPNN::Classify(double *Unknown, double *SummationNeuron, int Skip,
    double RejectThreshold, double *Activation)

```

```

{
//make sure m_nNumClasses and m_nNumCases are set prior to calling
int TransientClass, TrgCase, UnknownClass, *NoSamplesInClass;
double *TrgSetTransient, Distance, Diff, Best, psum;
NoSamplesInClass=new int[m_nNumClasses];
for (int ctr=0; ctr<m_nNumClasses; ctr++) {
    SummationNeuron[ctr]=0.0;           // will sum kernels here
    NoSamplesInClass[ctr]=0;
}
for (TrgCase=0; TrgCase<m_nNumCases; TrgCase++) {
    if (TrgCase!=Skip) {               //for trg purposes, don't compare same transients
                                        //skip will be -1 in classify mode...nothing skipped
        TrgSetTransient=m_TransientArray->
            GetAt(TrgCase)->GetTransientModel();
        TransientClass=m_TransientArray->
            GetAt(TrgCase)->GetTransientClass();
        Distance=0.0;                   // Will sum distance here
        for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
            Diff = Unknown[ctr]-TrgSetTransient[ctr];
            Diff /= m_Sigma[ctr]; // Scale per sigma
            Distance+=Diff*Diff; // Cumulate Euclidean distance
        }
        SummationNeuron[TransientClass]+=exp(-Distance); //Gaussian Kernal
        //SummationNeuron[TransientClass]+=1.0/(1.0+Distance*Distance); //other kernal
        NoSamplesInClass[TransientClass]+=1;
    }
}
//decide if we want to account for prior probabilities in this loop
psum=0.0;
for(ctr=0; ctr<m_nNumClasses; ctr++) {
    if(NoSamplesInClass[ctr]!=0)
        SummationNeuron[ctr]/=NoSamplesInClass[ctr]; //account for unequal trg
                                                    //sample representation
    psum+=SummationNeuron[ctr];
}
Best=0.0;                               // Keep track of max across pops
UnknownClass=-1; //if all summation neurons equal 0, then give impossible answer
for (ctr=0 ;ctr<m_nNumClasses; ctr++) {
    if (SummationNeuron[ctr]>Best) {       // find the highest activation
        Best=SummationNeuron[ctr] ;
        UnknownClass=ctr;
    }
}
*Activation=Best;                        //for multimodal classification

if (Skip==1 && Best<RejectThreshold)
    UnknownClass=-1;                     //reject cuz activation too low

if(psum<1.e-40)
    psum=1.e-40;
for (ctr=0; ctr<m_nNumClasses; ctr++) {
    SummationNeuron[ctr]/=psum;          //normalize
}
delete [] NoSamplesInClass;

```

```

    return UnknownClass;
}
BOOL CPNN::TrainSigmasInit(int NumSearchPoints, BOOL GlobalDone,
    double *LowerSigma, double *LowerError,
    double *MiddleSigma, double *MiddleError,
    double *UpperSigma, double *UpperError)
{
    BOOL IsNotSuccessful;

    m_bStopNow=FALSE;
    if(!GlobalDone) {
        m_strUserMessage.Format("In Global Minimization Algorithm.");
        m_nUserDisplaySigma=m_LowerSearchLimit;
        m_dUserDisplayError=m_dBestError;
        m_nUserDisplayDiscreteError=-999;
        IsNotSuccessful=GlobMin(NumSearchPoints, LowerSigma, LowerError,
            MiddleSigma, MiddleError, UpperSigma, UpperError);
        if (IsNotSuccessful)
            return FALSE;        //GlobMin Unsuccessful
    }
    m_bSigmasInitialized=TRUE;

    m_strUserMessage.Format("In Golden Section Minimization Algorithm.");
    GoldMin(LowerSigma, LowerError,
        MiddleSigma, MiddleError, UpperSigma, UpperError);
    return TRUE;
}
BOOL CPNN::GlobMin(int NumSearchPoints, double *LowerSigma, double *LowerError,
    double *MiddleSigma, double *MiddleError, double *UpperSigma,
    double *UpperError)
{
    int ctr, ibest, CurrentDiscreteError, PreviousDiscreteError,
        LowerDiscreteError, MiddleDiscreteError, UpperDiscreteError;
    double CurrentSigma, CurrentError, Rate, Previous=0.0;
    BOOL Turned_Up=FALSE, UserQuit=FALSE;
    Rate=exp(log(m_UpperSearchLimit/m_LowerSearchLimit)/(NumSearchPoints-1));
    CurrentSigma=m_LowerSearchLimit;
    ibest = -1 ; // For proper escape if error reaches 0
    for (ctr=0; ctr<NumSearchPoints; ctr++) {
        CurrentError=FindError(CurrentSigma, &CurrentDiscreteError);

        if ((ctr==0)||((CurrentError<*MiddleError)) { // Keep track of best here
            ibest=ctr ;
            *MiddleSigma=CurrentSigma;
            *MiddleError=CurrentError;
            MiddleDiscreteError=CurrentDiscreteError;
            *LowerError=Previous; // Function value to its left
            LowerDiscreteError=PreviousDiscreteError;
            Turned_Up=FALSE; // Flag that min is not yet bounded
        }
        else if (ctr==(ibest+1)) { // Didn't improve so this point may
            *UpperError=CurrentError; // be the right neighbor of the best
            UpperDiscreteError=CurrentDiscreteError;
            Turned_Up=TRUE; // Flag that min is bounded
        }
    }
}

```



```

}
Previous=CurrentError;           // Keep track for left neighbor of best
PreviousDiscreteError=CurrentDiscreteError;
if ((MiddleDiscreteError==0)&&(ibest>0)&&Turned_Up)
    break; // Done if good enough and both neighbors found
if (m_bStopNow) {
    UserQuit=TRUE;
    break;
}
CurrentSigma*=Rate;
}
*LowerSigma=*MiddleSigma/Rate;
*UpperSigma=*MiddleSigma*Rate;
if (!Turned_Up) { // Must extend to the right (larger x)
    for (;;) { // Endless loop goes as long as necessary
        *UpperError=FindError(*UpperSigma, &UpperDiscreteError);
        if (*UpperError>*MiddleError) // If function increased we are done
            break;
        if ((*LowerError==*MiddleError)&&(*MiddleError==*UpperError))
            break; // Give up if flat
        if (m_bStopNow) {
            UserQuit=TRUE;
            break;
        }
        *LowerSigma=*MiddleSigma; // Shift all points
        *LowerError=*MiddleError;
        LowerDiscreteError=MiddleDiscreteError;
        *MiddleSigma=*UpperSigma;
        *MiddleError=*UpperError;
        MiddleDiscreteError=UpperDiscreteError;
        Rate*=3.0; // Step further each time
        *UpperSigma*=Rate;
    }
}
}
else if(ibest==0) { // Must extend to the left (smaller x)
    for (;;) { // Endless loop goes as long as necessary
        *LowerError=FindError(*LowerSigma, &LowerDiscreteError);
        if (*LowerError>*MiddleError) // If function increased we are done
            break;
        if ((*LowerError==*MiddleError)&&(*MiddleError==*UpperError))
            break; // Give up if flat
        if (m_bStopNow) {
            UserQuit=TRUE;
            break;
        }
        *UpperSigma=*MiddleSigma; // Shift all points
        *UpperError=*MiddleError;
        UpperDiscreteError=MiddleDiscreteError;
        *MiddleSigma=*LowerSigma;
        *MiddleError=*LowerError;
        MiddleDiscreteError=LowerDiscreteError;
        Rate*=3.0; // Step further each time
        *LowerSigma/=Rate;
    }
}
}

```

```

    }
    m_LowerSearchLimit=*LowerSigma;
    m_UpperSearchLimit=*UpperSigma;
    m_dBestError=*MiddleError;
    m_dUserDisplayError=*MiddleError;    //for user update on timer 1
    m_nUserDisplayDiscreteError=MiddleDiscreteError;

    for(ctr=0; ctr<m_nTransientModelSize; ctr++) {
        m_Sigma[ctr]=*MiddleSigma;
    }
    return UserQuit;
}
double CPNN::FindError(double Sigma, int *DiscreteError)
{
    MSG message;
    int PredictedClass, CorrectClass, ctr1;
    double *CurrentTransient, *SummationNeuron, TotalError=0.0, Diff, Error, Act;
    *DiscreteError=0;
    SummationNeuron=new double[m_nNumClasses];
    for(int ctr=0; ctr<m_nTransientModelSize; ctr++) {
        m_Sigma[ctr]=Sigma;
    }
    for(ctr=0; ctr<m_nNumCases; ctr++) {
        CurrentTransient=m_TransientArray->
            GetAt(ctr)->GetTransientModel();
        PredictedClass=Classify(CurrentTransient, SummationNeuron, ctr,NULL,&Act);
        CorrectClass=m_TransientArray->GetAt(ctr)->GetTransientClass();
        if (PredictedClass!=CorrectClass) {
            *DiscreteError+=1;
        }
        Error=0.0;
        for (ctr1=0; ctr1<m_nNumClasses; ctr1++) {
            if (ctr1==CorrectClass) {
                Diff=1.0-SummationNeuron[ctr1];
                Error+=Diff*Diff;
            }
            else {
                Error+=SummationNeuron[ctr1]*SummationNeuron[ctr1];
            }
        }
        TotalError+=Error;
        if (::PeekMessage(&message, NULL, 0, 0, PM_REMOVE)) {
            ::TranslateMessage(&message);
            ::DispatchMessage(&message);
        }
        AfxGetApp()->DoWaitCursor(1);
    }
    TotalError/=m_nNumCases;

    m_nUserDisplayDiscreteError=*DiscreteError;    //for user output on timer 1
    m_nUserDisplaySigma=Sigma;
    m_dUserDisplayError=TotalError;
    delete [] SummationNeuron;
    return TotalError;
}

```

```

}
void CPNN::GoldMin(double *LowerSigma, double *LowerError, double *MiddleSigma,
    double *MiddleError, double *UpperSigma, double *UpperError)
{
    int CurrentDiscreteError, MiddleDiscreteError;
    double LeftWidth, RightWidth, CurrentSigma, CurrentError;
    double Gold=2.0/(1.0+sqrt(5.0)); //about 0.618 or (1.0 - 0.38197)
    for (;;) { // Endless loop goes as long as necessary
        if (m_bStopNow||*MiddleSigma==0) {
            break;
        }
        if ((*UpperSigma-*LowerSigma)<(3.e-8**MiddleSigma)) {
            break; //avoid refining beyond double precision
        }
        LeftWidth=*MiddleSigma/ *LowerSigma;
        RightWidth=*UpperSigma/ *MiddleSigma;
        if (LeftWidth>RightWidth) { //left interval larger so split it
            CurrentSigma=exp(log(*LowerSigma)+Gold*log(*MiddleSigma/ *LowerSigma));
            CurrentError=FindError(CurrentSigma, &CurrentDiscreteError);
            if ((CurrentError<*MiddleError)||((CurrentError==*MiddleError)
                &&(*LowerError<*UpperError))) { //if we improved, or tied with left
                //side favored, discard right point

                *UpperSigma=*MiddleSigma;
                *UpperError=*MiddleError;
                *MiddleSigma=CurrentSigma;
                *MiddleError=CurrentError;
                MiddleDiscreteError=CurrentDiscreteError;
            }
            else { //didn't improve so discard left point
                *LowerSigma=CurrentSigma;
                *LowerError=CurrentError;
            }
        }
        else {
            CurrentSigma=exp(log(*UpperSigma)+Gold*log(*MiddleSigma/ *UpperSigma));
            CurrentError=FindError(CurrentSigma, &CurrentDiscreteError);
            if ((CurrentError<*MiddleError)||((CurrentError==*MiddleError)
                &&(*LowerError>*UpperError))) { //if we improved, or tied with right
                //side favored, discard left point

                *LowerSigma=*MiddleSigma;
                *LowerError=*MiddleError;
                *MiddleSigma=CurrentSigma;
                *MiddleError=CurrentError;
                MiddleDiscreteError=CurrentDiscreteError;
            }
            else { //didn't improve so discard right point
                *UpperSigma=CurrentSigma;
                *UpperError=CurrentError;
            }
        }
    }
    m_dBestError=*MiddleError;
    m_dUserDisplayError=*MiddleError; //for user update on timer 1
}

```

```

    m_nUserDisplayDiscreteError=MiddleDiscreteError;
}
int CPNN::FindDerivs(double *Unknown, double *SummationNeuron,
                    int Skip, double *Deriv, double *Deriv2)
{
    double *VPtr, *WPtr, VSum, WSum;
    double Temp, Der1, Der2, psum;
    int TransientClass, TrgCase, UnknownClass, *NoSamplesInClass;
    double *TrgSetTransient, *v, *w, *DiffSqr, Distance, TrueDist, Diff, Best;
    v=new double[m_nTransientModelSize*m_nNumClasses];
    w=new double[m_nTransientModelSize*m_nNumClasses];
    DiffSqr=new double[m_nTransientModelSize];
    NoSamplesInClass=new int[m_nNumClasses];

    for (int ctr=0; ctr<m_nNumClasses; ctr++) {
        SummationNeuron[ctr]=0.0;// will sum kernels here
        NoSamplesInClass[ctr]=0;
        for (int ivar=0; ivar<m_nTransientModelSize; ivar++) {
            v[ctr*m_nTransientModelSize+ivar]=0.0; //Scratch for deriv calc
            w[ctr*m_nTransientModelSize+ivar]=0.0; //Ditto
        }
    }
    for (TrgCase=0; TrgCase<m_nNumCases; TrgCase++) {
        if (TrgCase!=Skip) { //for trg purposes, don't compare same transients
            //skip will be -1 in classify mode...nothing skipped
            TrgSetTransient=m_TransientArray->
                GetAt(TrgCase)->GetTransientModel();
            TransientClass=m_TransientArray->
                GetAt(TrgCase)->GetTransientClass();
            Distance=0.0; // Will sum distance here
            for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
                Diff=Unknown[ctr]-TrgSetTransient[ctr];
                Diff/=m_Sigma[ctr]; // Scale per sigma
                DiffSqr[ctr]=Diff*Diff; // Squared weighted distance
                Distance+=DiffSqr[ctr]; // Cumulate for all vars
            }
            Distance=exp(-Distance); //Gaussian kernal
            //Distance=1.0/(1.0+Distance*Distance); //other kernal
            TrueDist=Distance;
            //
            //
            Distance=1.e-40;
            SummationNeuron[TransientClass]+=Distance;
            NoSamplesInClass[TransientClass]+=1;
            VPtr=v+TransientClass*m_nTransientModelSize;//Point to this row in v
            WPtr=w+TransientClass*m_nTransientModelSize;//And w
            for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
                Temp=TrueDist*DiffSqr[ctr];
                VPtr[ctr]+=Temp;
                WPtr[ctr]+=Temp*(2.0*DiffSqr[ctr]-3.0);
            }
        }
    }
    //decide if we want to account for prior probabilities in this loop
    psum=0.0;
}

```

```

for(ctr=0; ctr<m_nNumClasses; ctr++) {
    if(NoSamplesInClass[ctr]!=0)
        SummationNeuron[ctr]/=NoSamplesInClass[ctr];    //account for unequal trg
                                                         //sample representation

    psum+=SummationNeuron[ctr];
}
if(psum<1.e-40)
    psum=1.e-40;
for (ctr=0; ctr<m_nNumClasses; ctr++) {
    SummationNeuron[ctr]/=psum;
}
//Compute the derivatives. VSum and WSum are the simple sums of v and w.
for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
    VSum=WSum=0.0;
    for(int OutVar=0; OutVar<m_nNumClasses; OutVar++) {    //Cumulate VSum and WSum
        v[OutVar*m_nTransientModelSize+ctr]*=
            2.0/(psum*m_Sigma[ctr]*NoSamplesInClass[OutVar]);
        w[OutVar*m_nTransientModelSize+ctr]*=
            2.0/(psum*m_Sigma[ctr]*m_Sigma[ctr]*NoSamplesInClass[OutVar]);
        VSum+=v[OutVar*m_nTransientModelSize+ctr];
        WSum+=w[OutVar*m_nTransientModelSize+ctr];
    }

    for (OutVar=0; OutVar<m_nNumClasses; OutVar++) {
        Der1=v[OutVar*m_nTransientModelSize+ctr]
            -SummationNeuron[OutVar]*VSum;
        Der2=w[OutVar*m_nTransientModelSize+ctr]
            +2.0*SummationNeuron[OutVar]*VSum*VSum
            -2.0*v[OutVar*m_nTransientModelSize+ctr]*VSum
            -SummationNeuron[OutVar]*WSum;

        if (OutVar==m_TransientArray->GetAt(Skip)->GetTransientClass()) {
            Temp=2.0*(SummationNeuron[OutVar]-1.0);
        }
        else {
            Temp=2.0*SummationNeuron[OutVar];
        }

        Deriv[ctr]+=Temp*Der1;
        Deriv2[ctr]+=Temp*Der2+2.0*Der1*Der1;
    }
}
Best=0.0;    // Keep track of max across pops
UnknownClass=-1;//if all summation neurons equal 0, then give impossible answer
for (ctr=0 ;ctr<m_nNumClasses; ctr++) {
    if (SummationNeuron[ctr]>Best) {    // find the highest activation
        Best=SummationNeuron[ctr];
        UnknownClass=ctr;
    }
}
delete [] v;
delete [] w;
delete [] DiffSqr;
delete [] NoSamplesInClass;

```

```

    return UnknownClass;
}
double CPNN::CumulateErrorAndDerivs(double *Sigma, double *Deriv,
                                   double *Deriv2, BOOL CumulateDerivs,
                                   int *DiscreteError)
{
    MSG message;
    int PredictedClass, CorrectClass, ctr1;
    double *CurrentTransient, *SummationNeuron, TotalError=0.0, Diff, Error,Act;
    *DiscreteError=0,

    SummationNeuron=new double[m_nNumClasses];
    for(int ctr=0; ctr<m_nTransientModelSize; ctr++) {
        m_Sigma[ctr]=Sigma[ctr];
    }
    if(CumulateDerivs) {
        for(ctr=0; ctr<m_nTransientModelSize; ctr++) {
            Deriv[ctr]=0.0;
            Deriv2[ctr]=0.0;
        }
    }
    for(ctr=0; ctr<m_nNumCases; ctr++) {
        CurrentTransient=m_TransientArray->
            GetAt(ctr)->GetTransientModel();
        if(CumulateDerivs) {
            PredictedClass=FindDerivs(CurrentTransient, SummationNeuron, ctr,
                Deriv, Deriv2);
        }
        else {
            PredictedClass=Classify(CurrentTransient,SummationNeuron,ctr,NULL,&Act);
        }
        CorrectClass=m_TransientArray->GetAt(ctr)->GetTransientClass();
        if (PredictedClass!=CorrectClass) {
            *DiscreteError+=1;
        }
        Error=0.0;
        for (ctr1=0; ctr1<m_nNumClasses; ctr1++) {
            if (ctr1==CorrectClass) {
                Diff=1.0-SummationNeuron[ctr1];
                Error+=Diff*Diff;
            }
            else {
                Error+=SummationNeuron[ctr1]*SummationNeuron[ctr1];
            }
        }
        TotalError+=Error;
        if (::PeekMessage(&message, NULL, 0, 0, PM_REMOVE)) {
            ::TranslateMessage(&message);
            ::DispatchMessage(&message);
        }
    }
    TotalError/=m_nNumCases;
    if(CumulateDerivs) {
        for (ctr=0; ctr<m_nTransientModelSize; ctr++) {

```

```

        Deriv[ctr]/=m_nNumCases;
        Deriv2[ctr]/=m_nNumCases;
    }
}
/* m_nUserDisplayDiscreteError=*DiscreteError;    //for user output on timer 1
   m_dUserDisplayError=TotalError;
*/
delete [] SummationNeuron;
return TotalError;
}
BOOL CPNN::ConjGradientsMin(double ImprovementTolerance)
{
    int ctr, ctr1, ItMax=32767, ConvergenceCtr, PoorCJCtr;
    double CurrentBest, CurrentValue, PreviousBest, Tolerance, Improvement;
    double Dot1, Dot2, DLen, High, Scale, Gamma;
    double LowerPoint, MiddlePoint, UpperPoint,
           LowerError, MiddleError, UpperError;
    double *Sigma, *Base, *Deriv, *Direc, *g, *h, *Deriv2;
    double ConvergenceTolerance=0.000000001;
    int CurrentDiscreteError;
    BOOL UserQuit, ImprovementToleranceReached=FALSE;
    m_dUserDisplayError=m_dBestError;
    m_nUserDisplayDiscreteError=-999;
    m_dUserDisplayImprovement=0.0;
    Sigma=new double[m_nTransientModelSize];
    for(ctr=0; ctr<m_nTransientModelSize; ctr++) {
        Sigma[ctr]=m_Sigma[ctr];
    }
    Base=new double[m_nTransientModelSize];
    Deriv=new double[m_nTransientModelSize];
    Direc=new double[m_nTransientModelSize];
    g=new double[m_nTransientModelSize];
    h=new double[m_nTransientModelSize];
    Deriv2=new double[m_nTransientModelSize];
    CurrentBest=CumulateErrorAndDerivs(Sigma, Deriv, Deriv2, TRUE,
        &CurrentDiscreteError);
    m_dUserDisplayError=CurrentBest;//should already equal m_dBestError
    m_nUserDisplayDiscreteError=CurrentDiscreteError;

    PreviousBest=1.e30;
    for (ctr=0; ctr<m_nTransientModelSize; ctr++) {
        Direc[ctr]=-Deriv[ctr];
    }
    memcpy(g, Direc, m_nTransientModelSize*sizeof(double));
    memcpy(h, Direc, m_nTransientModelSize*sizeof(double));
    ConvergenceCtr=0;
    PoorCJCtr=0;
    for(ctr=0; ctr<ItMax; ctr++) {
        if (PreviousBest <= 1.0)
            Tolerance=ConvergenceTolerance;
        else
            Tolerance=ConvergenceTolerance*PreviousBest;
        if((PreviousBest-CurrentBest)<=Tolerance) {
            if(++ConvergenceCtr>=3)
                // If the function is small
                // Work on absolutes
                // But if it is large
                // Keep things relative
                // If little improvement
                // Then count how many

```

```

    m_dBestError=CurrentBest;
    break; // And quit if too many
}
else // But a good iteration
    ConvergenceCtr=0; // Resets this counter
if (m_bStopNow) {
    m_dBestError=CurrentBest;
    break; //user has quit
}
Dot1=Dot2=DLen=0.0; // For finding directional derivs
High=1.e-4; // For scaling glob_min
for(ctr1=0; ctr1<m_nTransientModelSize; ctr1++) {
    Base[ctr1]=Sigma[ctr1]; // We step out from here
    if(Deriv2[ctr1]>High) // Keep track of second derivatives
        High=Deriv2[ctr1]; // For linear search via glob_min
    Dot1+=Direc[ctr1]*g[ctr1]; // Directional first derivative
    Dot2+=Direc[ctr1]*Direc[ctr1]*Deriv2[ctr1]; // and second
    DLen+=Direc[ctr1]*Direc[ctr1]; // Length of search vector
}
DLen=sqrt(DLen); // Actual length
Scale=Dot1/Dot2; // Newton's ideal but unstable scale
High=1.5/High; // Less ideal but more stable heuristic
if(High<1.e-4) // Subjectively keep it realistic
    High=1.e-4;

if(Scale<0.0) // This is truly pathological
    Scale=High; // So stick with old reliable
else if(Scale<0.1*High) // Bound the Newton scale
    Scale=0.1*High; // To be close to the stable scale
else if(Scale>10.0*High) // Bound it both above and below
    Scale=10.0*High;
m_LowerSearchLimit=0.0;
m_UpperSearchLimit=2.0*Scale;
MiddleError=PreviousBest=CurrentBest;
UserQuit=CGGlobMin(-3, &LowerPoint, &LowerError, &MiddlePoint, &MiddleError,
    &UpperPoint, &UpperError, Sigma, Base, Direc);
if(UserQuit) {
    if(MiddleError<CurrentBest) { // If global caused improvement
        for(ctr1=0; ctr1<m_nTransientModelSize; ctr1++) {
            Sigma[ctr1]=Base[ctr1]+MiddlePoint*Direc[ctr1];
            if(Sigma[ctr1]<1.e-10) // Limit it away from zero
                Sigma[ctr1]=1.e-10; // Fairly arbitrary constant
        }
        CurrentBest=MiddleError;
    }
    else { // Else revert to starting point
        for(ctr1=0; ctr1<m_nTransientModelSize; ctr1++)
            Sigma[ctr1]=Base[ctr1];
    }
    m_dBestError=CurrentBest;
    break; // user has quit
}
if(ConvergenceCtr)
    CurrentBest=BrentMin(20, &LowerPoint, &MiddlePoint, &UpperPoint,

```



```

    MiddleError, Sigma, Base, Direc);
else
    CurrentBest=BrentMin(10, &LowerPoint, &MiddlePoint, &UpperPoint,
    MiddleError, Sigma, Base, Direc);
for(ctr1=0; ctr1<m_nTransientModelSize; ctr1++) {
    Sigma[ctr1]=Base[ctr1]+MiddlePoint*Direc[ctr1];
    if(Sigma[ctr1]<1.e-10) // Limit it away from zero
        Sigma[ctr1]=1.e-10; // Fairly arbitrary constant
}
if(CurrentBest<0.0) { // If user quit during BrentMin
    m_dBestError=-CurrentBest;
    break; //user has quit
}
Improvement=(PreviousBest-CurrentBest)/PreviousBest;
m_dBestError=CurrentBest;
CurrentValue=CumulateErrorAndDerivs(Sigma, Deriv, Deriv2, TRUE,
    &CurrentDiscreteError); //calc derivatives
ASSERT(CurrentValue==CurrentBest);//make sure this occurs...
for(ctr1=0; ctr1<m_nTransientModelSize; ctr1++) // Flip sign to get
    Direc[ctr1]=-Deriv[ctr1]; // negative gradient
m_dUserDisplayError=CurrentBest;
m_nUserDisplayDiscreteError=CurrentDiscreteError;
m_dUserDisplayImprovement=Improvement*100.0;
if ((Improvement*100.0)<=ImprovementTolerance) {
    ImprovementToleranceReached=TRUE;
    break;
}
Gamma=FindGamma(g, Direc);
if(Gamma<0.0)
    Gamma=0.0;
if(Gamma>10.0)
    Gamma=10.0;

if(Improvement<0.001) // Count how many times we
    ++PoorCJCtr; // got poor improvement
else // in a row
    PoorCJCtr=0;

if(PoorCJCtr>=2) { // If several times
    if(Gamma>1.0) // limit gamma
        Gamma=1.0;
}

if (PoorCJCtr>=6) { // If too many times
    PoorCJCtr=0; // set gamma to 0
    Gamma=0.0; // to use steepest descent (gradient)
}
FindNewDir(Gamma, g, h, Direc); // Compute search direction
}
delete [] Sigma;
delete [] Base;
delete [] Deriv;
delete [] Direc;
delete [] g;

```

```

delete [] h;
delete [] Deriv2;
return ImprovementToleranceReached;          //FALSE if user quit or poor performance
}
double CPNN::FindGamma(double *g, double *Grad)
{
double Denominator=0.0, Numerator=0.0;
for (int ctr=0; ctr<m_nTransientModelSize; ctr++) {
    Denominator+=g[ctr]*g[ctr];
    Numerator+=(Grad[ctr]-g[ctr])*Grad[ctr]; // Grad is neg gradient
}

if (Denominator==0.0)                // Should never happen (means gradient is zero!)
    return 0.0;
else
    return Numerator/Denominator;
}
void CPNN::FindNewDir(double Gamma, double *g, double *h, double *Grad)
{
for(int ctr=0; ctr<m_nTransientModelSize; ctr++) {
    g[ctr]=Grad[ctr];
    Grad[ctr]=h[ctr]=g[ctr]+Gamma*h[ctr];
}
}
double CPNN::UniVarError(double Point, double *Sigma,
                        double *Base, double *Direc)
{
int DummyError;//not used here, but used in CumulateErrorAndDerivs()
for(int ctr=0; ctr<m_nTransientModelSize; ctr++) {
    Sigma[ctr]=Base[ctr]+Point*Direc[ctr];
    if(Sigma[ctr]<1.e-10) {
        TRACE("Don, Sigma too small.");
        Sigma[ctr]=1.e-10;
    }
}
return CumulateErrorAndDerivs(Sigma, (double *)NULL, (double *)NULL,
    FALSE, &DummyError);
}
BOOL CPNN::CGGlobMin(int NumSearchPoints, double *LowerPoint, double *LowerError,
                    double *MiddlePoint, double *MiddleError, double *UpperPoint,
                    double *UpperError, double *Sigma, double *Base,
                    double *Direc)
{
int ctr, ibest;
double CurrentPoint, CurrentError, Rate, Previous=0.0;
BOOL Turned_Up=FALSE, KnowFirstPoint=FALSE, UserQuit=FALSE;
if (NumSearchPoints<0) {
    NumSearchPoints=-NumSearchPoints;
    KnowFirstPoint=TRUE;
}
Rate=(m_UpperSearchLimit-m_LowerSearchLimit)/(NumSearchPoints-1);
CurrentPoint=m_LowerSearchLimit;
ibest=-1 ; // For proper escape if error reaches 0
for (ctr=0; ctr<NumSearchPoints; ctr++) {

```

```

if(ctr!=KnowFirstPoint) {
    CurrentError=UniVarError(CurrentPoint, Sigma, Base, Direc);
}
else {
    CurrentError=*MiddleError;
}
if ((ctr==0)||((CurrentError<*MiddleError))) { // Keep track of best here
    ibest=ctr ;
    *MiddlePoint=CurrentPoint;
    *MiddleError=CurrentError;
    *LowerError=Previous; // Function value to its left
    Turned_Up=FALSE; // Flag that min is not yet bounded
}
else if (ctr==(ibest+1)) { // Didn't improve so this point may
    *UpperError=CurrentError; // be the right neighbor of the best
    Turned_Up=TRUE; // Flag that min is bounded
}
Previous=CurrentError; // Keep track for left neighbor of best
if ((CurrentError==0)&&(ibest>0)&&Turned_Up)
    break; // Done if good enough and both neighbors found
if (m_bStopNow) {
    UserQuit=TRUE;
    break;
}
CurrentPoint+=Rate;
}
*LowerPoint=*MiddlePoint-Rate;
*UpperPoint=*MiddlePoint+Rate;
if (!Turned_Up) { // Must extend to the right (larger x)
    for (;;) { // Endless loop goes as long as necessary
        *UpperError=UniVarError(*UpperPoint, Sigma, Base, Direc);
        if (*UpperError>*MiddleError) // If function increased we are done
            break;
        if ((*LowerError==*MiddleError)&&(*MiddleError==*UpperError))
            break; // Give up if flat
        if (m_bStopNow) {
            UserQuit=TRUE;
            break;
        }
        *LowerPoint=*MiddlePoint; // Shift all points
        *LowerError=*MiddleError;
        *MiddlePoint=*UpperPoint;
        *MiddleError=*UpperError;
        Rate*=3.0; // Step further each time
        *UpperPoint+=Rate;
    }
}
else if(ibest==0) { // Must extend to the left (smaller x)
    for (;;) { // Endless loop goes as long as necessary
        *LowerError=UniVarError(*LowerPoint, Sigma, Base, Direc);
        if (*LowerError>*MiddleError) // If function increased we are done
            break;
        if ((*LowerError==*MiddleError)&&(*MiddleError==*UpperError))
            break; // Give up if flat
    }
}

```

```

        if (m_bStopNow) {
            UserQuit=TRUE;
            break;
        }
        *UpperPoint=*MiddlePoint;           // Shift all points
        *UpperError=*MiddleError;
        *MiddlePoint=*LowerPoint;
        *MiddleError=*LowerError;
        Rate*=3.0;                           // Step further each time
        *LowerPoint-=Rate;
    }
}
m_LowerSearchLimit=*LowerPoint;
m_UpperSearchLimit=*UpperPoint;
return UserQuit;
}
double CPNN::BrentMin(int ItMax, double *LowerPoint, double *MiddlePoint,
                    double *UpperPoint, double MiddleError, double *Sigma,
                    double *Base, double *Direc)
{
    int ctr;
    double PreviousDistance=0.0, Step=0.0, Tol=1.e-6, Tol1, Tol2, eps=1.e-7;
    double BestPoint, SecondBestPoint, ThirdBestPoint;
    double BestError, SecondBestError, ThirdBestError;
    double LowPoint, MidPoint, HighPoint, t1, t2;
    double Numerator, Denominator, TestDistance, RecentPoint, RecentError;
    BestPoint=SecondBestPoint=ThirdBestPoint=*MiddlePoint;
    LowPoint=*LowerPoint;
    HighPoint=*UpperPoint;
    BestError=SecondBestError=ThirdBestError=MiddleError;
    for(ctr=0; ctr<ItMax; ctr++) {
        if (m_bStopNow) {
            return -BestError;
        }
        MidPoint=0.5*(LowPoint+HighPoint);
        Tol1=Tol*(fabs(BestPoint)+eps);
        Tol2=2.*Tol1;

        if(fabs(BestPoint-MidPoint)<=(Tol2-0.5*(HighPoint-LowPoint)))
            break;
        if((ctr>=2)&&((ThirdBestError-BestError)<eps))
            break;

        if(fabs(PreviousDistance)>Tol1) {           //If moved far enough try parabolic fit
            t1=(BestPoint-SecondBestPoint)*(BestError-ThirdBestError);
            t2=(BestPoint-ThirdBestPoint)*(BestError-SecondBestError);
            Numerator=(BestPoint-ThirdBestPoint)*
                t2-(BestPoint-SecondBestPoint)*t1;
            Denominator=2.*(t1-t2);               // Estimate will be numer / denom
            TestDistance=PreviousDistance;        // Will verify interval is shrinking
            PreviousDistance=Step;               // Save for next iteration
            if(Denominator!=0.0)                 // Avoid dividing by zero
                Step=Numerator/Denominator;     // the parabolic estimate to min
            else

```

```

Step=1.e30; // Assures failure of next test

if ((fabs(Step)<fabs(0.5*TestDistance)) // If shrinking
    &&(Step+BestPoint>LowPoint) // and within known bounds
    &&(Step+BestPoint<HighPoint)) { // then we can use the
    RecentPoint=BestPoint+Step; // parabolic estimate
    if((RecentPoint-LowPoint<Tol2)|| // If we are very close
        (HighPoint-RecentPoint<Tol2)) { // to known bounds
        if(BestPoint<MidPoint) // then stabilize
            Step=Tol1;
        else
            Step=-Tol1;
    }
}
else { // Parabolic estimate poor, so use golden section
    PreviousDistance=(BestPoint>=MidPoint)?
        LowPoint-BestPoint:HighPoint-BestPoint;
    Step=.3819660*PreviousDistance;
}
}
else {
    PreviousDistance=(BestPoint>=MidPoint)?
        LowPoint-BestPoint:HighPoint-BestPoint;
    Step=.3819660*PreviousDistance;
}
}

if(fabs(Step)>=Tol1)
    RecentPoint=BestPoint+Step;
else {
    if(Step>0.0)
        RecentPoint=BestPoint+Tol1;
    else
        RecentPoint=BestPoint-Tol1;
}
RecentError=UniVarError(RecentPoint, Sigma, Base, Direc);

if(RecentError<=BestError) { // If we improved...
    if(RecentPoint>=BestPoint) // Shrink the (lowpt,highpt) interval by
        LowPoint=BestPoint; // replacing the appropriate endpoint
    else
        HighPoint=BestPoint;
    ThirdBestPoint=SecondBestPoint;// Update x and f values for best,
    SecondBestPoint=BestPoint;// second and third best
    BestPoint=RecentPoint;
    ThirdBestError=SecondBestError;
    SecondBestError=BestError;
    BestError=RecentError;
}
else { // We did not improve
    if(RecentPoint<BestPoint) // Shrink the (xlow,xhigh) interval by
        LowPoint=RecentPoint; // replacing the appropriate endpoint
    else
        HighPoint=RecentPoint;
}

```

```

        if((RecentError<=SecondBestError) // If we beat the second best
           !(SecondBestPoint==BestPoint)) { // or we had a duplication
            ThirdBestPoint=SecondBestPoint; // update the second and third
            SecondBestPoint=RecentPoint; // best, though not the best.
            ThirdBestError=SecondBestError;
            SecondBestError=RecentError;
        }
        else if((RecentError<=ThirdBestError) // Maybe at least we can
                !(ThirdBestPoint==BestPoint) // beat the third best or get
                !(ThirdBestPoint==SecondBestPoint)) { // rid of a duplication
            ThirdBestPoint=RecentPoint;
            ThirdBestError=RecentError;
        }
    }
}

*LowerPoint=LowPoint;
*MiddlePoint=BestPoint;
*UpperPoint=HighPoint;
return BestError;
}
BOOL CPNN::TrainSigmasOpt(double Tolerance)
{
    BOOL ImprovementToleranceReached=FALSE;
    m_bStopNow=FALSE;
    m_strUserMessage.Format("In Conjugate Gradients Algorithm.");
    ImprovementToleranceReached=ConjGradientsMin(Tolerance);
    m_bSigmasOptimized=TRUE;
    return ImprovementToleranceReached;
}
void CPNN::FillTrainingResultsArray(CResultsArray *ResultsArray)
{
    int PredictedClass, CorrectClass, ctr1;
    double *CurrentTransient, *SummationNeuron, TotalError=0.0, Diff, Error, Act;
    SummationNeuron=new double[m_nNumClasses];
    for(int ctr=0; ctr<m_nNumCases; ctr++) {
        CurrentTransient=m_TransientArray->
            GetAt(ctr)->GetTransientModel();
        PredictedClass=Classify(CurrentTransient, SummationNeuron, ctr, NULL, &Act);
        CorrectClass=m_TransientArray->GetAt(ctr)->GetTransientClass();
        Error=0.0;
        for (ctr1=0; ctr1<m_nNumClasses; ctr1++) {
            if (ctr1==CorrectClass) {
                Diff=1.0-SummationNeuron[ctr1];
                Error+=Diff*Diff;
            }
            else {
                Error+=SummationNeuron[ctr1]*SummationNeuron[ctr1];
            }
        }
    }
    CTrainingResults *TrainingResults=new CTrainingResults(1);
    //only 1 Segmentation mode
    TrainingResults->SetRawFilePath(m_TransientArray->
        GetAt(ctr)->GetRawFilePath());
}

```

Appendix B: TAC-MM Source Code

```
    TrainingResults->SetCorrectClass(CorrectClass);
    TrainingResults->SetPredictedClass(PredictedClass,0);
    TrainingResults->SetError(Error,0);
    ResultsArray->InsertAt(ctr, TrainingResults);
}

delete [] SummationNeuron;
return;
}
```

AddUnknownDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// AddUnknownDialog.h : header file
//
////////////////////////////////////////////////////////////////////
// CAddUnknownDialog dialog
class CAddUnknownDialog : public CDialog
{
// Construction
public:
    CAddUnknownDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
   //{{AFX_DATA(CAddUnknownDialog)
    enum { IDD = IDD_ADD_UNKNOWN_DIALOG };
    int     m_TransientClass;
    CString m_Date;
    int     m_Position;
    CString m_Time;
    CString m_TransmitterMake;
    CString m_TransmitterModel;
    CString m_TransmitterSerial;
    //}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAddUnknownDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CAddUnknownDialog)
    // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```


AddUnknownDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// AddUnknownDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "AddUnknownDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CAddUnknownDialog dialog
CAddUnknownDialog::CAddUnknownDialog(CWnd* pParent /*=NULL*/)
: CDialog(CAddUnknownDialog::IDD, pParent)
{
   //{{AFX_DATA_INIT(CAddUnknownDialog)
    m_TransientClass = 0;
    m_Date = _T("");
    m_Position = 0;
    m_Time = _T("");
    m_TransmitterMake = _T("");
    m_TransmitterModel = _T("");
    m_TransmitterSerial = _T("");
    //}}AFX_DATA_INIT
}

void CAddUnknownDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAddUnknownDialog)
    DDX_Text(pDX, IDC_CLASS, m_TransientClass);
    DDV_MinMaxInt(pDX, m_TransientClass, 0, 1000000);
    DDX_Text(pDX, IDC_DATE, m_Date);
    DDX_Text(pDX, IDC_POSITION, m_Position);
    DDV_MinMaxInt(pDX, m_Position, 0, 1000000);
    DDX_Text(pDX, IDC_TIME, m_Time);
    DDX_Text(pDX, IDC_TXMAKE, m_TransmitterMake);
    DDX_Text(pDX, IDC_TXMODEL, m_TransmitterModel);
    DDX_Text(pDX, IDC_TXSERIAL, m_TransmitterSerial);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAddUnknownDialog, CDialog)
    //{{AFX_MSG_MAP(CAddUnknownDialog)
    // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CAddUnknownDialog message handlers
```

BatchProgressDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// BatchProgressDialog.h : header file
//
////////////////////////////////////////////////////////////////////
// CBatchProgressDialog dialog
class CBatchProgressDialog : public CDialog
{
// Construction
public:
    CBatchProgressDialog(CWnd* pParent = NULL); // standard constructor
    int m_ProgressBarLimit;
    BOOL Create();
    void OnUpdateProgress(int CurrentProgress);
    // Dialog Data
   //{{AFX_DATA(CBatchProgressDialog)
    enum { IDD = IDD_BATCHPROGRESS_DIALOG };
    CProgressCtrl m_UpdateProgress;
    //}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CBatchProgressDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
   //{{AFX_MSG(CBatchProgressDialog)
    virtual BOOL OnInitDialog();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

BatchProgressDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// BatchProgressDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "BatchProgressDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// CBatchProgressDialog dialog
CBatchProgressDialog::CBatchProgressDialog(CWnd* pParent /*=NULL*/)
: CDialog(CBatchProgressDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CBatchProgressDialog)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}
void CBatchProgressDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CBatchProgressDialog)
    DDX_Control(pDX, IDC_UPDATEPROGRESS, m_UpdateProgress);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CBatchProgressDialog, CDialog)
    //{{AFX_MSG_MAP(CBatchProgressDialog)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CBatchProgressDialog message handlers
BOOL CBatchProgressDialog::Create()
{
    return CDialog::Create(CBatchProgressDialog::IDD);
}
BOOL CBatchProgressDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_UPDATEPROGRESS);
    pProg->SetRange(0,m_ProgressBarLimit);
    pProg->SetPos(0);
    CString strText;
    strText.Format ("%d",m_ProgressBarLimit);
    SetDlgItemText(IDC_BARLIMIT, strText);
    strText.Format("0");
    SetDlgItemText(IDC_CURRENT, strText);
}

```

```
        return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
    }
void CBatchProgressDialog::OnUpdateProgress(int CurrentProgress)
{
    CString strText;
    strText.Format ("%d", CurrentProgress);
    SetDlgItemText(IDC_CURRENT, strText);

    MSG message;
    if (::PeekMessage(&message, NULL, 0, 0, PM_REMOVE)) {
        ::TranslateMessage(&message);
        ::DispatchMessage(&message);
    }
}
```

ClassifyDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// ClassifyDialog.h : header file
//
///////////////////////////////////////////////////////////////////
// CClassifyDialog dialog
class CClassifyDialog : public CDialog
{
// Construction
public:
    CClassifyDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
    //({AFX_DATA(CClassifyDialog)
    enum { IDD = IDD_CLASSIFY_DIALOG };
    double   m_Confidence;
    int      m_PredictedClass;
    double   m_Activation;
    //})AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //({AFX_VIRTUAL(CClassifyDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //})AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    //({AFX_MSG(CClassifyDialog)
    // NOTE: the ClassWizard will add member functions here
    //})AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

ClassifyDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// ClassifyDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "ClassifyDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CClassifyDialog dialog
CClassifyDialog::CClassifyDialog(CWnd* pParent /*=NULL*/)
: CDialog(CClassifyDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CClassifyDialog)
    m_Confidence = 0.0;
    m_PredictedClass = 0;
    m_Activation = 0.0;
    //}}AFX_DATA_INIT
}
void CClassifyDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CClassifyDialog)
    DDX_Text(pDX, IDC_CONFIDENCE, m_Confidence);
    DDV_MinMaxDouble(pDX, m_Confidence, 0., 100.);
    DDX_Text(pDX, IDC_PREDICTEDCLASS, m_PredictedClass);
    DDV_MinMaxInt(pDX, m_PredictedClass, 0, 10000000);
    DDX_Text(pDX, IDC_ACTIVATION, m_Activation);
    DDV_MinMaxInt(pDX, m_Activation, 0.0, 1.0);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CClassifyDialog, CDialog)
    //{{AFX_MSG_MAP(CClassifyDialog)
    // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CClassifyDialog message handlers
```

EditFPPProgressDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// EditFPPProgressDialog.h : header file
//
////////////////////////////////////////////////////////////////////
// CEditFPPProgressDialog dialog
class CEditFPPProgressDialog : public CDialog
{
// Construction
public:
    CEditFPPProgressDialog(CWnd* pParent = NULL); // standard constructor
    int m_ProgressBarLimit;
    BOOL Create();
    void OnUpdateProgress(int CurrentProgress);
// Dialog Data
    //{AFX_DATA(CEditFPPProgressDialog)
    enum { IDD = IDD_EDITFPPROGRESS_DIALOG };
    CProgressCtrl m_UpdateProgress;
    //}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CEditFPPProgressDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}AFX_VIRTUAL
// Implementation
    protected:
    // Generated message map functions
    //{AFX_MSG(CEditFPPProgressDialog)
    virtual BOOL OnInitDialog();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

EditFPPProgressDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// EditFPPProgressDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "EditFPPProgressDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// CEditFPPProgressDialog dialog
CEditFPPProgressDialog::CEditFPPProgressDialog(CWnd* pParent /*=NULL*/)
: CDialog(CEditFPPProgressDialog::IDD, pParent)
{
//{{AFX_DATA_INT(CEditFPPProgressDialog)
//}}AFX_DATA_INT
}
void CEditFPPProgressDialog::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CEditFPPProgressDialog)
DDX_Control(pDX, IDC_UPDATEPROGRESS, m_UpdateProgress);
//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CEditFPPProgressDialog, CDialog)
//{{AFX_MSG_MAP(CEditFPPProgressDialog)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CEditFPPProgressDialog message handlers
BOOL CEditFPPProgressDialog::Create()
{
return CDialog::Create(CEditFPPProgressDialog::IDD);
}
BOOL CEditFPPProgressDialog::OnInitDialog()
{
CDialog::OnInitDialog();

CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_UPDATEPROGRESS);
pProg->SetRange(0,m_ProgressBarLimit);
pProg->SetPos(0);
CString strText;
strText.Format ("%d",m_ProgressBarLimit);
SetDlgItemText(IDC_BARLIMIT, strText);
strText.Format("0");
SetDlgItemText(IDC_CURRENT, strText);
return TRUE; // return TRUE unless you set the focus to a control
```



```
        // EXCEPTION: OCX Property Pages should return FALSE
    }
void CEditFPProgressDialog::OnUpdateProgress(int CurrentProgress)
{
    CString strText;
    strText.Format ("%d",CurrentProgress);
    SetDlgItemText(IDC_CURRENT, strText);
}
```

EnterClassDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// EnterClassDialog.h : header file
//
////////////////////////////////////////////////////////////////////
// CEnterClassDialog dialog
class CEnterClassDialog : public CDialog
{
// Construction
public:
    CEnterClassDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
    //{{AFX_DATA(CEnterClassDialog)
    enum { IDD = IDD_ENTER_CLASS_DIALOG };
    int     m_TransientClass;
    //}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CEnterClassDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CEnterClassDialog)
    // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

EnterClassDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// EnterClassDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "EnterClassDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CEnterClassDialog dialog
CEnterClassDialog::CEnterClassDialog(CWnd* pParent /*=NULL*/)
: CDialog(CEnterClassDialog::IDD, pParent)
{
   //{{AFX_DATA_INIT(CEnterClassDialog)
    m_TransientClass = 0;
   //}}AFX_DATA_INIT
}
void CEnterClassDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    {{{AFX_DATA_MAP(CEnterClassDialog)
    DDX_Text(pDX, IDC_TRANSIENTCLASS, m_TransientClass);
    DDV_MinMaxInt(pDX, m_TransientClass, 0, 100000);
    }}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CEnterClassDialog, CDialog)
    {{{AFX_MSG_MAP(CEnterClassDialog)
        // NOTE: the ClassWizard will add message map macros here
    }}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CEnterClassDialog message handlers

```

EnterSigmaInitParamsDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// EnterSigmaInitParamsDialog.h : header file
//
// EnterSigmaInitParamsDialog dialog
class CEnterSigmaInitParamsDialog : public CDialog
{
// Construction
public:
    CEnterSigmaInitParamsDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
    {{{AFX_DATA(CEnterSigmaInitParamsDialog)
    enum { IDD = IDD_ENTER_SIGMAINIT_PARAMS };
    double m_LowerSearchLimit;
    int     m_NumSearchPoints;
    double m_UpperSearchLimit;
    }}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    {{{AFX_VIRTUAL(CEnterSigmaInitParamsDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    }}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    {{{AFX_MSG(CEnterSigmaInitParamsDialog)
    // NOTE: the ClassWizard will add member functions here
    }}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

EnterSigmaInitParamsDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// EnterSigmaInitParamsDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "EnterSigmaInitParamsDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CEnterSigmaInitParamsDialog dialog
CEnterSigmaInitParamsDialog::CEnterSigmaInitParamsDialog(CWnd* pParent /*=NULL*/)
    : CDialog(CEnterSigmaInitParamsDialog::IDD, pParent)
{
   //{{AFX_DATA_INIT(CEnterSigmaInitParamsDialog)
    m_LowerSearchLimit = 0.0;
    m_NumSearchPoints = 0;
    m_UpperSearchLimit = 0.0;
    //}}AFX_DATA_INIT
}

void CEnterSigmaInitParamsDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CEnterSigmaInitParamsDialog)
    DDX_Text(pDX, IDC_LOWERLIM, m_LowerSearchLimit);
    DDV_MinMaxDouble(pDX, m_LowerSearchLimit, 1.e-004, 1000.);
    DDX_Text(pDX, IDC_NUMPOINTS, m_NumSearchPoints);
    DDV_MinMaxInt(pDX, m_NumSearchPoints, 3, 10000);
    DDX_Text(pDX, IDC_UPPERLIM, m_UpperSearchLimit);
    DDV_MinMaxDouble(pDX, m_UpperSearchLimit, 1.e-003, 1000.);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CEnterSigmaInitParamsDialog, CDialog)
    //{{AFX_MSG_MAP(CEnterSigmaInitParamsDialog)
    // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CEnterSigmaInitParamsDialog message handlers

```

FractalParamDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// FractalParamDialog.h : header file
//
///////////////////////////////////////////////////////////////////
// CFractalParamDialog dialog
class CFractalParamDialog : public CDialog
{
// Construction
public:
    CFractalParamDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
   //{{AFX_DATA(CFractalParamDialog)
    enum { IDD = IDD_FRACTALPARAM_DIALOG };
    BOOL    m_setSegmentationParams;
    BOOL    m_SetModelParams;
    CEdit   m_ModelWindowShiftEdit;
    CEdit   m_ModelWindowSizeEdit;
    CEdit   m_ModelVariancePairsEdit;
    CEdit   m_ThresholdEdit;
    CEdit   m_WindowSizeEdit;
    CEdit   m_VariancePairsEdit;
    CEdit   m_TransientSizeEdit;
    CEdit   m_RawFileSizeEdit;
    int     m_RawFileSize;
    int     m_TransientSize;
    int     m_WindowSize;
    int     m_VariancePairs;
    double  m_Threshold;
    int     m_ModelVariancePairs;
    int     m_ModelWindowSize;
    int     m_ModelWindowShift;
    //}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CFractalParamDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:

    // Generated message map functions
   //{{AFX_MSG(CFractalParamDialog)
    virtual BOOL OnInitDialog();
    afx_msg void OnModelparams();
    afx_msg void OnSegmentationparams();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

FractalParamDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// FractalParamDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "FractalParamDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CFractalParamDialog dialog
CFractalParamDialog::CFractalParamDialog(CWnd* pParent /*=NULL*/)
    : CDialog(CFractalParamDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CFractalParamDialog)
m_SetSegmentationParams = FALSE;
m_SetModelParams = FALSE;
//}}AFX_DATA_INIT
}
void CFractalParamDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFractalParamDialog)
DDX_Check(pDX, IDC_SEGMENTATIONPARAMS, m_SetSegmentationParams);
DDX_Check(pDX, IDC_MODELPARAMS, m_SetModelParams);
DDX_Control(pDX, IDC_MODELWINDOWSHIFT, m_ModelWindowShiftEdit);
DDX_Control(pDX, IDC_WINDOWSIZE2, m_ModelWindowSizeEdit);
DDX_Control(pDX, IDC_VARIANCEPAIRS2, m_ModelVariancePairsEdit);
DDX_Control(pDX, IDC_THRESHOLD, m_ThresholdEdit);
DDX_Control(pDX, IDC_WINDOWSIZE, m_WindowSizeEdit);
DDX_Control(pDX, IDC_VARIANCEPAIRS, m_VariancePairsEdit);
DDX_Control(pDX, IDC_TRANSIENTSIZE, m_TransientSizeEdit);
DDX_Control(pDX, IDC_RAWFILESIZE, m_RawFileSizeEdit);
DDX_Text(pDX, IDC_RAWFILESIZE, m_RawFileSize);
DDV_MinMaxInt(pDX, m_RawFileSize, 1000, 20000);
DDX_Text(pDX, IDC_TRANSIENTSIZE, m_TransientSize);
DDV_MinMaxInt(pDX, m_TransientSize, 100, 5000);
DDX_Text(pDX, IDC_WINDOWSIZE, m_WindowSize);
DDV_MinMaxInt(pDX, m_WindowSize, 128, 2048);
DDX_Text(pDX, IDC_VARIANCEPAIRS, m_VariancePairs);
DDV_MinMaxInt(pDX, m_VariancePairs, 3, 25);
DDX_Text(pDX, IDC_THRESHOLD, m_Threshold);
DDV_MinMaxDouble(pDX, m_Threshold, 0., 20.);
DDX_Text(pDX, IDC_VARIANCEPAIRS2, m_ModelVariancePairs);
DDV_MinMaxInt(pDX, m_ModelVariancePairs, -1, 9);
DDX_Text(pDX, IDC_WINDOWSIZE2, m_ModelWindowSize);
DDV_MinMaxInt(pDX, m_ModelWindowSize, 128, 1024);
}
}

```

```

DDX_Text(pDX, IDC_MODELWINDOWSHIFT, m_ModelWindowShift);
DDV_MinMaxInt(pDX, m_ModelWindowShift, 1, 512);
//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CFractalParamDialog, CDialog)
    //{{AFX_MSG_MAP(CFractalParamDialog)
ON_BN_CLICKED(IDC_MODELPARAMS, OnModelparams)
ON_BN_CLICKED(IDC_SEGMENTATIONPARAMS, OnSegmentationparams)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CFractalParamDialog message handlers
BOOL CFractalParamDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_VariancePairsEdit.LimitText(2);
    m_ModelWindowShiftEdit.LimitText(3);
    m_ModelVariancePairsEdit.LimitText(2);
    m_ModelWindowSizeEdit.LimitText(4);
    m_WindowSizeEdit.LimitText(4);
    m_ThresholdEdit.LimitText(6); // # of chars user can enter in edit box
    GetDlgItem(IDC_WINDOWSIZE)->EnableWindow(FALSE);
    GetDlgItem(IDC_VARIANCEPAIRS)->EnableWindow(FALSE);
    GetDlgItem(IDC_THRESHOLD)->EnableWindow(FALSE);
    GetDlgItem(IDC_WINDOWSIZE2)->EnableWindow(FALSE);
    GetDlgItem(IDC_VARIANCEPAIRS2)->EnableWindow(FALSE);
    GetDlgItem(IDC_MODELWINDOWSHIFT)->EnableWindow(FALSE);
    m_SetModelParams=FALSE;
    m_SetSegmentationParams=FALSE;

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}
void CFractalParamDialog::OnModelparams()
{
    if(m_SetModelParams==IsDlgButtonChecked(IDC_MODELPARAMS)) {
        GetDlgItem(IDC_WINDOWSIZE2)->EnableWindow(TRUE);
        GetDlgItem(IDC_VARIANCEPAIRS2)->EnableWindow(TRUE);
        GetDlgItem(IDC_MODELWINDOWSHIFT)->EnableWindow(TRUE);
    }
    else {
        GetDlgItem(IDC_WINDOWSIZE2)->EnableWindow(FALSE);
        GetDlgItem(IDC_VARIANCEPAIRS2)->EnableWindow(FALSE);
        GetDlgItem(IDC_MODELWINDOWSHIFT)->EnableWindow(FALSE);
    }
}
void CFractalParamDialog::OnSegmentationparams()
{
    if(m_SetSegmentationParams==IsDlgButtonChecked(IDC_SEGMENTATIONPARAMS)) {
        GetDlgItem(IDC_WINDOWSIZE)->EnableWindow(TRUE);
        GetDlgItem(IDC_VARIANCEPAIRS)->EnableWindow(TRUE);
        GetDlgItem(IDC_THRESHOLD)->EnableWindow(TRUE);
    }
}

```


Appendix B: TAC-MM Source Code

```
else {  
    GetDlgItem(IDC_WINDOWSIZE)->EnableWindow(FALSE);  
    GetDlgItem(IDC_VARIANCEPAIRS)->EnableWindow(FALSE);  
    GetDlgItem(IDC_THRESHHOLD)->EnableWindow(FALSE);  
}  
}
```

ModeParamsDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// ModeParamsDialog.h : header file
//
///////////////////////////////////////////////////////////////////
// CModeParamsDialog dialog
class CModeParamsDialog : public CDialog
{
// Construction
public:
    CModeParamsDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
    {{{ AFX_DATA(CModeParamsDialog)
    enum { IDD = IDD_MODEPARAMETERSDIALOG };
    int     m_Mode;
    int     m_NumModes;
    int     m_MinSeparation;
    double  m_LowThreshold;
    double  m_UpperThreshhold;
    }}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    {{{ AFX_VIRTUAL(CModeParamsDialog)
    protected:
        virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    }}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    {{{ AFX_MSG(CModeParamsDialog)
    afx_msg void OnMultimode();
    afx_msg void OnSinglemode();
    virtual BOOL OnInitDialog();
    }}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

ModeParamsDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// ModeParamsDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "ModeParamsDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//////////////////////////////////////
// CModeParamsDialog dialog
CModeParamsDialog::CModeParamsDialog(CWnd* pParent /*=NULL*/)
: CDialog(CModeParamsDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CModeParamsDialog)
    m_Mode = -1;
    m_NumModes = 0;
    m_MinSeparation = 0;
    m_LowThreshold = 0.0;
    m_UpperThreshold = 0.0;
    //}}AFX_DATA_INIT
}

void CModeParamsDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CModeParamsDialog)
    DDX_Radio(pDX, IDC_SINGLEMODE, m_Mode);
    DDX_Text(pDX, IDC_NUMMODES, m_NumModes);
    DDV_MinMaxInt(pDX, m_NumModes, 1, 1000);
    DDX_Text(pDX, IDC_MINSEPARATION, m_MinSeparation);
    DDV_MinMaxInt(pDX, m_MinSeparation, 1, 100000);
    DDX_Text(pDX, IDC_LOWTHRESHHOLD, m_LowThreshold);
    DDV_MinMaxDouble(pDX, m_LowThreshold, 0., 50.);
    DDX_Text(pDX, IDC_UPPERTHRESHHOLD, m_UpperThreshold);
    DDV_MinMaxDouble(pDX, m_UpperThreshold, 0., 50.);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CModeParamsDialog, CDialog)
    //{{AFX_MSG_MAP(CModeParamsDialog)
    ON_BN_CLICKED(IDC_MULTIMODE, OnMultimode)
    ON_BN_CLICKED(IDC_SINGLEMODE, OnSinglemode)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
//////////////////////////////////////
// CModeParamsDialog message handlers
void CModeParamsDialog::OnMultimode()
{
```

Appendix B: TAC-MM Source Code

```
GetDlgItem(IDC_NUMMODES)->EnableWindow(TRUE);
GetDlgItem(IDC_LOWTHRESHHOLD)->EnableWindow(TRUE);
GetDlgItem(IDC_UPPERTHRESHHOLD)->EnableWindow(TRUE);
GetDlgItem(IDC_MINSEPARATION)->EnableWindow(TRUE);
}
void CModeParamsDialog::OnSinglemode()
{
    GetDlgItem(IDC_NUMMODES)->EnableWindow(FALSE);
    GetDlgItem(IDC_LOWTHRESHHOLD)->EnableWindow(FALSE);
    GetDlgItem(IDC_UPPERTHRESHHOLD)->EnableWindow(FALSE);
    GetDlgItem(IDC_MINSEPARATION)->EnableWindow(FALSE);
}
BOOL CModeParamsDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_Mode=0;
    UpdateData(FALSE);

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}
```

NewDatabase.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// NewDatabase.h : header file
//
///////////////////////////////////////////////////////////////////
// CNewDatabase dialog
class CNewDatabase : public CDialog
{
// Construction
public:
    CNewDatabase(CWnd* pParent = NULL); // standard constructor
// Dialog Data
   //{{AFX_DATA(CNewDatabase)
    enum { IDD = IDD_NEW_DATABASE_DIALOG };
    CEdit    m_ModelWindowShiftEdit;
    CEdit    m_ModelWindowSizeEdit;
    CEdit    m_ModelVariancePairsEdit;
    CEdit    m_ThresholdEdit;
    CEdit    m_WindowSizeEdit;
    CEdit    m_VariancePairsEdit;
    CEdit    m_TransientSizeEdit;
    CEdit    m_RawFileSizeEdit;
    int      m_RawFileSize;
    int      m_TransientSize;
    int      m_WindowSize;
    int      m_VariancePairs;
    double   m_Threshold;
    int      m_ModelVariancePairs;
    int      m_ModelWindowSize;
    int      m_ModelWindowShift;
    //}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNewDatabase)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
   //{{AFX_MSG(CNewDatabase)
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

NewDatabase.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// NewDatabase.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "NewDatabase.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CNewDatabase dialog
CNewDatabase::CNewDatabase(CWnd* pParent /*=NULL*/)
    : CDialog(CNewDatabase::IDD, pParent)
{
   //{{AFX_DATA_INIT(CNewDatabase)
    m_RawFileSize = 0;
    m_TransientSize = 0;
    m_VariancePairs = 0;
    m_WindowSize = 0;
    m_Threshold = 0.0;
    m_ModelVariancePairs = 0;
    m_ModelWindowSize = 0;
    m_ModelWindowShift = 0;
    //}}AFX_DATA_INIT
}

void CNewDatabase::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    {{{AFX_DATA_MAP(CNewDatabase)
    DDX_Control(pDX, IDC_MODELWINDOWSHIFT, m_ModelWindowShiftEdit);
    DDX_Control(pDX, IDC_WINDOWSIZE2, m_ModelWindowSizeEdit);
    DDX_Control(pDX, IDC_VARIANCEPAIRS2, m_ModelVariancePairsEdit);
    DDX_Control(pDX, IDC_THRESHOLD, m_ThresholdEdit);
    DDX_Control(pDX, IDC_WINDOWSIZE, m_WindowSizeEdit);
    DDX_Control(pDX, IDC_VARIANCEPAIRS, m_VariancePairsEdit);
    DDX_Control(pDX, IDC_TRANSIENTSIZE, m_TransientSizeEdit);
    DDX_Control(pDX, IDC_RAWFILESIZE, m_RawFileSizeEdit);
    DDX_Text(pDX, IDC_RAWFILESIZE, m_RawFileSize);
    DDV_MinMaxInt(pDX, m_RawFileSize, 1000, 25000);
    DDX_Text(pDX, IDC_TRANSIENTSIZE, m_TransientSize);
    DDV_MinMaxInt(pDX, m_TransientSize, 100, 5000);
    DDX_Text(pDX, IDC_WINDOWSIZE, m_WindowSize);
    DDV_MinMaxInt(pDX, m_WindowSize, 128, 2048);
    DDX_Text(pDX, IDC_VARIANCEPAIRS, m_VariancePairs);
    DDV_MinMaxInt(pDX, m_VariancePairs, 3, 25);
    DDX_Text(pDX, IDC_THRESHOLD, m_Threshold);
    DDV_MinMaxDouble(pDX, m_Threshold, 1., 20.);
    }}}AFX_DATA_MAP
}

```

```

DDX_Text(pDX, IDC_VARIANCEPAIRS2, m_ModelVariancePairs);
DDV_MinMaxInt(pDX, m_ModelVariancePairs, -1, 9);
DDX_Text(pDX, IDC_WINDOWSIZE2, m_ModelWindowSize);
DDV_MinMaxInt(pDX, m_ModelWindowSize, 128, 1024);
DDX_Text(pDX, IDC_MODELWINDOWSHIFT, m_ModelWindowShift);
DDV_MinMaxInt(pDX, m_ModelWindowShift, 1, 512);
//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CNewDatabase, CDialog)
//{{AFX_MSG_MAP(CNewDatabase)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CNewDatabase message handlers
BOOL CNewDatabase::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_RawFileSizeEdit.LimitText(5);
    m_TransientSizeEdit.LimitText(4);
    m_VariancePairsEdit.LimitText(2);
    m_ModelWindowShiftEdit.LimitText(3);
    m_ModelVariancePairsEdit.LimitText(2);
    m_ModelWindowSizeEdit.LimitText(4);
    m_WindowSizeEdit.LimitText(4);
    m_ThresholdEdit.LimitText(6);    ///<# of chars user can enter in edit box

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

```

SearchDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// SearchDialog.h : header file
//
////////////////////////////////////////////////////////////////////
// CSearchDialog dialog
class CSearchDialog : public CDialog
{
// Construction
public:
    CSearchDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
    //{AFX_DATA(CSearchDialog)
    enum { IDD = IDD_SEARCH_DIALOG };
    int     m_FieldToSearch;
    int     m_Class;
    CString m_Date;
    CString m_Make;
    CString m_Model;
    CString m_Serial;
    CString m_Time;
    //}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CSearchDialog)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}AFX_VIRTUAL
// Implementation
    protected:
    void DisableAllEditBoxes();
    // Generated message map functions
    //{AFX_MSG(CSearchDialog)
    virtual BOOL OnInitDialog();
    afx_msg void OnClassRadio();
    afx_msg void OnRadio1();
    afx_msg void OnRadio2();
    afx_msg void OnRadio3();
    afx_msg void OnRadio4();
    afx_msg void OnRadio5();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```


SearchDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// SearchDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "SearchDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//////////////////////////////////////
// CSearchDialog dialog
CSearchDialog::CSearchDialog(CWnd* pParent /*=NULL*/)
: CDialog(CSearchDialog::IDD, pParent)
{
   //{{AFX_DATA_INIT(CSearchDialog)
    m_FieldToSearch = -1;
    m_Class = 0;
    m_Date = _T("");
    m_Make = _T("");
    m_Model = _T("");
    m_Serial = _T("");
    m_Time = _T("");
    //}}AFX_DATA_INIT
}

void CSearchDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CSearchDialog)
    DDX_Radio(pDX, IDC_CLASS_RADIO, m_FieldToSearch);
    DDX_Text(pDX, IDC_CLASS_EDIT, m_Class);
    DDV_MinMaxInt(pDX, m_Class, 0, 20000);
    DDX_Text(pDX, IDC_DATE_EDIT, m_Date);
    DDV_MaxChars(pDX, m_Date, 8);
    DDX_Text(pDX, IDC_MAKE_EDIT, m_Make);
    DDX_Text(pDX, IDC_MODEL_EDIT, m_Model);
    DDX_Text(pDX, IDC_SERIAL_EDIT, m_Serial);
    DDX_Text(pDX, IDC_TIME_EDIT, m_Time);
    DDV_MaxChars(pDX, m_Time, 8);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSearchDialog, CDialog)
    {{{AFX_MSG_MAP(CSearchDialog)
    ON_BN_CLICKED(IDC_CLASS_RADIO, OnClassRadio)
    ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
    ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
    ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
    ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
    }}}AFX_MSG_MAP
END_MESSAGE_MAP()

```

```

    ON_BN_CLICKED(IDC_RADIO5, OnRadio5)
    //}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CSearchDialog message handlers
BOOL CSearchDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // m_FieldToSearch=0;
    // UpdateData(FALSE);
    DisableAllEditBoxes();
    GetDlgItem(IDC_CLASS_EDIT)->EnableWindow(TRUE);
    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}
void CSearchDialog::OnClassRadio()
{
    DisableAllEditBoxes();
    GetDlgItem(IDC_CLASS_EDIT)->EnableWindow(TRUE);
}
void CSearchDialog::OnRadio1()
{
    DisableAllEditBoxes();
    GetDlgItem(IDC_MAKE_EDIT)->EnableWindow(TRUE);
}
void CSearchDialog::OnRadio2()
{
    DisableAllEditBoxes();
    GetDlgItem(IDC_MODEL_EDIT)->EnableWindow(TRUE);
}
void CSearchDialog::OnRadio3()
{
    DisableAllEditBoxes();
    GetDlgItem(IDC_SERIAL_EDIT)->EnableWindow(TRUE);
}
void CSearchDialog::OnRadio4()
{
    DisableAllEditBoxes();
    GetDlgItem(IDC_DATE_EDIT)->EnableWindow(TRUE);
}
void CSearchDialog::OnRadio5()
{
    DisableAllEditBoxes();
    GetDlgItem(IDC_TIME_EDIT)->EnableWindow(TRUE);
}
void CSearchDialog::DisableAllEditBoxes()
{
    CString strText;
    strText.Format ("%d",0);
    SetDlgItemText(IDC_CLASS_EDIT, strText);
    strText.Format("");
    SetDlgItemText(IDC_MAKE_EDIT, strText);
    SetDlgItemText(IDC_MODEL_EDIT, strText);
}

```

Appendix B: TAC-MM Source Code

```
SetDlgItemText(IDC_SERIAL_EDIT, strText);
SetDlgItemText(IDC_DATE_EDIT, strText);
SetDlgItemText(IDC_TIME_EDIT, strText);
GetDlgItem(IDC_CLASS_EDIT)->EnableWindow(FALSE);
GetDlgItem(IDC_MODEL_EDIT)->EnableWindow(FALSE);
GetDlgItem(IDC_MAKE_EDIT)->EnableWindow(FALSE);
GetDlgItem(IDC_SERIAL_EDIT)->EnableWindow(FALSE);
GetDlgItem(IDC_DATE_EDIT)->EnableWindow(FALSE);
GetDlgItem(IDC_TIME_EDIT)->EnableWindow(FALSE);
}
```

SetRejectionDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// SetRejectionDialog.h : header file
//
////////////////////////////////////////////////////////////////////
//if
#ifdef(AFX_SETREJECTIONDIALOG_H_C624B441_043F_11D1_BF55_444553540000__INCLUD
ED_)
#define
AFX_SETREJECTIONDIALOG_H_C624B441_043F_11D1_BF55_444553540000__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// CSetRejectionDialog dialog
class CSetRejectionDialog : public CDialog
{
// Construction
public:
    CSetRejectionDialog(CWnd* pParent = NULL); // standard constructor
// Dialog Data
   //{{AFX_DATA(CSetRejectionDialog)
    enum { IDD = IDD_SETREJECTION_DIALOG };
    double m_dRejectionThreshold;
    //}}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CSetRejectionDialog)
protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
   //{{AFX_MSG(CSetRejectionDialog)
    // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.
#endif
#ifdef(AFX_SETREJECTIONDIALOG_H_C624B441_043F_11D1_BF55_444553540000__INCLUD
ED_)

```

SetRejectionDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// SetRejectionDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "SetRejectionDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CSetRejectionDialog dialog
CSetRejectionDialog::CSetRejectionDialog(CWnd* pParent /*=NULL*/)
: CDialog(CSetRejectionDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CSetRejectionDialog)
    m_dRejectionThreshold = 0.0;
    //}}AFX_DATA_INIT
}
void CSetRejectionDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSetRejectionDialog)
    DDX_Text(pDX, IDC_REJECTIONTHRESHOLD, m_dRejectionThreshold);
    DDV_MinMaxDouble(pDX, m_dRejectionThreshold, 0., 1.);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSetRejectionDialog, CDialog)
    //{{AFX_MSG_MAP(CSetRejectionDialog)
    // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////
// CSetRejectionDialog message handlers
```

SigmaInitDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// SigmaInitDialog.h : header file
//
///////////////////////////////////////////////////////////////////
// CSigmaInitDialog dialog
class CSigmaInitDialog : public CDialog
{
// Construction
public:
    CSigmaInitDialog(CWnd* pParent = NULL); // standard constructor
    CTransientArray *m_TransientArray;
    CPNN *m_PNN;
    int m_nNumSearchPoints;
    BOOL m_bGlobalSearchDone;
// Dialog Data
    //{{AFX_DATA(CSigmaInitDialog)
    enum { IDD = IDD_TRAINSIGMASINT_DIALOG };
        // NOTE: the ClassWizard will add data members here
    //}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSigmaInitDialog)
    protected:
        virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}AFX_VIRTUAL
// Implementation
    protected:
        double m_dLowerSigma, m_dLowerError, m_dMiddleSigma, m_dMiddleError,
            m_dUpperSigma, m_dUpperError;
        // Generated message map functions
    //{{AFX_MSG(CSigmaInitDialog)
    afx_msg void OnStopbutton();
    afx_msg void OnTrainbutton();
    virtual BOOL OnInitDialog();
    afx_msg void OnTimer(UINT nIDEvent);
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

SigmaInitDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// SigmaInitDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "Trandata.h"
#include "PNN.h"
#include "SigmaInitDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//////////////////////////////////////
// CSigmaInitDialog dialog
CSigmaInitDialog::CSigmaInitDialog(CWnd* pParent /*=NULL*/)
: CDialog(CSigmaInitDialog::IDD, pParent)
{
   //{{AFX_DATA_INIT(CSigmaInitDialog)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}
void CSigmaInitDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CSigmaInitDialog)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSigmaInitDialog, CDialog)
   //{{AFX_MSG_MAP(CSigmaInitDialog)
    ON_BN_CLICKED(IDC_STOPBUTTON, OnStopbutton)
    ON_BN_CLICKED(IDC_TRAINBUTTON, OnTrainbutton)
    ON_WM_TIMER()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
//////////////////////////////////////
// CSigmaInitDialog message handlers
void CSigmaInitDialog::OnStopbutton()
{
    CString strText;
    m_PNN->m_bStopNow=TRUE;
    strText.Format("Press Train to resume or OK to Exit.");
    SetDlgItemText(IDC_STATIC_CURRENT, strText);
    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(FALSE);
    GetDlgItem(IDC_TRAINBUTTON)->EnableWindow(TRUE);
    GetDlgItem(IDOK)->EnableWindow(TRUE);
}
}
```

```

void CSigmaInitDialog::OnTrainbutton()
{
    CString strText;
    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(TRUE);
    GetDlgItem(IDC_TRAINBUTTON)->EnableWindow(FALSE);
    GetDlgItem(IDOK)->EnableWindow(FALSE);
    int Timer=SetTimer(1,200,NULL); // ID #1, 1/5 second, NULL
    ASSERT(Timer!=0);
    if (m_bGlobalSearchDone) {
        m_PNN->TrainSigmasInit(m_nNumSearchPoints, TRUE, &m_dLowerSigma,
            &m_dLowerError,&m_dMiddleSigma, &m_dMiddleError, &m_dUpperSigma,
            &m_dUpperError);
    }
    else {
        m_bGlobalSearchDone=m_PNN->TrainSigmasInit(m_nNumSearchPoints, FALSE,
            &m_dLowerSigma, &m_dLowerError, &m_dMiddleSigma, &m_dMiddleError,
            &m_dUpperSigma, &m_dUpperError);
    }
    KillTimer(1);

    strText.Format("%.12lf",m_PNN->m_nUserDisplaySigma);
    SetDlgItemText(IDC_STATIC_SIGMA, strText);
    strText.Format("%.12lf",m_PNN->m_dUserDisplayError);
    SetDlgItemText(IDC_STATIC_ERROR, strText);
    CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_INITPROGRESS);
    pProg->SetPos(m_PNN->m_nUserDisplayDiscreteError);
    strText.Format("%d",m_PNN->m_nUserDisplayDiscreteError);
    SetDlgItemText(IDC_STATIC_MISCLASS, strText);

    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(FALSE);
    GetDlgItem(IDC_TRAINBUTTON)->EnableWindow(TRUE);
    GetDlgItem(IDOK)->EnableWindow(TRUE);
}
BOOL CSigmaInitDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_INITPROGRESS);
    pProg->SetRange(0,m_TransientArray->GetSize());
    pProg->SetPos(0);
    CString strText;
    strText.Format("%d",m_TransientArray->GetSize());
    SetDlgItemText(IDC_STATIC_MAXERROR, strText);
    strText.Format("Press Train to begin.");
    SetDlgItemText(IDC_STATIC_CURRENT, strText);
    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(FALSE);
    m_bGlobalSearchDone=FALSE;

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}
void CSigmaInitDialog::OnTimer(UINT nIDEvent)
{
    if (nIDEvent==1) {

```



```
CString strText;
if (!m_PNN->m_bStopNow)
    SetDlgItemText(IDC_STATIC_CURRENT, m_PNN->m_strUserMessage);
strText.Format("%.12lf", m_PNN->m_nUserDisplaySigma);
SetDlgItemText(IDC_STATIC_SIGMA, strText);
strText.Format("%.12lf", m_PNN->m_dUserDisplayError);
SetDlgItemText(IDC_STATIC_ERROR, strText);
CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_INITPROGRESS);
pProg->SetPos(m_PNN->m_nUserDisplayDiscreteError);
strText.Format("%d", m_PNN->m_nUserDisplayDiscreteError);
SetDlgItemText(IDC_STATIC_MISCLASS, strText);
}

CDialog::OnTimer(nIDEvent);
}
```

TrainSigmasOptDialog.h

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```

// TrainSigmasOptDialog.h : header file
//
///////////////////////////////////////////////////////////////////
// CTrainSigmasOptDialog dialog
class CTrainSigmasOptDialog : public CDialog
{
// Construction
public:
    CTrainSigmasOptDialog(CWnd* pParent = NULL); // standard constructor
    CTransientArray *m_TransientArray;
    CPNN *m_PNN;
    double m_dImprovementTolerance;
// Dialog Data
    //{ AFX_DATA(CTrainSigmasOptDialog)
    enum { IDD = IDD_TRAINSIGMASOFT_DIALOG };
        // NOTE: the ClassWizard will add data members here
    //}AFX_DATA
// Overrides
    // ClassWizard generated virtual function overrides
    //{ AFX_VIRTUAL(CTrainSigmasOptDialog)
    protected:
        virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}AFX_VIRTUAL
// Implementation
protected:
    // Generated message map functions
    //{ AFX_MSG(CTrainSigmasOptDialog)
    afx_msg void OnStopbutton();
    afx_msg void OnTrainbutton();
    virtual BOOL OnInitDialog();
    afx_msg void OnTimer(UINT nIDEvent);
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

TrainSigmasOptDialog.cpp

This file was initially generated using the Microsoft Visual C++ 5.0 ClassWizard. It has been modified as required for the specific application.

```
// TrainSigmasOptDialog.cpp : implementation file
//
#include "stdafx.h"
#include "TAC_MM.h"
#include "Trandata.h"
#include "PNN.h"
#include "TrainSigmasOptDialog.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// CTrainSigmasOptDialog dialog
CTrainSigmasOptDialog::CTrainSigmasOptDialog(CWnd* pParent /*=NULL*/)
: CDialog(CTrainSigmasOptDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CTrainSigmasOptDialog)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}
void CTrainSigmasOptDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CTrainSigmasOptDialog)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CTrainSigmasOptDialog, CDialog)
    //{{AFX_MSG_MAP(CTrainSigmasOptDialog)
    ON_BN_CLICKED(IDC_STOPBUTTON, OnStopbutton)
    ON_BN_CLICKED(IDC_TRAINBUTTON, OnTrainbutton)
    ON_WM_TIMER()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CTrainSigmasOptDialog message handlers
void CTrainSigmasOptDialog::OnStopbutton()
{
    CString strText;
    m_PNN->m_bStopNow=TRUE;
    strText.Format("Press Train to resume or OK to Exit.");
    SetDlgItemText(IDC_STATIC_CURRENT, strText);
    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(FALSE);
    GetDlgItem(IDC_TRAINBUTTON)->EnableWindow(TRUE);
    GetDlgItem(IDOK)->EnableWindow(TRUE);
}
void CTrainSigmasOptDialog::OnTrainbutton()
```

```

{
    BOOL ToleranceReached=FALSE;
    CString strText;
    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(TRUE);
    GetDlgItem(IDC_TRAINBUTTON)->EnableWindow(FALSE);
    GetDlgItem(IDOK)->EnableWindow(FALSE);
    int Timer=SetTimer(1,1000,NULL); // ID #1, 1 second, NULL
    ASSERT(Timer!=0);
    ToleranceReached=m_PNN->TrainSigmasOpt(m_dImprovementTolerance);
    KillTimer(1);

    strText.Format("%.12lf",m_PNN->m_dUserDisplayError);
    SetDlgItemText(IDC_STATIC_ERROR, strText);
    CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_INITPROGRESS);
    pProg->SetPos(m_PNN->m_nUserDisplayDiscreteError);
    strText.Format("%d",m_PNN->m_nUserDisplayDiscreteError);
    SetDlgItemText(IDC_STATIC_MISCLASS, strText);

    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(FALSE);
    if(!ToleranceReached)
        GetDlgItem(IDC_TRAINBUTTON)->EnableWindow(TRUE);
    else {
        strText.Format("Tolerance of %.10f reached. Hit OK to Exit."
            ,m_dImprovementTolerance);
        SetDlgItemText(IDC_STATIC_CURRENT, strText);
    }
    GetDlgItem(IDOK)->EnableWindow(TRUE);
}
BOOL CTrainSigmasOptDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_INITPROGRESS);
    pProg->SetRange(0,m_TransientArray->GetSize());
    pProg->SetPos(0);
    CString strText;
    strText.Format("%d",m_TransientArray->GetSize());
    SetDlgItemText(IDC_STATIC_MAXERROR, strText);
    strText.Format("Press Train to begin.");
    SetDlgItemText(IDC_STATIC_CURRENT, strText);
    GetDlgItem(IDC_STOPBUTTON)->EnableWindow(FALSE);

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}
void CTrainSigmasOptDialog::OnTimer(UINT nIDEvent)
{
    if (nIDEvent==1) {
        CString strText;
        if (!m_PNN->m_bStopNow)
            SetDlgItemText(IDC_STATIC_CURRENT, m_PNN->m_strUserMessage);
        strText.Format("%.12lf",m_PNN->m_dUserDisplayImprovement);
        SetDlgItemText(IDC_STATIC_IMPROVEMENT, strText);
        strText.Format("%.12lf",m_PNN->m_dUserDisplayError);
    }
}

```

Appendix B: TAC-MM Source Code

```
SetDlgItemText(IDC_STATIC_ERROR, strText);
CProgressCtrl* pProg=(CProgressCtrl*)GetDlgItem(IDC_INITPROGRESS);
pProg->SetPos(m_PNN->m_nUserDisplayDiscreteError);
strText.Format("%d",m_PNN->m_nUserDisplayDiscreteError);
SetDlgItemText(IDC_STATIC_MISCLASS, strText);
}

CDialog::OnTimer(nIDEvent);
}
```

TrainingResults.h

```

#ifndef _TRAINRESULTS
#define _TRAINRESULTS
class CTrainingResults:public CObject
{
    //DECLARE_SERIAL(CTrainingResults)
protected:
    CString m_strRawFilePath;
    int m_nCorrectClass;
    int *m_nPredictedClass;
    double *m_dError;
    double *m_dWinningActivation;
public:
    CTrainingResults() { }
    CTrainingResults(int NumClassifyModes)
        { m_nPredictedClass=new int[NumClassifyModes];
          m_dError=new double[NumClassifyModes];
          m_dWinningActivation=new double[NumClassifyModes];
        }
    ~CTrainingResults() { delete [] m_nPredictedClass;
                        delete [] m_dError;
                        delete [] m_dWinningActivation;}
    void SetRawFilePath(CString FilePath) { m_strRawFilePath=FilePath; }
    void SetCorrectClass(int CorrectClass) { m_nCorrectClass=CorrectClass; }
    void SetPredictedClass(int PredictedClass, int ModeIndex)
        { m_nPredictedClass[ModeIndex]=PredictedClass; }
    void SetError(double Error, int ModeIndex)
        { m_dError[ModeIndex]=Error; }
    void SetActivation(double Activation, int ModeIndex)
        { m_dWinningActivation[ModeIndex]=Activation; }
    CString GetRawFilePath() { return m_strRawFilePath; }
    int GetCorrectClass() { return m_nCorrectClass; }
    int GetPredictedClass(int ModeIndex) { return m_nPredictedClass[ModeIndex]; }
    double GetError(int ModeIndex) { return m_dError[ModeIndex]; }
    double GetActivation(int ModeIndex)
        { return m_dWinningActivation[ModeIndex]; }
#ifdef _DEBUG
    void Dump(CDumpContext& dc) const;
#endif
};
typedef CTypedPtrArray <CObArray, CTrainingResults*> CResultsArray;
#endif

```

TrainingResults.cpp

```
#include "StdAfx.h"
#include "TrainingResults.h"
//IMPLEMENT_SERIAL(CTrainingResults, CObject,0)
#ifdef _DEBUG
void CTrainingResults::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);
    dc << "\nm_strRawFilePath = " << m_strRawFilePath;
}
#endif
```

stdafx.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required for the specific application.

```
// stdafx.h : include file for standard system include files,  
// or project specific include files that are used frequently, but  
// are changed infrequently  
//  
#define VC_EXTRALEAN// Exclude rarely-used stuff from Windows headers  
#include <afxtempl.h>  
#include <afxwin.h> // MFC core and standard components  
#include <afxext.h> // MFC extensions  
#ifndef _AFX_NO_AFXCMN_SUPPORT  
#include <afxcmn.h> // MFC support for Windows Common Controls  
#include <afxdlgs.h>  
#endif // _AFX_NO_AFXCMN_SUPPORT
```


resource.h

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required using the visual tools in the Microsoft Developer Studio.

```

//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by TAC_MM.rc
//
#define IDD_ABOUTBOX                100
#define IDR_MAINFRAME               128
#define IDR_TRANSITYPE              129
#define IDD_NEW_DATABASE_DIALOG     130
#define IDD_MODIFY_SETTINGS_DIALOG1 131
#define IDD_ENTER_CLASS_DIALOG      132
#define IDD_CLASSIFY_DIALOG         133
#define IDD_ADD_UNKNOWN_DIALOG      134
#define IDR_ACCELERATOR1            135
#define IDD_TRAINSIGMASINIT_DIALOG  136
#define IDD_ENTER_SIGMAINIT_PARAMS  137
#define IDD_FRACTALPARAM_DIALOG     138
#define IDD_EDITFPPROGRESS_DIALOG   140
#define IDD_SEARCH_DIALOG           141
#define IDD_TRAINSIGMASOFT_DIALOG   142
#define IDD_BATCHPROGRESS_DIALOG    143
#define IDD_MODEPARAMETERSDIALOG    144
#define IDD_SETREJECTION_DIALOG     146
#define IDC_RAWFILESIZE             1001
#define IDC_TRANSIENTSIZE           1002
#define IDC_WINDOWSIZE              1003
#define IDC_WINDOWSHIFT             1004
#define IDC_WINDOWSIZE2             1004
#define IDC_VARIANCEPAIRS           1005
#define IDC_VARIANCEPAIRS2          1006
#define IDC_THRESHOLD                1011
#define IDC_TRANSIENTCLASS           1012
#define IDC_MODELWINDOWSHIFT        1012
#define IDC_PREDICTEDCLASS           1013
#define IDC_CONFIDENCE               1014
#define IDC_TXMAKE                   1016
#define IDC_TXMODEL                  1017
#define IDC_TXSERIAL                 1018
#define IDC_CLASS                    1019
#define IDC_DATE                     1020
#define IDC_TIME                     1021
#define IDC_POSITION                 1022
#define IDC_INITPROGRESS             1023
#define IDC_STOPBUTTON               1025
#define IDC_TRAINBUTTON              1026
#define IDC_STATIC_SIGMA             1027
#define IDC_STATIC_MAXERROR          1028
#define IDC_STATIC_MISCLASS          1029
#define IDC_LOWERLIM                 1030

```

Appendix B: TAC-MM Source Code

```

#define IDC_STATIC_ERROR 1030
#define IDC_UPPERLIM 1031
#define IDC_NUMPOINTS 1032
#define IDC_STATIC_CURRENT 1033
#define IDC_SEGMENTATIONPARAMS 1037
#define IDC_MODELPARAMS 1038
#define IDC_UPDATEPROGRESS 1039
#define IDC_BARLIMIT 1040
#define IDC_CURRENT 1041
#define IDC_MAKE_EDIT 1047
#define IDC_MODEL_EDIT 1048
#define IDC_SERIAL_EDIT 1049
#define IDC_DATE_EDIT 1050
#define IDC_TIME_EDIT 1051
#define IDC_CLASS_EDIT 1052
#define IDC_CLASS_RADIO 1053
#define IDC_RADIO1 1054
#define IDC_STATIC_IMPROVEMENT 1054
#define IDC_RADIO2 1055
#define IDC_RADIO3 1056
#define IDC_SINGLEMODE 1056
#define IDC_RADIO4 1057
#define IDC_MULTIMODE 1057
#define IDC_RADIO5 1058
#define IDC_NUMMODES 1058
#define IDC_LOWTHRESHHOLD 1059
#define IDC_UPPERTHRESHHOLD 1060
#define IDC_MINSEPARATION 1061
#define IDC_REJECTIONTHRESHHOLD 1062
#define IDC_ACTIVATION 1064
#define ID_DATABASE_ADDTRANSIENT 32771
#define ID_DATABASE_NEXT 32773
#define ID_DATABASE_PREV 32774
#define ID_VIEW_SEARCH 32775
#define ID_DATABASE_DELETETRANSIENT 32776
#define ID_DATABASE_VARIANCESETTINGS 32784
#define ID_VIEW_PREV 32785
#define ID_VIEW_NEXT 32786
#define ID_VIEW_RAWSIGNAL 32794
#define ID_VIEW_FRACTALTRAJECTORY 32795
#define ID_SEGMENTATION_VIEW 32797
#define ID_VIEW_SEGMENTATION 32798
#define ID_VIEW_TRANSIENT 32799
#define ID_NEURALNET_CLASSIFY 32800
#define ID_NEURALNET_INITIALIZESIGMAS 32801
#define ID_NEURALNET_OPTIMIZESIGMAS 32802
#define ID_NEURALNET_NETWORKPARAMETERS 32803
#define ID_ACCEL32804 32804
#define ID_EDIT_FRACTALPARAMETERS 32805
#define ID_VIEW_ZOOMEDRAWSIGNAL 32810
#define ID_VIEW_TRAININGRESULTS 32811
#define ID_NEURALNET_BATCHCLASSIFY 32812
#define ID_NEURALNET_MODEPARAMETERS 32813
#define ID_NEURALNET_SETREJECTIONTHRESHHOLD 32814

```

Appendix B: TAC-MM Source Code

```
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS 1
#define _APS_NEXT_RESOURCE_VALUE 147
#define _APS_NEXT_COMMAND_VALUE 32815
#define _APS_NEXT_CONTROL_VALUE 1065
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif
```

TAC_MM.rc

This file was initially generated using the Microsoft Visual C++ 5.0 AppWizard. It has been modified as required using the visual tools in the Microsoft Developer Studio.

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"
// Generated Help ID header file
#define APSTUDIO_HIDDEN_SYMBOLS
#include "resource.hm"
#undef APSTUDIO_HIDDEN_SYMBOLS
#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif // _WIN32
#ifdef APSTUDIO_INVOKED
////////////////////////////////////
// TEXTINCLUDE
//
1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif\r\n"
    "#include ""res\\TAC_MM.rc2"" // non-Microsoft Visual C++ edited resources\r\n"
    "#include ""afxres.rc"" // Standard components\r\n"
    "#include ""afxprint.rc"" // printing/print preview resources\r\n"

```

```

    "#endif0"
END
#endif // APSTUDIO_INVOKED
////////////////////////////////////
//
// Icon
//
// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME      ICON DISCARDABLE "res\\TAC_MM.ico"
IDR_TRANSITYPE     ICON DISCARDABLE "res\\TAC_MMDoc.ico"
////////////////////////////////////
//
// Bitmap
//
IDR_MAINFRAME      BITMAP MOVEABLE PURE "res\\Toolbar.bmp"
////////////////////////////////////
//
// Toolbar
//
IDR_MAINFRAME TOOLBAR DISCARDABLE 16, 15
BEGIN
    BUTTON ID_FILE_NEW
    BUTTON ID_FILE_OPEN
    BUTTON ID_FILE_SAVE
    SEPARATOR
    BUTTON ID_EDIT_COPY
    SEPARATOR
    BUTTON ID_VIEW_NEXT
    BUTTON ID_VIEW_PREV
    BUTTON ID_VIEW_SEARCH
    SEPARATOR
    BUTTON ID_DATABASE_ADDTRANSIENT
    BUTTON ID_DATABASE_DELETETRANSIENT
    SEPARATOR
    BUTTON ID_NEURALNET_CLASSIFY
    SEPARATOR
    BUTTON ID_FILE_PRINT
    BUTTON ID_APP_ABOUT
END
////////////////////////////////////
//
// Menu
//
IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\\tCtrl+N", ID_FILE_NEW
        MENUITEM "&Open...\\tCtrl+O", ID_FILE_OPEN
        MENUITEM "&Save\\tCtrl+S", ID_FILE_SAVE
        MENUITEM "Save &As...", ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print...\\tCtrl+P", ID_FILE_PRINT
    END
END

```

```

MENUITEM "Print Pre&view",          ID_FILE_PRINT_PREVIEW
MENUITEM "Pa&ge Setup...",         ID_FILE_PAGE_SETUP
MENUITEM SEPARATOR
MENUITEM "Recent File",           ID_FILE_MRU_FILE1, GRAYED
MENUITEM SEPARATOR
MENUITEM "E&xit",                  ID_APP_EXIT
END
POPUP "&Edit"
BEGIN
  MENUITEM "&Copy Image\tCtrl+C",    ID_EDIT_COPY
  MENUITEM SEPARATOR
  MENUITEM "&Add Transient",        ID_DATABASE_ADDTRANSIENT
  MENUITEM "&Delete Transient",     ID_DATABASE_DELETETRANSIENT
  MENUITEM SEPARATOR
  MENUITEM "&Fractal Parameters",   ID_EDIT_FRACTALPARAMETERS
END
POPUP "&View"
BEGIN
  MENUITEM "&Toolbar",              ID_VIEW_TOOLBAR
  MENUITEM "&Status Bar",          ID_VIEW_STATUS_BAR
  MENUITEM "S&plit",                ID_WINDOW_SPLIT
  MENUITEM SEPARATOR
  MENUITEM "S&egmentation View",    ID_VIEW_SEGMENTATION
  , CHECKED
  MENUITEM "&Transient View",      ID_VIEW_TRANSIENT
  MENUITEM "&Raw Signal",          ID_VIEW_RAWSIGNAL, CHECKED
  MENUITEM "&Fractal Trajectory",  ID_VIEW_FRACTALTRAJECTORY
  , CHECKED
  MENUITEM SEPARATOR
  MENUITEM "&Zoomed Raw Signal",    ID_VIEW_ZOOMEDRAWSIGNAL
  , CHECKED
  MENUITEM "&Classification Stats", ID_VIEW_TRAININGRESULTS
  MENUITEM SEPARATOR
  MENUITEM "&Next",                 ID_VIEW_NEXT
  MENUITEM "&Previous",            ID_VIEW_PREV
  MENUITEM "Se&arch",              ID_VIEW_SEARCH
END
POPUP "&Neural Net"
BEGIN
  MENUITEM "&Classify",             ID_NEURALNET_CLASSIFY
  MENUITEM "&Batch Classify",      ID_NEURALNET_BATCHCLASSIFY
  MENUITEM "&Mode Parameters",     ID_NEURALNET_MODEPARAMETERS
  MENUITEM "&Set Rejection Threshold ", ID_NEURALNET_SETREJECTIONTHRESHOLD
  MENUITEM SEPARATOR
  MENUITEM "&Initialize Sigmas",   ID_NEURALNET_INITIALIZESIGMAS
  MENUITEM "&Optimize Sigmas",    ID_NEURALNET_OPTIMIZESIGMAS
END
POPUP "&Help"
BEGIN
  MENUITEM "&About TAC_MM...",      ID_APP_ABOUT
END
END
////////////////////////////////////
//

```

```

// Accelerator
//
IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
BEGIN
    "N",      ID_FILE_NEW,      VIRTKEY, CONTROL
    "O",      ID_FILE_OPEN,    VIRTKEY, CONTROL
    "S",      ID_FILE_SAVE,    VIRTKEY, CONTROL
    "P",      ID_FILE_PRINT,   VIRTKEY, CONTROL
    "Z",      ID_EDIT_UNDO,    VIRTKEY, CONTROL
    "X",      ID_EDIT_CUT,     VIRTKEY, CONTROL
    "C",      ID_EDIT_COPY,    VIRTKEY, CONTROL
    "V",      ID_EDIT_PASTE,   VIRTKEY, CONTROL
    VK_BACK,  ID_EDIT_UNDO,    VIRTKEY, ALT
    VK_DELETE, ID_EDIT_CUT,    VIRTKEY, SHIFT
    VK_INSERT, ID_EDIT_COPY,   VIRTKEY, CONTROL
    VK_INSERT, ID_EDIT_PASTE,  VIRTKEY, SHIFT
    VK_F6,    ID_NEXT_PANE,    VIRTKEY
    VK_F6,    ID_PREV_PANE,    VIRTKEY, SHIFT
END
IDR_ACCELERATOR1 ACCELERATORS DISCARDABLE
BEGIN
    "C",      ID_ACCEL32804,    VIRTKEY, CONTROL, NOINVERT
END
////////////////////////////////////
//
// Dialog
//
IDD_ABOUTBOX DIALOG DISCARDABLE 0, 0, 217, 55
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About TAC-MM"
FONT 8, "MS Sans Serif"
BEGIN
    ICON      IDR_MAINFRAME, IDC_STATIC, 11, 17, 20, 20
    LTEXT     "TAC-MM Version 1.0", IDC_STATIC, 40, 10, 119, 8, SS_NOPREFIX
    LTEXT     "Copyright © 1997", IDC_STATIC, 40, 25, 119, 8
    DEFPUSHBUTTON "OK", IDOK, 178, 7, 32, 14, WS_GROUP
END
IDD_NEW_DATABASE_DIALOG DIALOGEX 0, 0, 243, 258
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Create New Database"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON "OK", IDOK, 186, 10, 50, 15
    GROUPBOX     "Enter the following fixed parameters:", IDC_STATIC, 16, 5,
        160, 70
    LTEXT        "&Raw File Size", IDC_STATIC, 20, 30, 44, 8
    EDITTEXT     IDC_RAWFILESIZE, 85, 25, 40, 14, ES_AUTOHSCROLL
    LTEXT        "&Transient Size", IDC_STATIC, 20, 50, 46, 8
    EDITTEXT     IDC_TRANSIENTSIZE, 85, 45, 40, 14, ES_AUTOHSCROLL
    GROUPBOX     "Segmentation Parameters", IDC_STATIC, 16, 82, 213, 78
    LTEXT        "&Window Size (128-2048)", IDC_STATIC, 19, 105, 79, 8
    EDITTEXT     IDC_WINDOWSIZE, 135, 99, 40, 14, ES_AUTOHSCROLL
    LTEXT        "&Variance Pairs (3-25)", IDC_STATIC, 19, 121, 71, 8
    EDITTEXT     IDC_VARIANCEPAIRS, 135, 116, 15, 14, ES_AUTOHSCROLL

```

```

LTEXT      "Threshold for transient start (1-20)", IDC_STATIC, 19,
           137, 109, 8, 0, 0, HIDC_STATIC
EDITTEXT   IDC_THRESHOLD, 135, 133, 30, 14, ES_AUTOHSCROLL
PUSHBUTTON "Cancel", IDCANCEL, 186, 32, 50, 14
GROUPBOX   "Feature Extraction Parameters", IDC_STATIC, 16, 167, 213, 78
LTEXT      "&Window Size (128-1024)", IDC_STATIC, 19, 189, 79, 8
EDITTEXT   IDC_WINDOWSIZE2, 135, 182, 40, 14, ES_AUTOHSCROLL
LTEXT      "&Variance Pairs (3-9)", IDC_STATIC, 19, 205, 62, 8
EDITTEXT   IDC_VARIANCEPAIRS2, 135, 199, 15, 14, ES_AUTOHSCROLL
LTEXT      "Window Shift", IDC_STATIC, 19, 221, 100, 8, 0, 0, HIDC_STATIC
EDITTEXT   IDC_MODELWINDOWSHIFT, 135, 217, 30, 14, ES_AUTOHSCROLL
END
IDD_ENTER_CLASS_DIALOG DIALOG DISCARDABLE 0, 0, 186, 68
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Transient Found!"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT   IDC_TRANSIENTCLASS, 72, 31, 40, 14, ES_AUTOHSCROLL
    DEFPUSHBUTTON "OK", IDOK, 129, 47, 50, 14
    LTEXT      "Enter class integer for this transmitter:", IDC_STATIC,
           33, 18, 118, 8
END
IDD_CLASSIFY_DIALOG DIALOG DISCARDABLE 0, 0, 217, 130
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Classification Results"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON "OK", IDCANCEL, 7, 109, 79, 14
    LTEXT      "Predicted Class:", IDC_STATIC, 30, 25, 52, 8
    EDITTEXT   IDC_PREDICTEDCLASS, 99, 21, 31, 12, ES_AUTOHSCROLL |
           ES_READONLY
    LTEXT      "Confidence (%):", IDC_STATIC, 30, 54, 49, 8
    EDITTEXT   IDC_CONFIDENCE, 99, 50, 98, 12, ES_AUTOHSCROLL | ES_READONLY
    DEFPUSHBUTTON "&ADD TO DATABASE", IDOK, 131, 109, 79, 14
    LTEXT      "Neuron Activation:", IDC_STATIC, 30, 83, 60, 8
    EDITTEXT   IDC_ACTIVATION, 99, 77, 98, 14, ES_AUTOHSCROLL | ES_READONLY
END
IDD_ADD_UNKNOWN_DIALOG DIALOG DISCARDABLE 0, 0, 186, 183
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Add Transient to Database"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT      "&Class:", IDC_STATIC, 59, 23, 20, 8
    EDITTEXT   IDC_CLASS, 85, 21, 40, 12, ES_AUTOHSCROLL
    LTEXT      "&Transmitter Make:", IDC_STATIC, 7, 50, 58, 8
    EDITTEXT   IDC_TXMAKE, 85, 47, 79, 14, ES_AUTOHSCROLL
    LTEXT      "Transmitter &Model:", IDC_STATIC, 7, 68, 59, 8
    EDITTEXT   IDC_TXMODEL, 85, 64, 79, 14, ES_AUTOHSCROLL
    LTEXT      "Transmitter &Serial #:", IDC_STATIC, 7, 86, 64, 8
    EDITTEXT   IDC_TXSERIAL, 85, 82, 79, 14, ES_AUTOHSCROLL
    LTEXT      "Transmit &Date:", IDC_STATIC, 7, 104, 47, 8
    EDITTEXT   IDC_DATE, 85, 100, 49, 14, ES_AUTOHSCROLL
    LTEXT      "Transmit T&ime:", IDC_STATIC, 7, 122, 47, 8
    EDITTEXT   IDC_TIME, 85, 117, 40, 14, ES_AUTOHSCROLL

```



```

LTEXT      "&Position in Database:",IDC_STATIC,7,140,68,8
EDITTEXT   IDC_POSITION,85,137,40,14,ES_AUTOHSCROLL
DEFPUSHBUTTON "OK",IDOK,7,162,50,14
PUSHBUTTON  "Cancel",IDCANCEL,129,162,50,14
END
IDD_TRAINSIGMASINIT_DIALOG DIALOG DISCARDABLE 0, 0, 332, 138
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
CAPTION "Initialize Sigmas"
FONT 8, "MS Sans Serif"
BEGIN
PUSHBUTTON  "&Train",IDC_TRAINBUTTON,128,117,75,14
PUSHBUTTON  "&Stop",IDC_STOPBUTTON,7,117,75,14
PUSHBUTTON  "OK",IDOK,249,117,75,14
CONTROL     "Progress!",IDC_INITPROGRESS,"msctls_progress32",
            WS_BORDER,33,78,255,14
LTEXT       "Sigma:",IDC_STATIC,7,26,22,8
LTEXT       "Number of Misclassifications:",IDC_STATIC,7,64,92,8
LTEXT       "",IDC_STATIC_SIGMA,57,26,96,8
CTEXT      "0",IDC_STATIC,25,95,17,8
CTEXT      "",IDC_STATIC_MAXERROR,276,95,23,8
LTEXT      "",IDC_STATIC_MISCLASS,105,64,25,8
CTEXT      "",IDC_STATIC_CURRENT,89,7,153,8
LTEXT      "Error Criterion:",IDC_STATIC,7,44,47,8
LTEXT      "",IDC_STATIC_ERROR,57,44,95,8
END
IDD_ENTER_SIGMAINIT_PARAMS DIALOG DISCARDABLE 0, 0, 206, 118
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Enter Parameters for Sigma Initialization"
FONT 8, "MS Sans Serif"
BEGIN
DEFPUSHBUTTON "OK",IDOK,7,97,70,14
PUSHBUTTON  "Cancel",IDCANCEL,129,97,70,14
LTEXT       "&Lower Search Limit:",IDC_STATIC,7,20,63,8
LTEXT       "&Upper Search Limit:",IDC_STATIC,7,40,63,8
LTEXT       "&Number of Search Points:",IDC_STATIC,7,60,82,8
EDITTEXT    IDC_LOWERLIM,96,17,40,14,ES_AUTOHSCROLL
EDITTEXT    IDC_UPPERLIM,96,37,40,14,ES_AUTOHSCROLL
EDITTEXT    IDC_NUMPOINTS,96,57,40,14,ES_AUTOHSCROLL
END
IDD_FRACTALPARAM_DIALOG DIALOGEX 0, 0, 243, 290
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Change Fractal Parameters"
FONT 8, "MS Sans Serif"
BEGIN
DEFPUSHBUTTON "Start",IDOK,186,10,50,15
GROUPBOX    "Fixed Parameters:",IDC_STATIC,15,5,160,70
LTEXT       "&Raw File Size",IDC_STATIC,20,30,44,8
EDITTEXT    IDC_RAWFILESIZE,85,25,40,14,ES_AUTOHSCROLL | ES_READONLY
LTEXT       "&Transient Size",IDC_STATIC,20,50,46,8
EDITTEXT    IDC_TRANSIENTSIZE,85,45,40,14,ES_AUTOHSCROLL |
            ES_READONLY
GROUPBOX    "Segmentation Parameters",IDC_STATIC,15,97,213,78
LTEXT       "&Window Size (128-2048)",IDC_STATIC,19,121,79,8
EDITTEXT    IDC_WINDOWSIZE,135,114,40,14,ES_AUTOHSCROLL

```

```

LTEXT      "&Variance Pairs (3-25)",IDC_STATIC,19,137,71,8
EDITTEXT   IDC_VARIANCEPAIRS,135,132,15,14,ES_AUTOHSCROLL
LTEXT      "Threshold for transient start (1-20)",IDC_STATIC,19,
          153,109,8,0,0,HIDC_STATIC
EDITTEXT   IDC_THRESHHOLD,135,148,30,14,ES_AUTOHSCROLL
PUSHBUTTON "Cancel",IDCANCEL,186,32,50,14
GROUPBOX   "Feature Extraction Parameters",IDC_STATIC,15,199,213,78
LTEXT      "&Window Size (128-1024)",IDC_STATIC,19,221,79,8
EDITTEXT   IDC_WINDOWSIZE2,135,214,40,14,ES_AUTOHSCROLL
LTEXT      "&Variance Pairs (3-9)",IDC_STATIC,19,237,62,8
EDITTEXT   IDC_VARIANCEPAIRS2,135,231,15,14,ES_AUTOHSCROLL
LTEXT      "Window Shift",IDC_STATIC,19,253,100,8,0,0,HIDC_STATIC
EDITTEXT   IDC_MODELWINDOWSHIFT,135,249,30,14,ES_AUTOHSCROLL
CONTROL    "Change Segmentation Parameters.",IDC_SEGMENTATIONPARAMS,
          "Button",BS_AUTOCHECKBOX | WS_TABSTOP,15,82,125,10
CONTROL    "Change Feature Extraction Parameters.",IDC_MODELPARAMS,
          "Button",BS_AUTOCHECKBOX | WS_TABSTOP,15,183,139,10
END
IDD_EDITFPFPROGRESS_DIALOG DIALOG DISCARDABLE 0, 0, 371, 111
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Updating Database With New Fractal Parameters"
FONT 8, "MS Sans Serif"
BEGIN
CONTROL    "Progress1",IDC_UPDATEPROGRESS,"msctls_progress32",
          WS_BORDER,21,58,327,14
CTEXT     "0",IDC_STATIC,13,75,18,8
CTEXT     "",IDC_BARLIMIT,336,75,22,8
LTEXT     "Currently Updating Transient #:",IDC_STATIC,21,36,99,8
LTEXT     "",IDC_CURRENT,125,36,31,8
END
IDD_SEARCH_DIALOG DIALOG DISCARDABLE 0, 0, 186, 190
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Search for Transient in Database"
FONT 8, "MS Sans Serif"
BEGIN
DEFPUSHBUTTON "Search",IDOK,34,169,50,14
PUSHBUTTON   "Cancel",IDCANCEL,101,169,50,14
CONTROL      "Transmitter Class: ",IDC_CLASS_RADIO,"Button",
          BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP,14,33,73,10
EDITTEXT     IDC_CLASS_EDIT,91,29,40,14,ES_AUTOHSCROLL
CONTROL      "Transmitter Make: ",IDC_RADIO1,"Button",
          BS_AUTORADIOBUTTON | WS_TABSTOP,14,52,75,10
EDITTEXT     IDC_MAKE_EDIT,91,48,73,14,ES_AUTOHSCROLL
CONTROL      "Transmitter Model: ",IDC_RADIO2,"Button",
          BS_AUTORADIOBUTTON | WS_TABSTOP,14,71,76,10
EDITTEXT     IDC_MODEL_EDIT,91,67,73,14,ES_AUTOHSCROLL
CONTROL      "Transmitter Serial: ",IDC_RADIO3,"Button",
          BS_AUTORADIOBUTTON | WS_TABSTOP,14,90,74,10
EDITTEXT     IDC_SERIAL_EDIT,91,86,73,14,ES_AUTOHSCROLL
CONTROL      "Transmit Date: ",IDC_RADIO4,"Button",BS_AUTORADIOBUTTON |
          WS_TABSTOP,14,109,64,10
EDITTEXT     IDC_DATE_EDIT,91,105,40,14,ES_AUTOHSCROLL
CONTROL      "Transmit Time:",IDC_RADIO5,"Button",BS_AUTORADIOBUTTON |
          WS_TABSTOP,14,128,62,10

```

```

EDITTEXT IDC_TIME_EDIT,91,124,40,14,ES_AUTOHSCROLL
GROUPBOX "Search For:",IDC_STATIC,7,15,172,137
END
IDD_TRAINSIGMASOPT_DIALOG DIALOG DISCARDABLE 0, 0, 332, 138
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
CAPTION "Optimize Sigmas"
FONT 8, "MS Sans Serif"
BEGIN
PUSHBUTTON "&Train",IDC_TRAINBUTTON,128,117,75,14
PUSHBUTTON "&Stop",IDC_STOPBUTTON,7,117,75,14
PUSHBUTTON "OK",IDOK,249,117,75,14
CONTROL "Progress!",IDC_INITPROGRESS,"msctls_progress32",
WS_BORDER,33,78,255,14
LTEXT "Error Criterion:",IDC_STATIC,7,26,47,8
LTEXT "Number of Misclassifications:",IDC_STATIC,7,64,92,8
LTEXT "",IDC_STATIC_ERROR,67,26,86,8
CTEXT "0",IDC_STATIC,25,95,17,8
CTEXT "",IDC_STATIC_MAXERROR,276,95,23,8
LTEXT "",IDC_STATIC_MISCLASS,105,64,25,8
CTEXT "",IDC_STATIC_CURRENT,89,7,153,8
LTEXT "Improvement this Iteration (%):",IDC_STATIC,7,45,95,8
LTEXT "",IDC_STATIC_IMPROVEMENT,106,45,79,8
END
IDD_BATCHPROGRESS_DIALOG DIALOG DISCARDABLE 0, 0, 371, 111
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Performing Batch Classification Test on Known Transients."
FONT 8, "MS Sans Serif"
BEGIN
CONTROL "Progress!",IDC_UPDATEPROGRESS,"msctls_progress32",
WS_BORDER,21,58,327,14
CTEXT "0",IDC_STATIC,13,75,18,8
CTEXT "",IDC_BARLIMIT,336,75,22,8
LTEXT "Currently Classifying Transient #:",IDC_STATIC,21,36,
105,8
LTEXT "",IDC_CURRENT,135,36,104,8
END
IDD_MODEPARAMETERSDIALOG DIALOG DISCARDABLE 0, 0, 169, 190
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Select Classification Mode"
FONT 8, "MS Sans Serif"
BEGIN
DEFPUSHBUTTON "OK",IDOK,7,169,50,14
PUSHBUTTON "Cancel",IDCANCEL,112,169,50,14
CONTROL "Single mode segmentation.",IDC_SINGLEMODE,"Button",
BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP,32,20,101,10
CONTROL "Multi-mode segmentation.",IDC_MULTIMODE,"Button",
BS_AUTORADIOBUTTON | WS_TABSTOP,32,49,96,10
GROUPBOX "",IDC_STATIC,7,43,155,113
GROUPBOX "",IDC_STATIC,7,7,155,32
EDITTEXT IDC_NUMMODES,97,70,40,14,ES_AUTOHSCROLL | WS_DISABLED
LTEXT "Number of modes:",IDC_STATIC,32,72,58,12
EDITTEXT IDC_LOWTHRESHHOLD,97,91,40,14,ES_AUTOHSCROLL |
WS_DISABLED
EDITTEXT IDC_UPPERTHRESHHOLD,97,112,40,14,ES_AUTOHSCROLL |

```

```

        WS_DISABLED
LTEXT      "Lower threshold:", IDC_STATIC, 32, 93, 58, 12
LTEXT      "Upper threshold:", IDC_STATIC, 32, 114, 58, 12
LTEXT      "Minimum Separation:", IDC_STATIC, 32, 135, 66, 12
EDITTEXT   IDC_MINSEPARATION, 97, 133, 40, 14, ES_AUTOHSCROLL |
        WS_DISABLED
END
IDD_SETREJECTION_DIALOG DIALOG DISCARDABLE 0, 0, 173, 82
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Set Rejection Threshold"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON "OK", IDOK, 116, 61, 50, 14
    LTEXT      "Enter a rejection threshold.", IDC_STATIC, 7, 17, 85, 8
    EDITTEXT   IDC_REJECTIONTHRESHOLD, 37, 33, 99, 14, ES_AUTOHSCROLL
END
#ifdef _MAC
////////////////////////////////////
//
// Version
//
VS_VERSION_INFO VERSIONINFO
FILEVERSION 1,0,0,1
PRODUCTVERSION 1,0,0,1
FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
FILEFLAGS 0x1L
#else
FILEFLAGS 0x0L
#endif
FILEOS 0x4L
FILETYPE 0x1L
FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "TRANSIENT MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "TRANSIENT\0"
            VALUE "LegalCopyright", "Copyright © 1996\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "TRANSIENT.EXE\0"
            VALUE "ProductName", "TRANSIENT Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END
END

```

```

#endif // !_MAC
//
// DESIGNINFO
//
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 210
        TOPMARGIN, 7
        BOTTOMMARGIN, 48
    END
    IDD_NEW_DATABASE_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 236
        TOPMARGIN, 7
        BOTTOMMARGIN, 251
    END
    IDD_ENTER_CLASS_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 61
    END
    IDD_CLASSIFY_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 210
        TOPMARGIN, 7
        BOTTOMMARGIN, 123
    END
    IDD_ADD_UNKNOWN_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 176
    END
    IDD_TRAINSIGMASINTT_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 325
        TOPMARGIN, 7
        BOTTOMMARGIN, 131
    END
    IDD_ENTER_SIGMAINTT_PARAMS, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 199

```

```

    TOPMARGIN, 7
    BOTTOMMARGIN, 111
END
IDD_FRACTALPARAM_DIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 236
    TOPMARGIN, 7
    BOTTOMMARGIN, 283
END
IDD_EDITFPPROGRESS_DIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 364
    TOPMARGIN, 7
    BOTTOMMARGIN, 104
END
IDD_SEARCH_DIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 179
    TOPMARGIN, 7
    BOTTOMMARGIN, 183
END
IDD_TRAINSIGMASOFT_DIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 325
    TOPMARGIN, 7
    BOTTOMMARGIN, 131
END
IDD_BATCHPROGRESS_DIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 364
    TOPMARGIN, 7
    BOTTOMMARGIN, 104
END
IDD_MODEPARAMETERSDIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 162
    TOPMARGIN, 7
    BOTTOMMARGIN, 183
END
IDD_SETREJECTION_DIALOG, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 166
    TOPMARGIN, 7
    BOTTOMMARGIN, 75
END
END
#endif // APSTUDIO_INVOKED

```

```

////////////////////////////////////
//
// String Table
//
STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    IDR_MAINFRAME          "TAC_MM\\n\\nTAC_MM\\ntrt\\n.trt\\nTACMM.Document\\nTAC_MM
Document"
END
STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    AFX_IDS_APP_TITLE      "TAC_MM"
    AFX_IDS_IDLEMESSAGE    "Ready"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_INDICATOR_EXT       "EXT"
    ID_INDICATOR_CAPS      "CAP"
    ID_INDICATOR_NUM       "NUM"
    ID_INDICATOR_SCRL      "SCRL"
    ID_INDICATOR_OVR       "OVR"
    ID_INDICATOR_REC       "REC"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_FILE_NEW            "Create a new document\\nNew"
    ID_FILE_OPEN           "Open an existing document\\nOpen"
    ID_FILE_CLOSE          "Close the active document\\nClose"
    ID_FILE_SAVE           "Save the active document\\nSave"
    ID_FILE_SAVE_AS        "Save the active document with a new name\\nSave As"
    ID_FILE_PAGE_SETUP     "Change the printing options\\nPage Setup"
    ID_FILE_PRINT_SETUP    "Change the printer and printing options\\nPrint Setup"
    ID_FILE_PRINT          "Print the active document\\nPrint"
    ID_FILE_PRINT_PREVIEW  "Display full pages\\nPrint Preview"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_APP_ABOUT           "Display program information, version number and copyright\\nAbout"
    ID_APP_EXIT            "Quit the application; prompts to save documents\\nExit"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_FILE_MRU_FILE1      "Open this document"
    ID_FILE_MRU_FILE2      "Open this document"
    ID_FILE_MRU_FILE3      "Open this document"
    ID_FILE_MRU_FILE4      "Open this document"
    ID_FILE_MRU_FILE5      "Open this document"
    ID_FILE_MRU_FILE6      "Open this document"
    ID_FILE_MRU_FILE7      "Open this document"
    ID_FILE_MRU_FILE8      "Open this document"
    ID_FILE_MRU_FILE9      "Open this document"
    ID_FILE_MRU_FILE10     "Open this document"
    ID_FILE_MRU_FILE11     "Open this document"
    ID_FILE_MRU_FILE12     "Open this document"

```

```

ID_FILE_MRU_FILE13  "Open this document"
ID_FILE_MRU_FILE14  "Open this document"
ID_FILE_MRU_FILE15  "Open this document"
ID_FILE_MRU_FILE16  "Open this document"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_NEXT_PANE      "Switch to the next window pane\nNext Pane"
    ID_PREV_PANE      "Switch back to the previous window pane\nPrevious Pane"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_WINDOW_SPLIT   "Split the active window into panes\nSplit"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_EDIT_CLEAR      "Erase the selection\nErase"
    ID_EDIT_CLEAR_ALL  "Erase everything\nErase All"
    ID_EDIT_COPY        "Copy the transient image and put it on the Clipboard\nCopy"
    ID_EDIT_CUT         "Cut the selection and put it on the Clipboard\nCut"
    ID_EDIT_FIND        "Find the specified text\nFind"
    ID_EDIT_PASTE       "Insert Clipboard contents\nPaste"
    ID_EDIT_REPEAT      "Repeat the last action\nRepeat"
    ID_EDIT_REPLACE     "Replace specific text with different text\nReplace"
    ID_EDIT_SELECT_ALL  "Select the entire document\nSelect All"
    ID_EDIT_UNDO        "Undo the last action\nUndo"
    ID_EDIT_REDO        "Redo the previously undone action\nRedo"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_VIEW_TOOLBAR    "Show or hide the toolbar\nToggle ToolBar"
    ID_VIEW_STATUS_BAR "Show or hide the status bar\nToggle StatusBar"
END
STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCSIZE     "Change the window size"
    AFX_IDS_SCMOVE     "Change the window position"
    AFX_IDS_SCMINIMIZE "Reduce the window to an icon"
    AFX_IDS_SCMAXIMIZE "Enlarge the window to full size"
    AFX_IDS_SCNEXTWINDOW "Switch to the next document window"
    AFX_IDS_SCPREVWINDOW "Switch to the previous document window"
    AFX_IDS_SCCLOSE    "Close the active window and prompts to save the documents"
END
STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCRESTORE  "Restore the window to normal size"
    AFX_IDS_SCTASKLIST "Activate Task List"
END
STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_PREVIEW_CLOSE "Close print preview mode\nCancel Preview"
END
STRINGTABLE DISCARDABLE
BEGIN

```



```

ID_DATABASE_ADDTRANSIENT "Add transient to database.\nAdd"
ID_DATABASE_NEXT      "Go to the next transient in database.\nNext"
ID_DATABASE_PREV      "Go to the previous transient in database.\nPrevious"
ID_VIEW_SEARCH        "Search for transient in database.\nSearch"
ID_DATABASE_DELETETRANSIENT "Delete transient from database.\nDelete"
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_DATABASE_VARIANCESETTINGS
        "Adjust variance settings for transient segmentation. "
    ID_VIEW_PREV      "Go to the previous transient in database.\nPrevious"
    ID_VIEW_NEXT      "Go to the next transient in database.\nNext"
    ID_VIEW_RAWSIGNAL  "View the raw signal on the screen."
    ID_VIEW_FRACTALTRAJECTORY "View variance dimension trajectory on screen."
    ID_SEGMENTATION_VIEW "Show "
    ID_VIEW_SEGMENTATION "View entire raw signal with segmentation information."
    ID_VIEW_TRANSIENT  "View transient with amplified fractal dimension (dyadic)."
END
STRINGTABLE DISCARDABLE
BEGIN
    ID_NEURALNET_CLASSIFY "Read an unknown transient from disk and classify it.\nClassify"
    ID_NEURALNET_INITIALIZESIGMAS "Perform univariate sigma initialization. "
    ID_NEURALNET_OPTIMIZESIGMAS
        "Optimize sigmas using conjugate gradients algorithm."
    ID_NEURALNET_NETWORKPARAMETERS "Set neural network parameters."
    ID_EDIT_FRACTALPARAMETERS
        "Modify parameters for fractal segmentation and feature extraction."
    ID_VIEW_ZOOMEDRAWNSIGNAL "View zoomed raw signal in lower view."
    ID_VIEW_TRAININGRESULTS "View results from recent training or batch classification."
    ID_NEURALNET_BATCHCLASSIFY
        "Test classification on several new, but known, transients."
    ID_NEURALNET_MODEPARAMETERS "Set multi-mode classification parameters."
END
#endif // English (U.S.) resources
////////////////////////////////////
#ifndef APSTUDIO_INVOKED
////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif
#endif
#include "res\TAC_MM.rc2" // non-Microsoft Visual C++ edited resources
#include "afxres.rc" // Standard components
#include "afxprint.rc" // printing/print preview resources
#endif
////////////////////////////////////
#endif // not APSTUDIO_INVOKED

```

APPENDIX C

DATA AND SOFTWARE FOR FRACTAL DIMENSION VERIFICATION

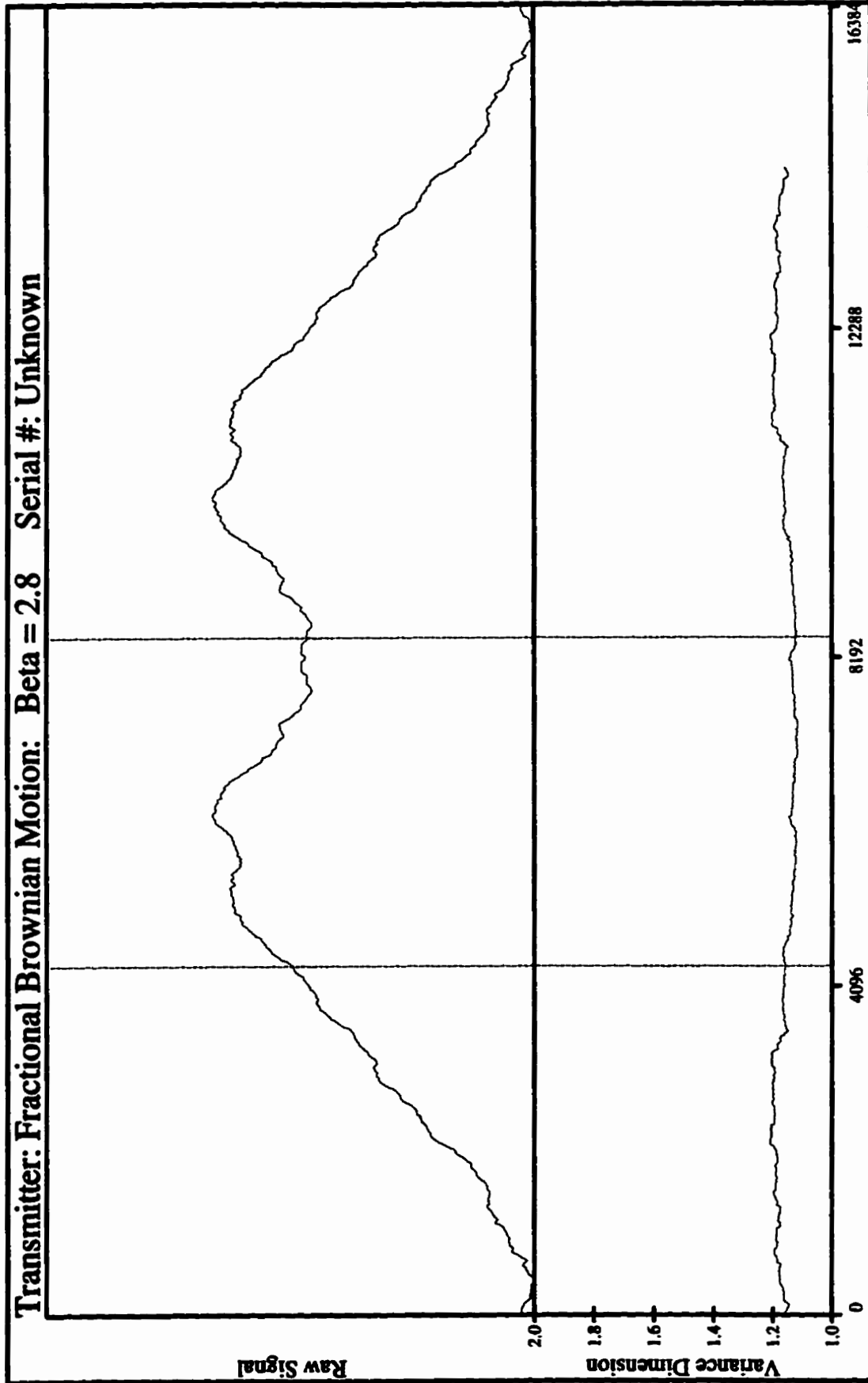
This appendix contains images of the Fractional Brownian Motion (fBm) signals used for verification of the fractal dimension calculations in TAC-MM. The images are printed using TAC-MM with the following parameter settings:

Raw File Size	16384
Transient Size	4096
Segmentation Window Size	2048
Segmentation Variance Pairs	25
Segmentation Threshold	1
Feature Extraction Window Size	512
Feature Extraction Variance Pairs	5
Feature Extraction Window Shift	16

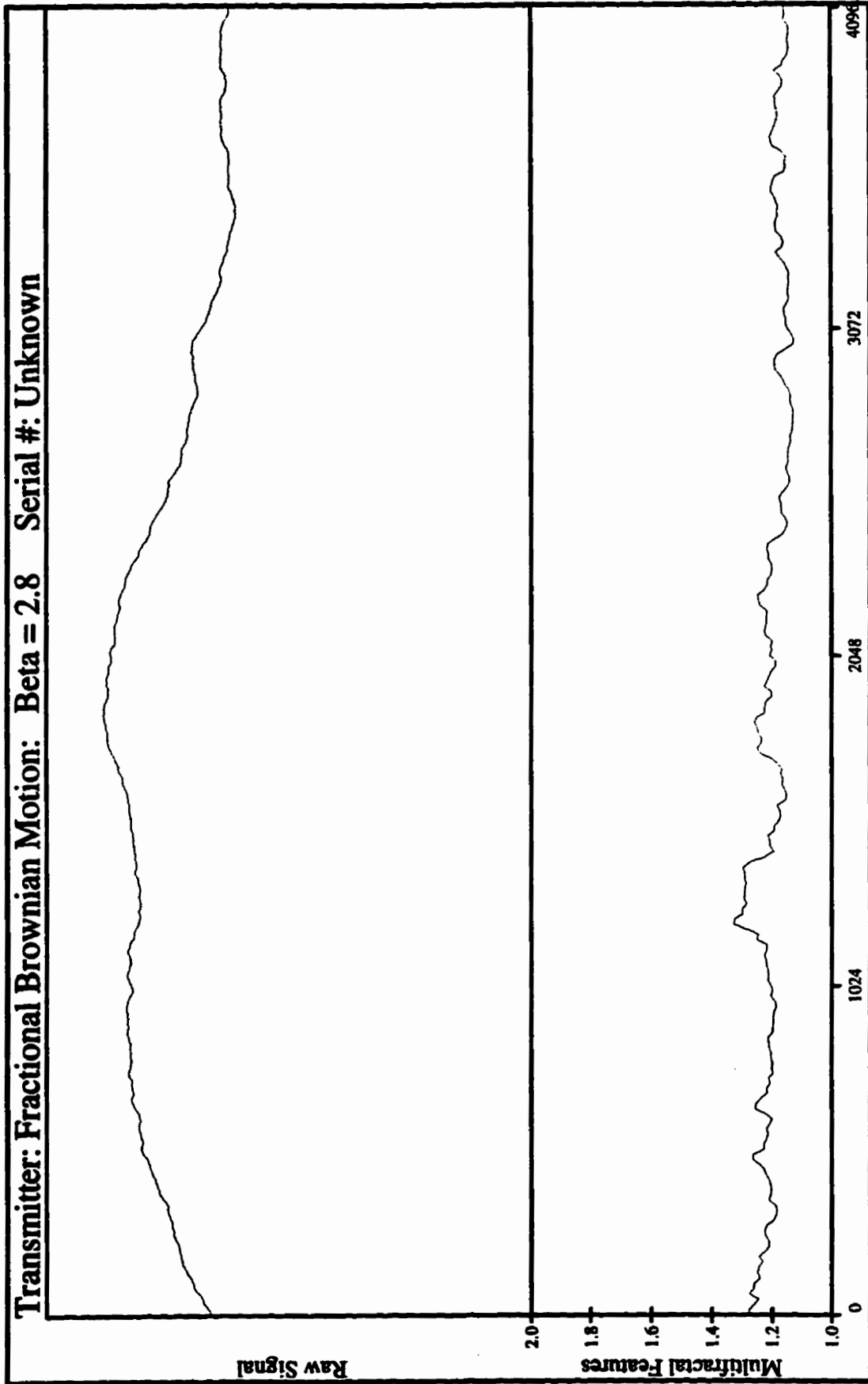
The images are arranged so that they represent increasing theoretical fractal dimensions from 1.1 through 1.9. An image of the fractal dimension trajectory calculated using both segmentation parameters and feature extraction parameters is provided for comparison purposes.

Immediately following the images is the source code of the software used to generate the fBm signals.

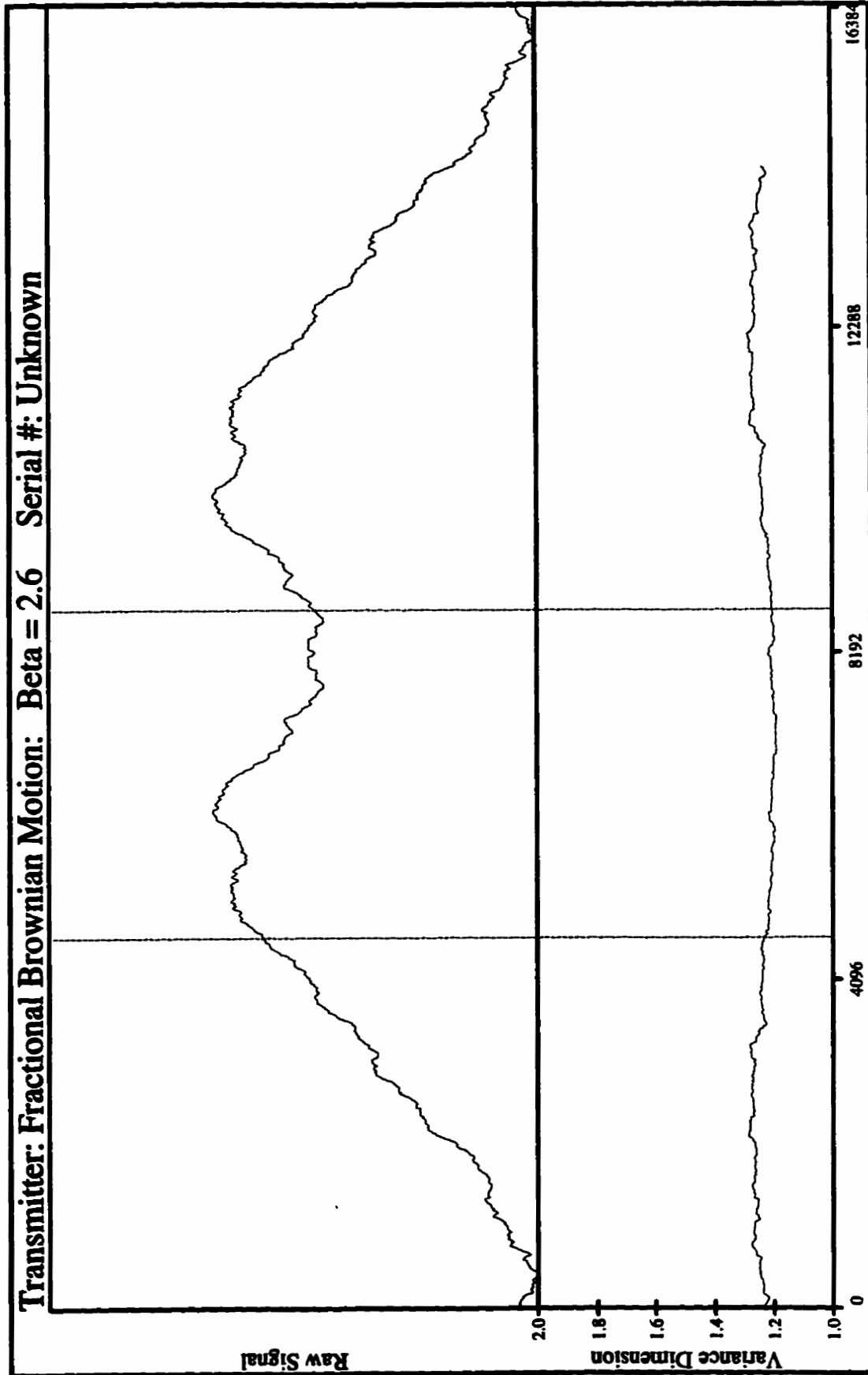
Segmentation, $D_{\sigma} = 1.1$



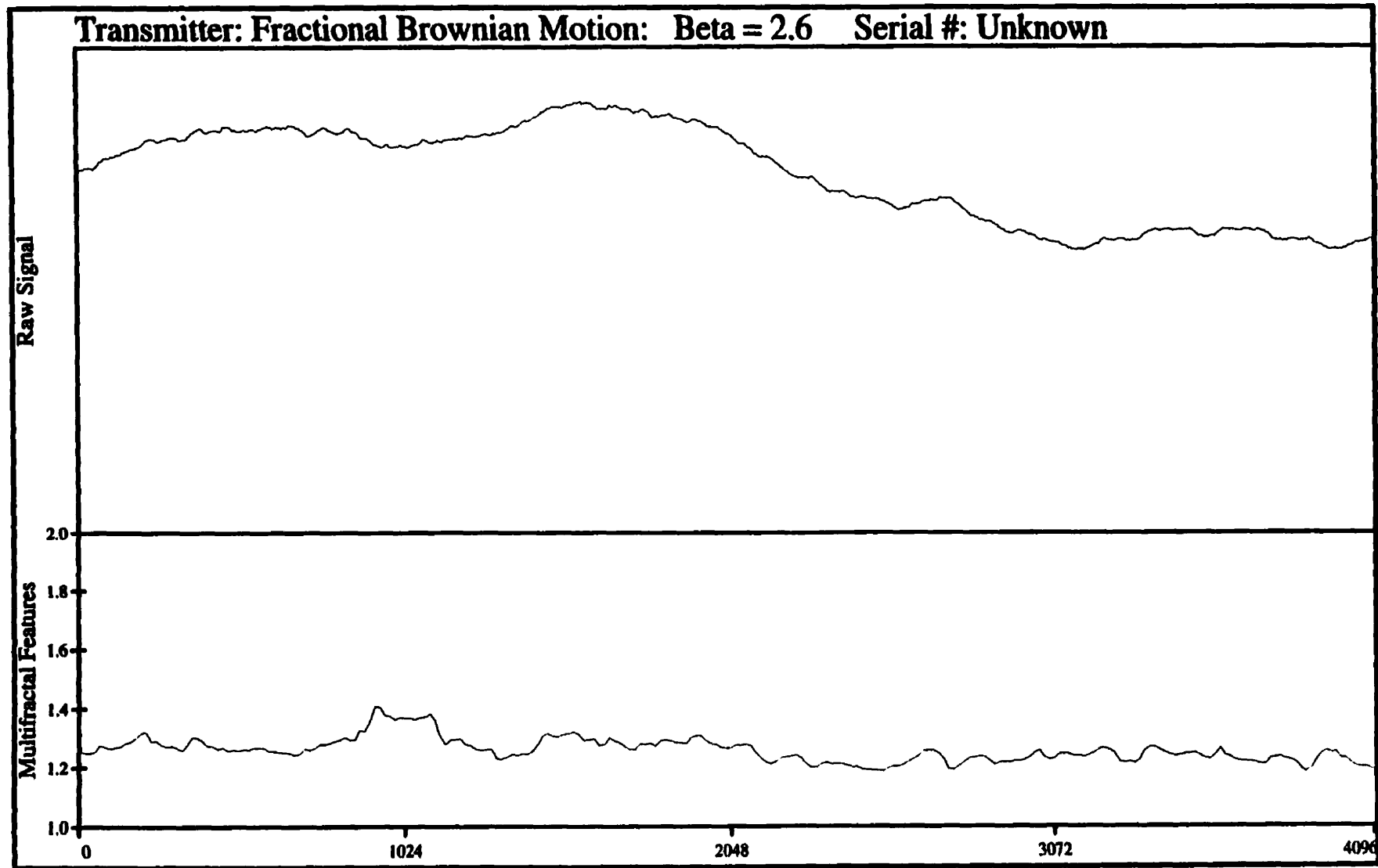
Feature Extraction, $D_{\sigma} = 1.1$



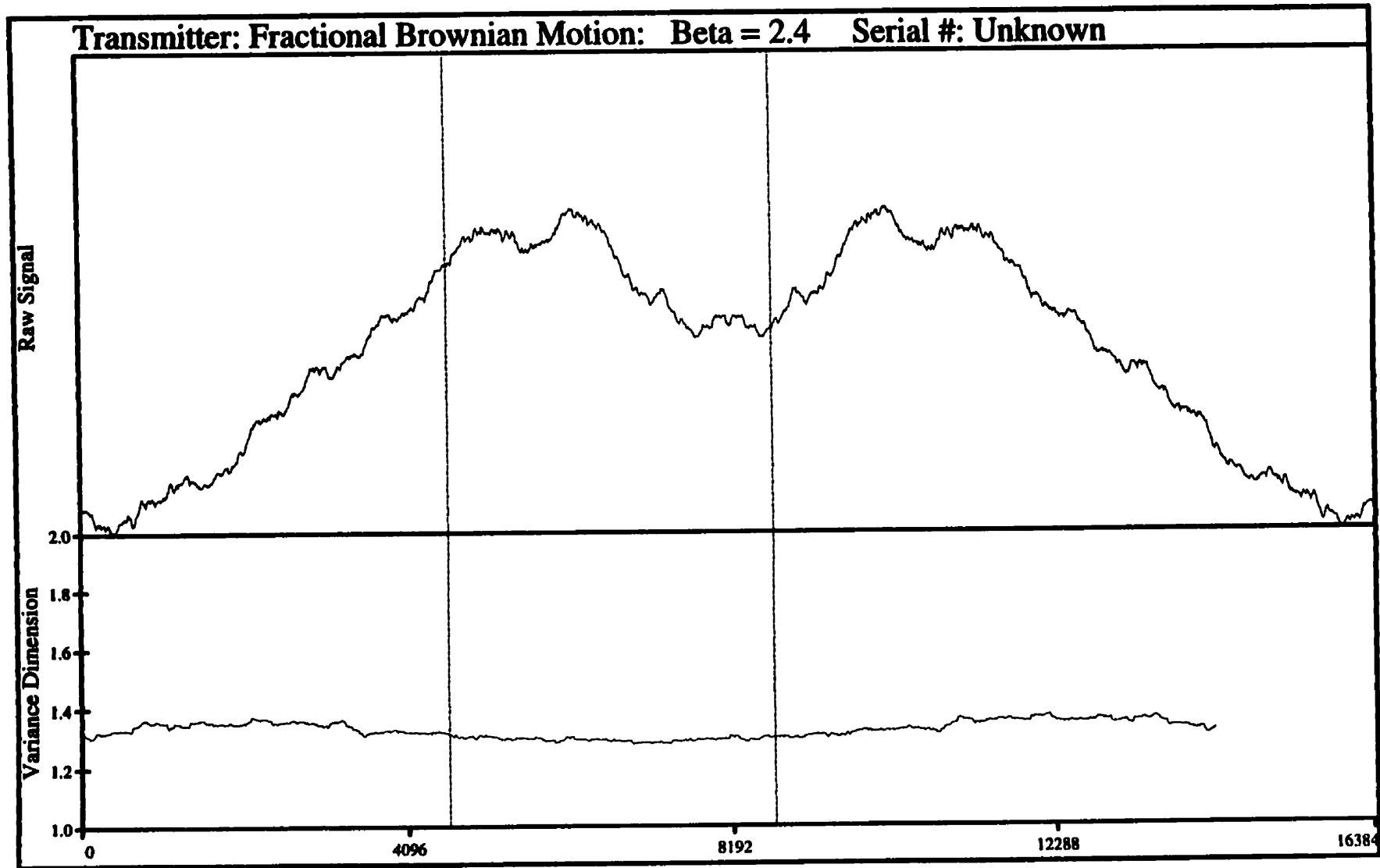
Segmentation, $D_{\sigma} = 1.2$



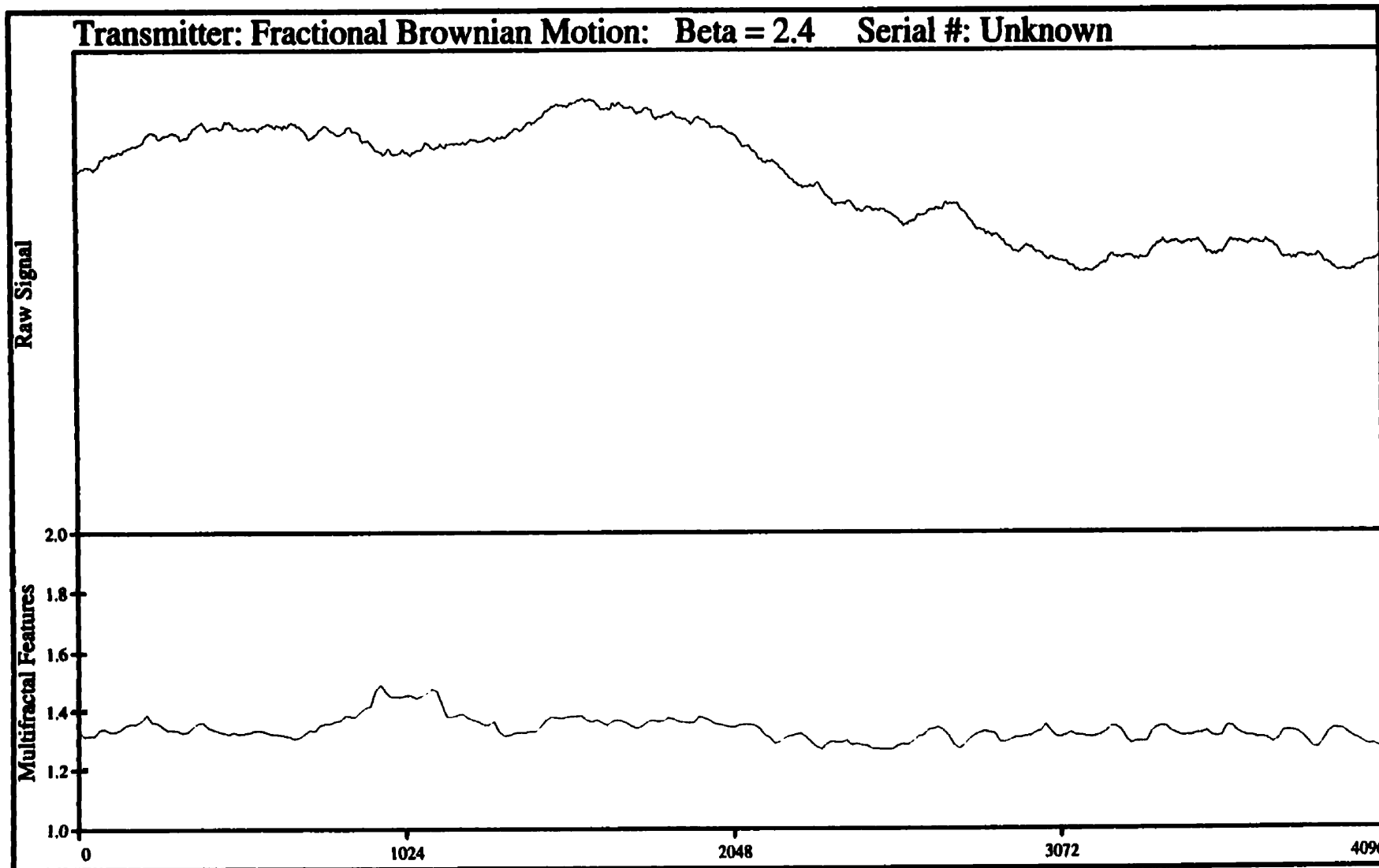
Feature Extraction, $D_{\sigma} = 1.2$



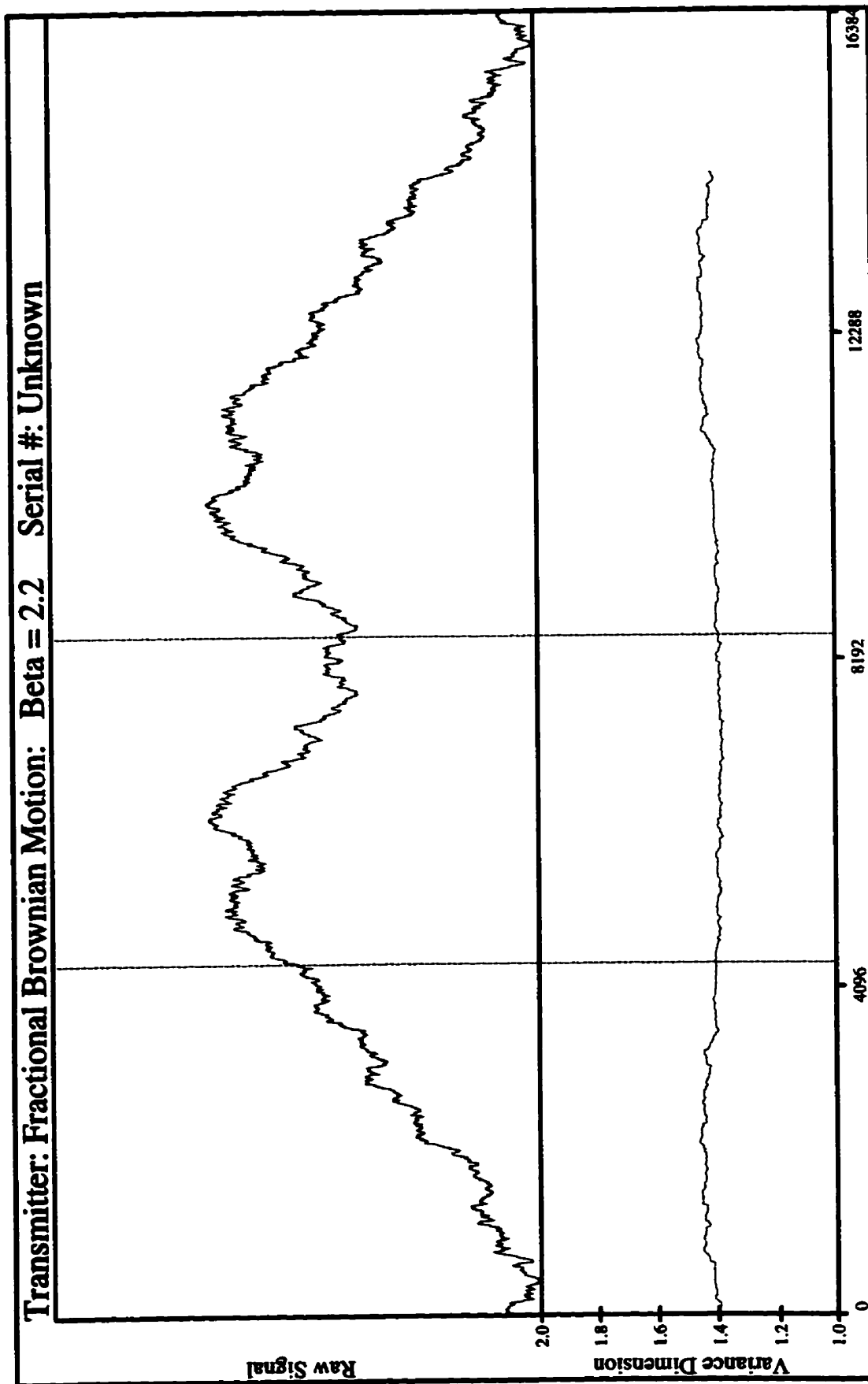
Segmentation, $D_{\sigma} = 1.3$



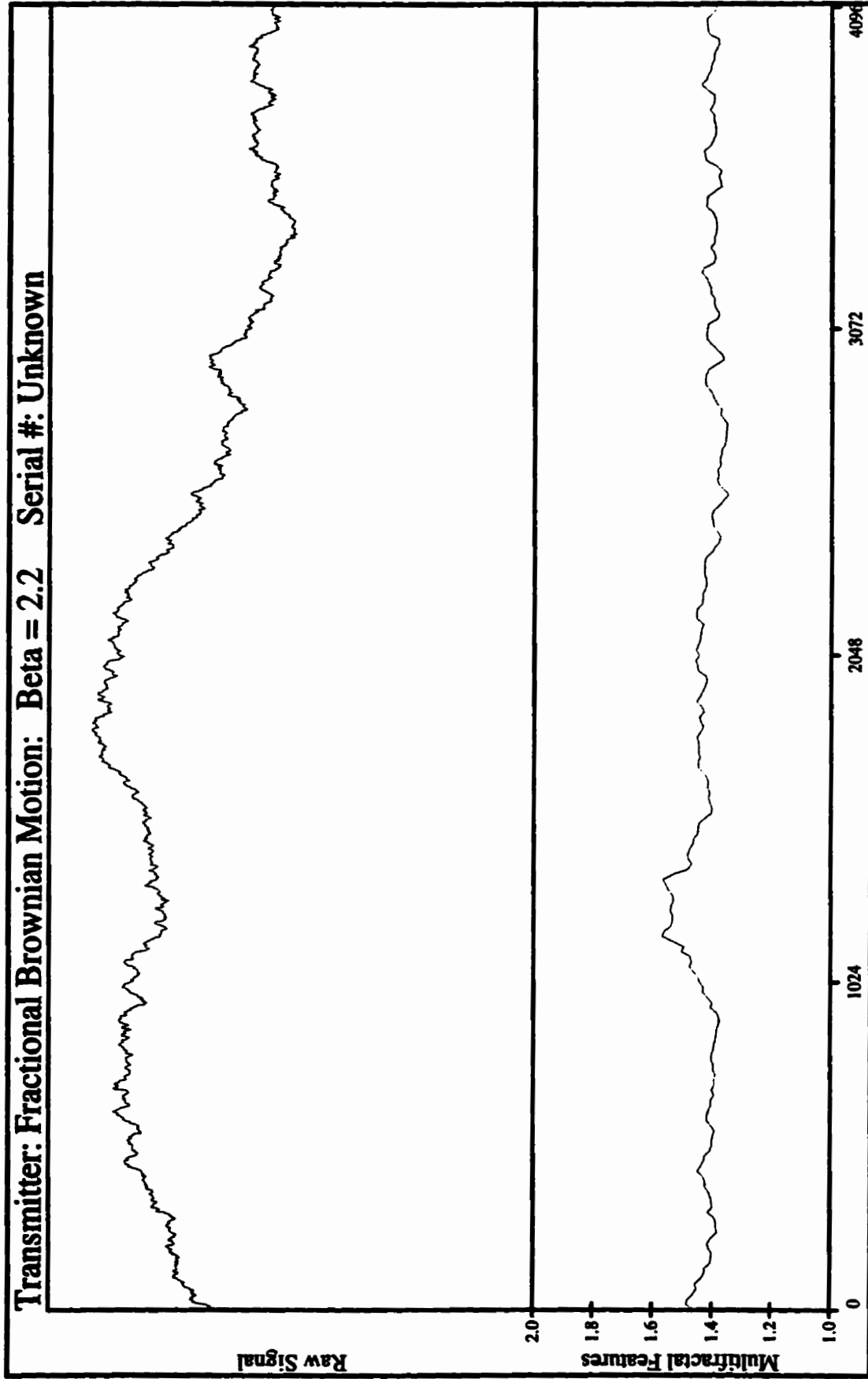
Feature Extraction, $D_{\sigma} = 1.3$



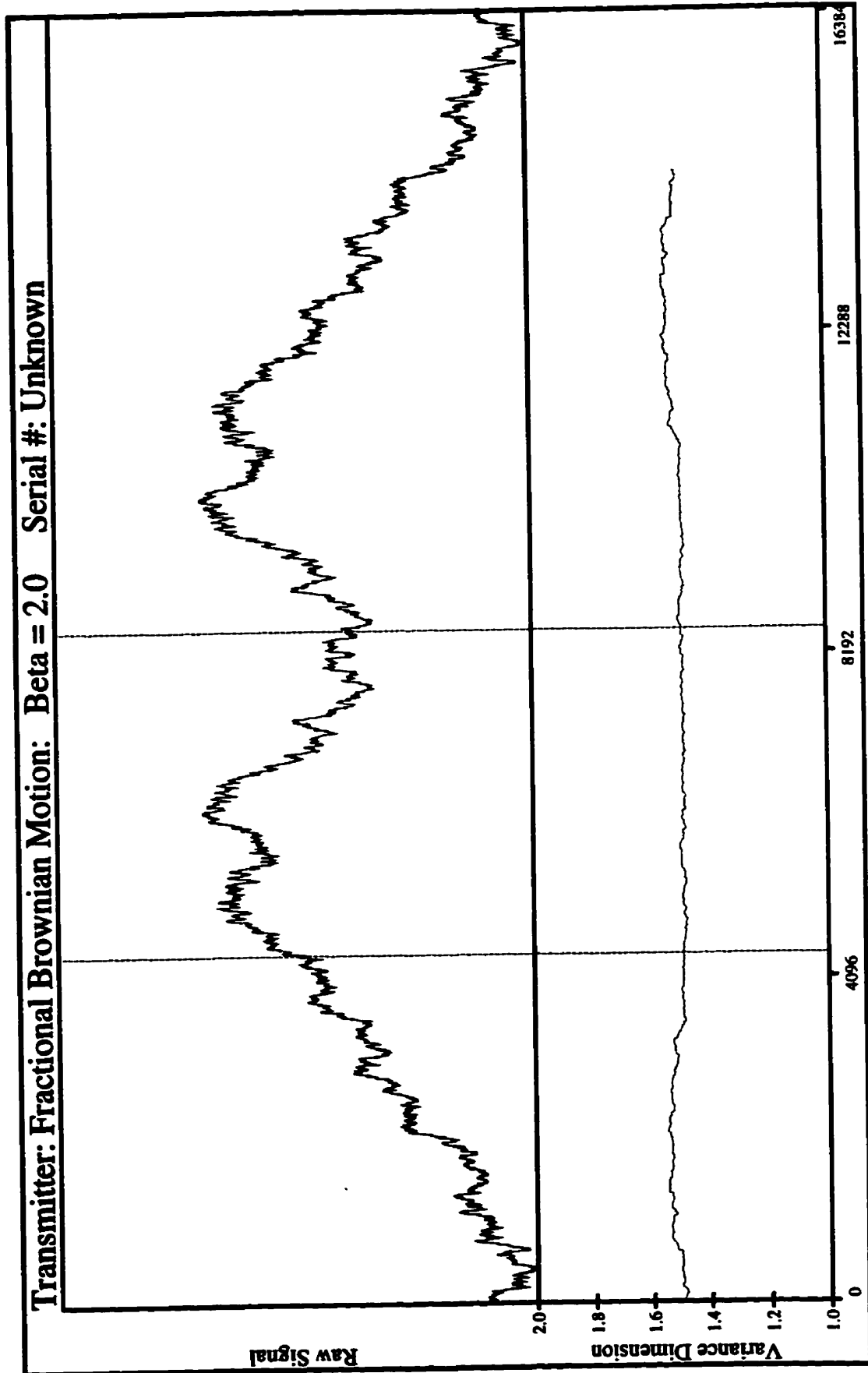
Segmentation, $D_{\sigma} = 1.4$



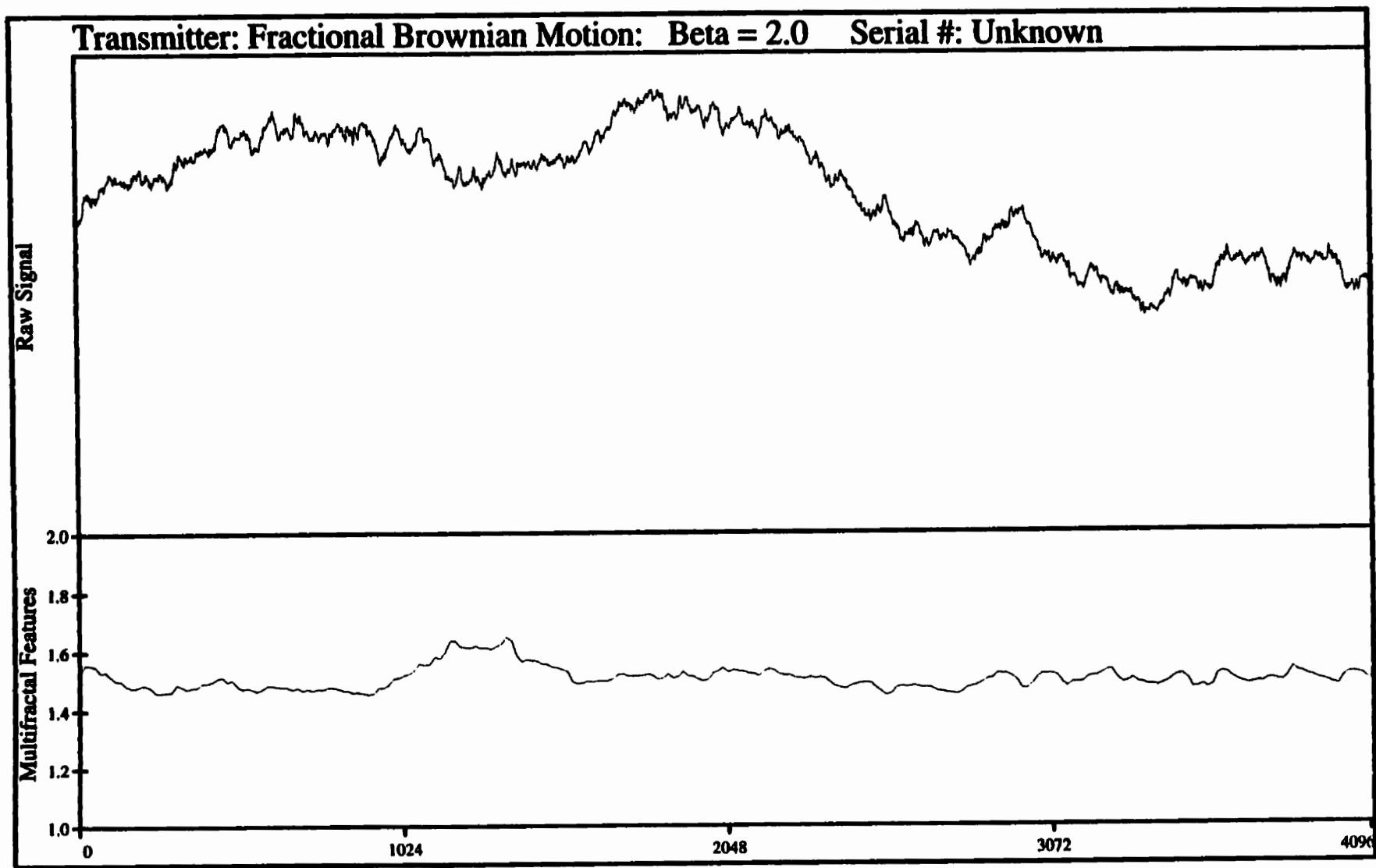
Feature Extraction, $D_G = 1.4$



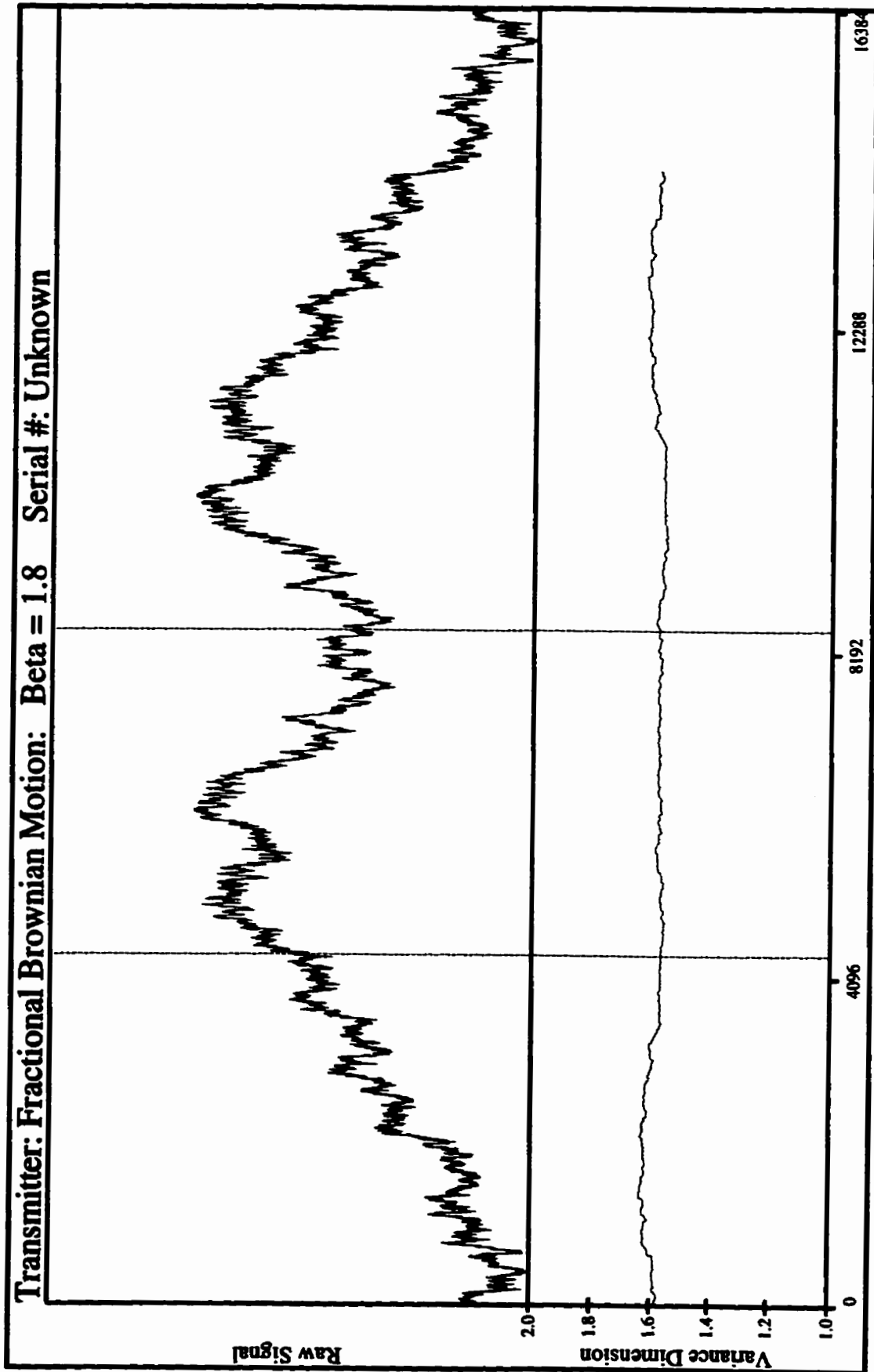
Segmentation, $D_{\sigma} = 1.5$



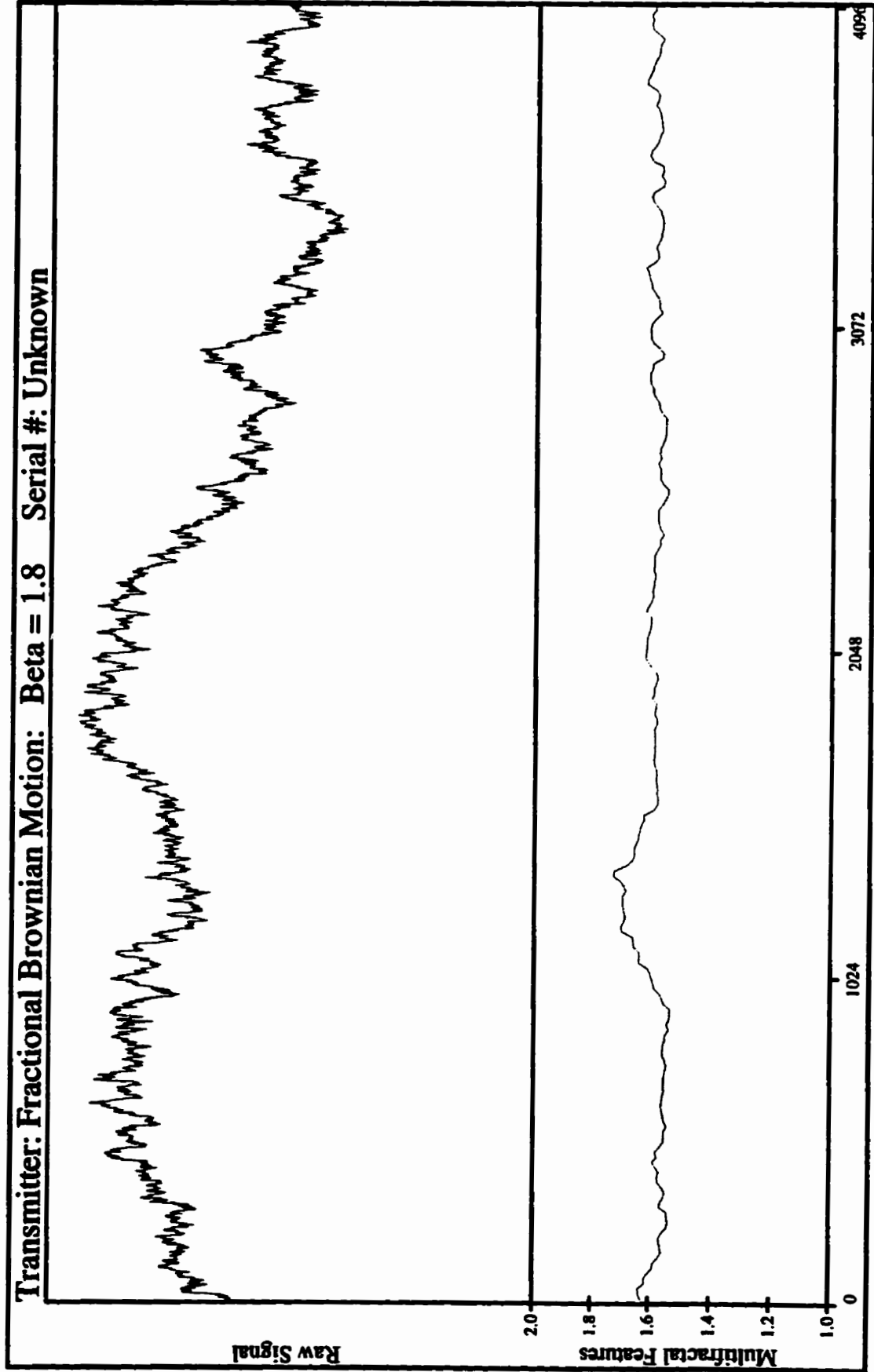
Feature Extraction, $D_{\sigma} = 1.5$



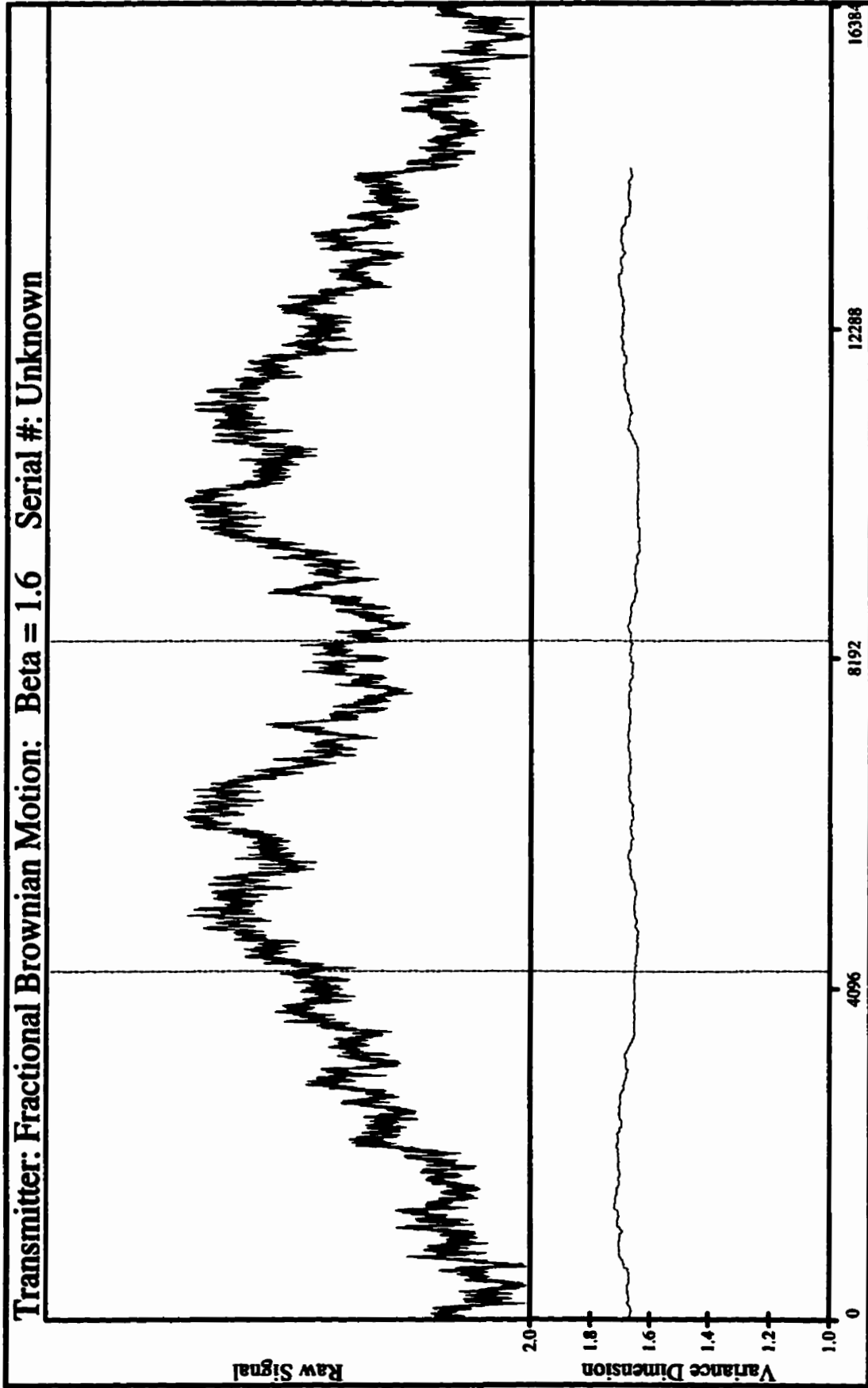
Segmentation, $D_{\sigma} = 1.6$



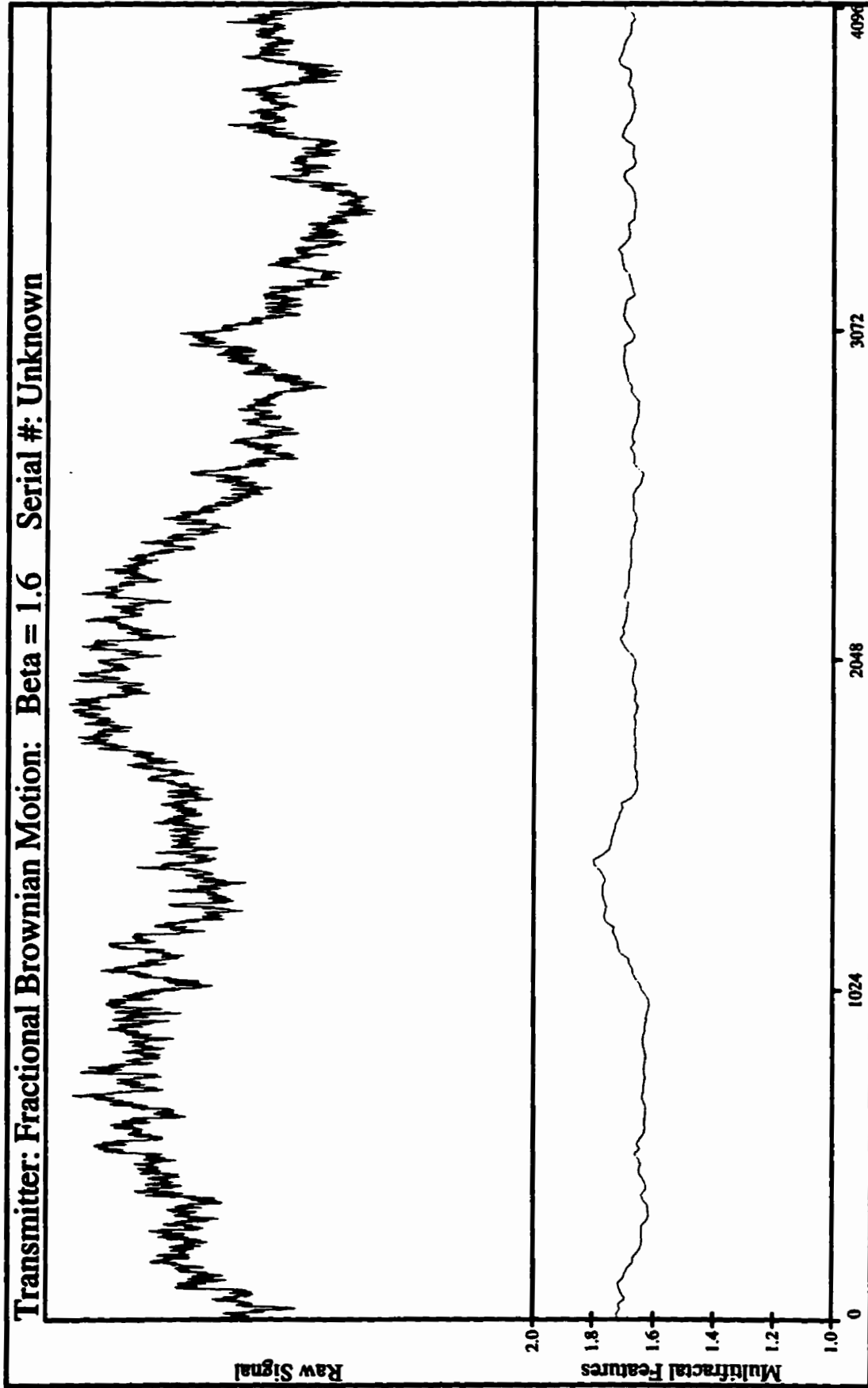
Feature Extraction, $D_{\sigma} = 1.6$



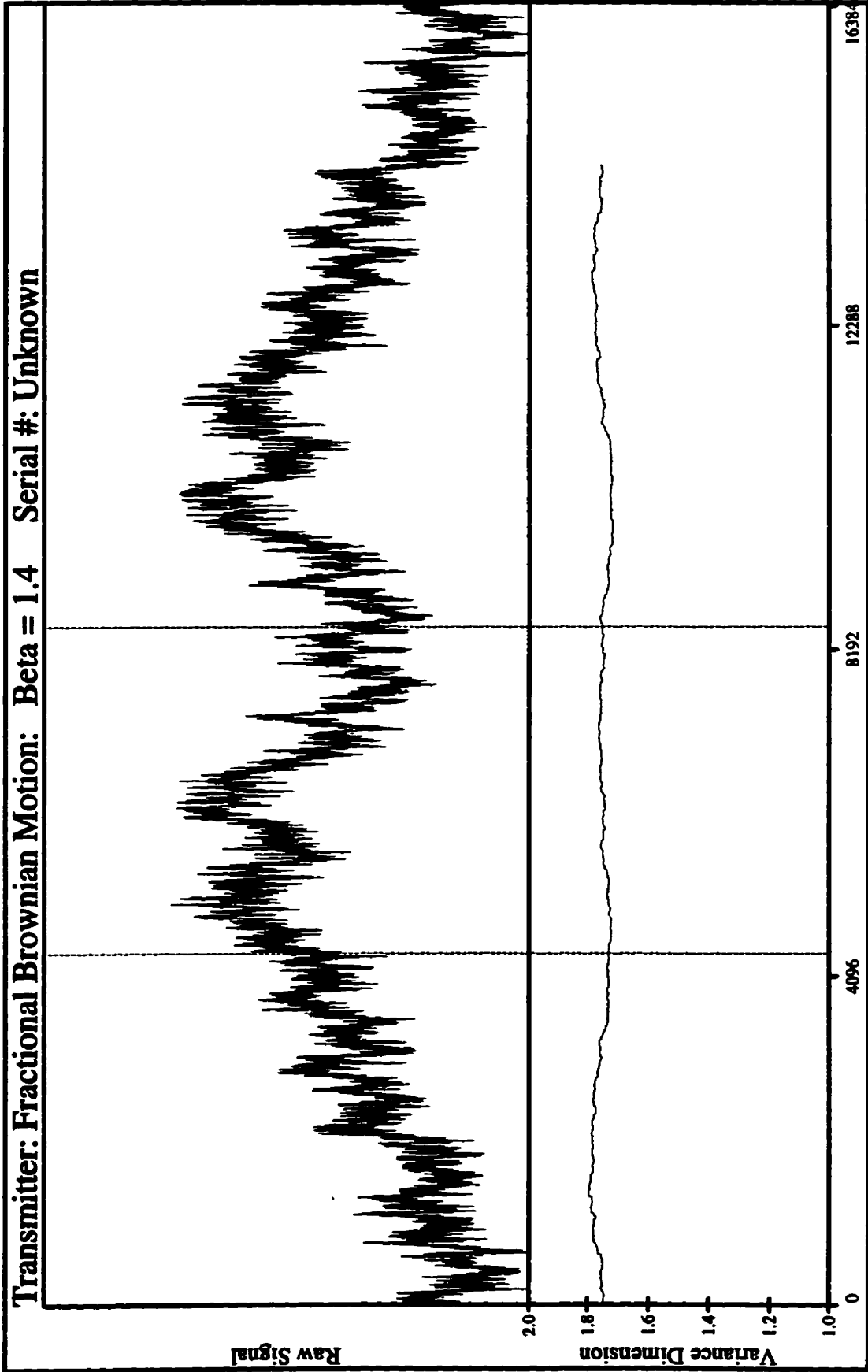
Segmentation, $D_{\sigma} = 1.7$



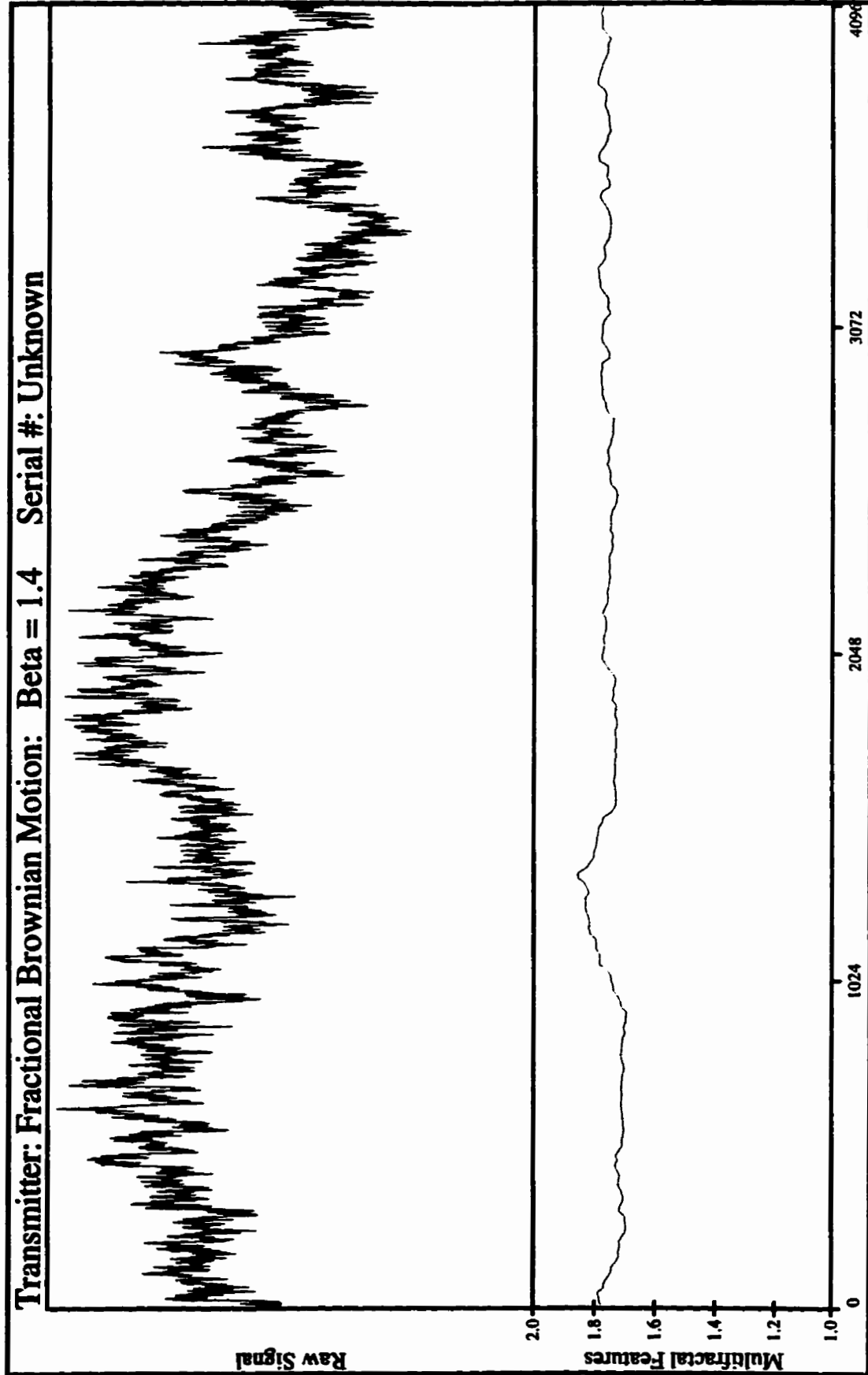
Feature Extraction, $D_{\sigma} = 1.7$



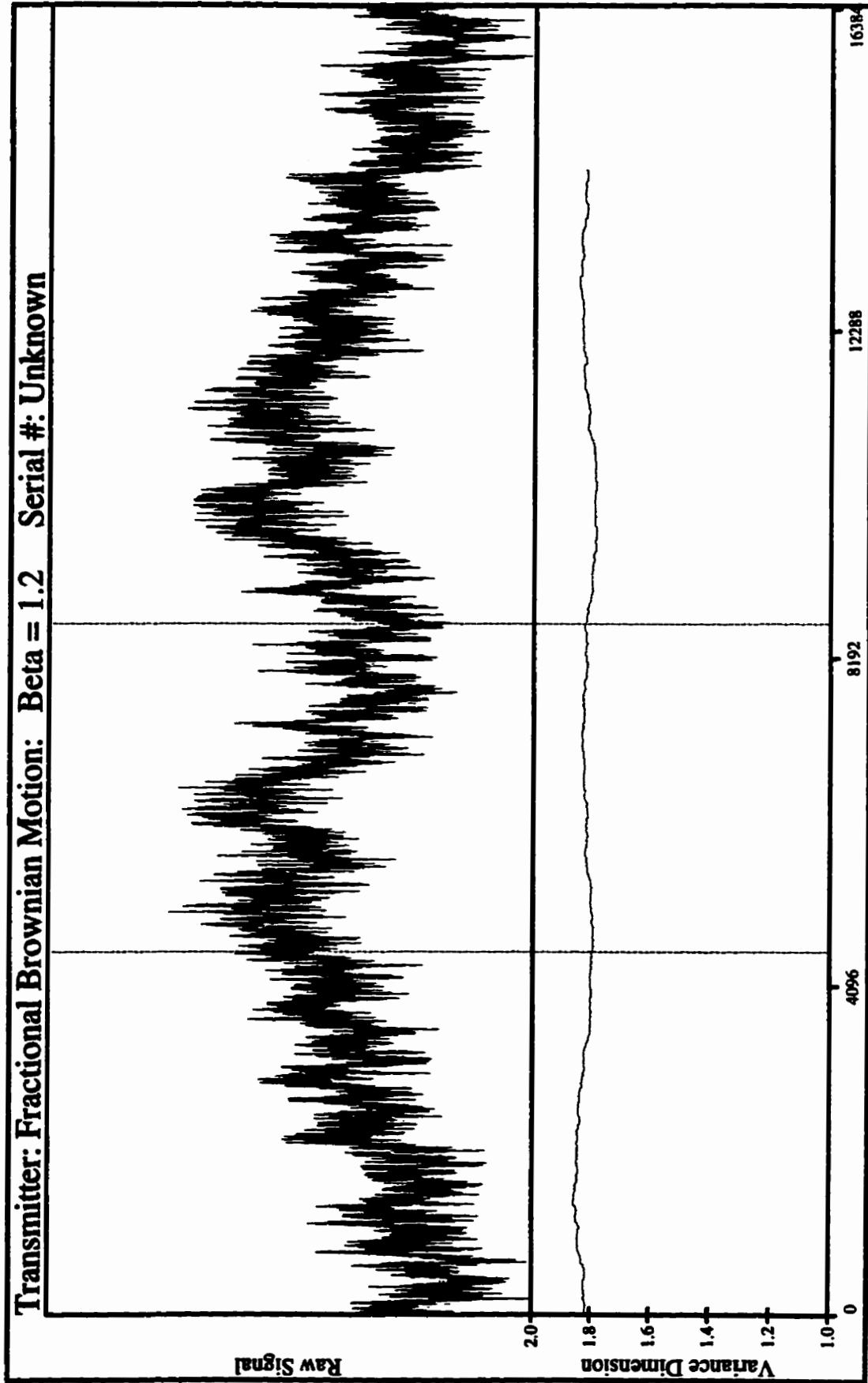
Segmentation, $D_{\sigma} = 1.8$



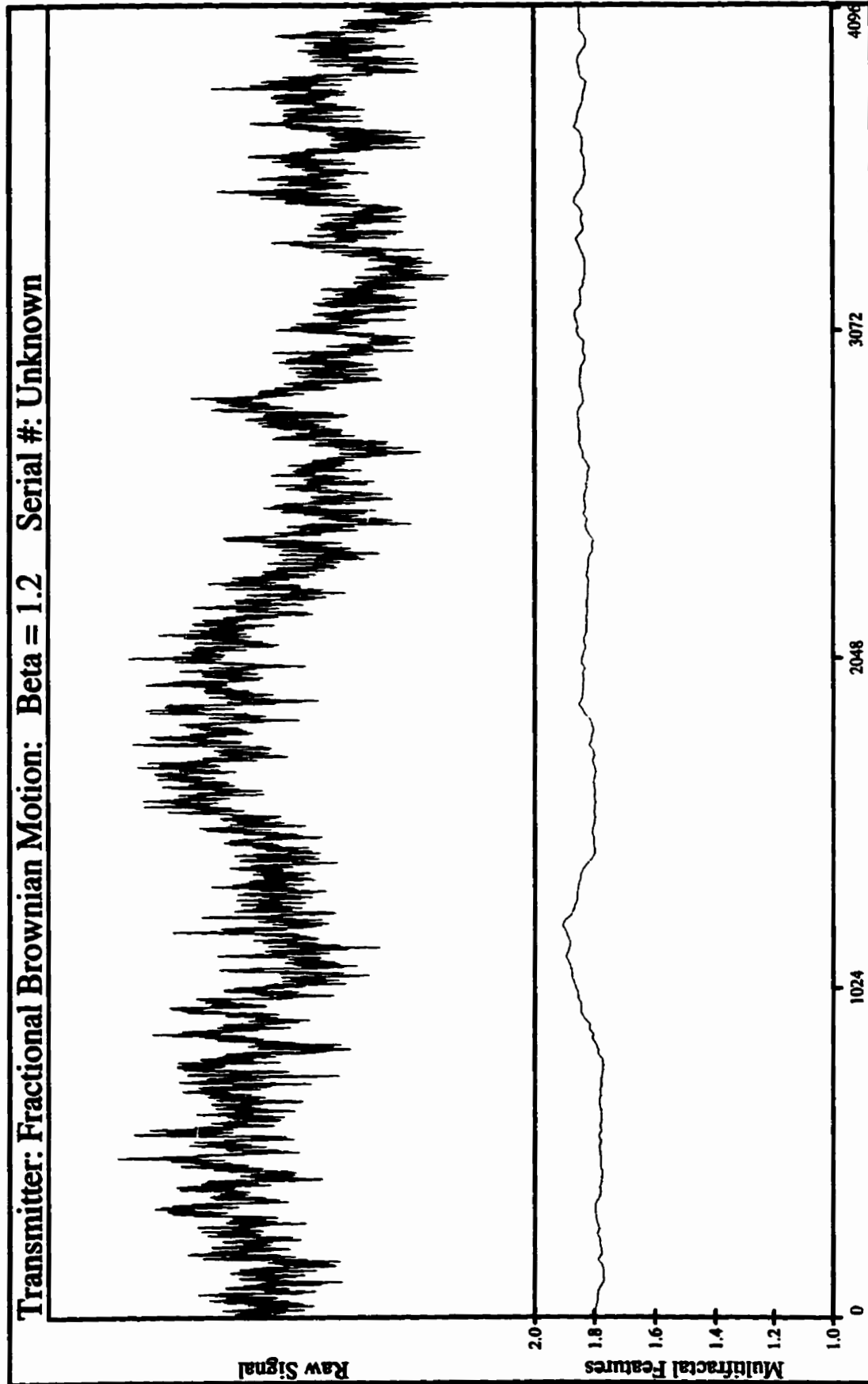
Feature Extraction, $D_{\sigma} = 1.8$



Segmentation, $D_{\sigma} = 1.9$



Feature Extraction, $D_{\sigma} = 1.9$



SOURCE CODE FOR FBM.C

```

// FBM.c
// This program produces Fractional Brownian Motion
// using Direct Spectral Filtering.
// @1997 Don Shaw

#include <stdio.h>
#include <string.h>
#include <math.h>

float gasdev(long *idum);
void four1(float data[], unsigned long nn, int isign);
main()
{
    FILE *fileptr;
    int ctr, number;
    long init=-3;
    char name[30];
    float beta, maxdev, biggest, rannum[100000],smallest;
    printf("\nHow many discrete points in fbm file?(int power of 2)\n");
    scanf("%d",&number);
    for (ctr=0; ctr<=(number-1); ctr++) {
        rannum[2*ctr]=gasdev(&init);
    }

    printf("\n\nWhat is the value of Beta?");
    scanf("\n%f",&beta);
    if (beta >= 1.0) {          /* Not white noise? */
        four1(rannum-1, number, 1);
        strcpy(name,"ftran");
        fileptr=fopen(name, "w");
        if (fileptr == NULL) {
            printf("Error\n");
        }
        else {
            for (ctr=0; ctr <=(number-1); ctr++) {
                fprintf(fileptr, "%f\n",rannum[2*ctr]);
            }
        }
        fclose (fileptr);
        rannum[0]=0.0;

        for (ctr=1; ctr<=(number/2); ctr++) {
            rannum[(2*ctr)]*=pow(((double)ctr)/((double)number-1.0),(-beta/2.0));
        }
        for (ctr=(number/2); ctr<=(number-1); ctr++) {
            rannum[(2*ctr)]=rannum[2*(number-ctr)];
        }
    }
}

```

Appendix C: Data and Software for Fractal Dimension Verification

```

        strcpy(name,"fil");
        fileptr=fopen(name, "w");
        if (fileptr == NULL) {
            printf("Error\n");
        }
        else {
            for (ctr=0; ctr <= (number-1); ctr++) {
                fprintf(fileptr, "%f\n",rannum[2*ctr]);
            }
        }
        fclose (fileptr);

        fourl(rannum-1, number, -1);
    }

    maxdev=25000;
    biggest=0.0;
    for (ctr=0; ctr<=(2*number); ctr+=2) {
        if (fabs(rannum[ctr])>biggest) {
            biggest=fabs(rannum[ctr]);
        }
    }
    biggest/=maxdev;
    for (ctr=0; ctr<=(2*number); ctr+=2) {
        rannum[ctr]/=biggest;
    }
    smallest=0.0;
    for (ctr=0; ctr<=(2*number); ctr+=2) {
        if (rannum[ctr]<smallest) {
            smallest=rannum[ctr];
        }
    }
    for (ctr=0; ctr<=(2*number); ctr+=2) {
        rannum[ctr]+=fabs(smallest);
    }

    printf("\nWhat is filename?: ");
    fflush(stdin);
    gets( name );
    fileptr=fopen(name, "w");
    if (fileptr == NULL) {
        printf("Error\n");
    }
    else {
        for (ctr=0; ctr <= (number-1); ctr++) {
            fprintf(fileptr, "%d\n",(int)rannum[2*ctr]);
        }
        fprintf(fileptr, "Fractional Brownian Motion: \n");
        fprintf(fileptr, "Beta = %.1f\n", beta);
    }
    fclose (fileptr);

    return 0;
}

```

Appendix C:Data and Software for Fractal Dimension Verification

```
// The following routines are from the book Numerical Recipes in C
// Copyright 1992 Cambridge University Press [PTFV92]

/*****gaussian deviates*****/
float gasdev(long *idum)
{
    float ran1(long *idum);
    static int iset=0;
    static float gset;
    float fac, rsq, v1, v2;
    if (iset==0) {
        do {
            v1=2.0*ran1(idum)-1.0;
            v2=2.0*ran1(idum)-1.0;
            rsq=v1*v1+v2*v2;
        } while (rsq >= 1.0 || rsq == 0.0);
        fac=sqrt(-2.0*log(rsq)/rsq);
        gset=v1*fac;
        iset=1;
        return v2*fac;
    } else {
        iset=0;
        return gset;
    }
}

/*****uniform deviates*****/
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

float ran1(long *idum)
{
    int j;
    long k;
    static long iy=0;
    static long iv[NTAB];
    float temp;
    if (*idum <= 0 || !iy) {
        if (-(*idum) < 1) *idum=1;
        else *idum = -(*idum);
        for (j=NTAB+7; j>=0; j--) {
            k=(*idum)/IQ;
            *idum=IA*( *idum-k*IQ)-IR*k;
            if (*idum<0) *idum +=IM;
            if (j < NTAB) iv[j] = *idum;
        }
        iy=iv[0];
    }

```

```

}
k=(*idum)/IQ;
*idum=IA*( *idum-k*IQ)-IR*k;
if (*idum < 0) *idum += IM;
j=iy/NDIV;
iy=iv[j];
iv[j] = *idum;
if ((temp=AM*iy) > RNMX) return RNMX;
else return temp;
}

/*****F.F.T*****/
#include <math.h>
#define SWAP(a,b) tempr=(a);(a)=(b);b=tempr

void fourl(float data[], unsigned long nn, int isign)
{
    unsigned long n, mmax, m, j, istep, i;
    double wtemp, wr, wpr, wpi, wi, theta;
    float tempr, tempi;

    n=nn << 1;
    j=1;
    for (i=1; i<n; i+=2) {
        if (j > i) {
            SWAP(data[j], data[i]);
            SWAP(data[j+1], data[i+1]);
        }
        m=n >> 1;
        while (m >= 2 && j > m) {
            j -= m;
            m >>= 1;
        }
        j += m;
    }

    mmax=2;
    while (n > mmax) {
        istep=mmax << 1;
        theta=isign*(6.28318530717959/mmax);
        wtemp=sin(0.5*theta);
        wpr = -2.0*wtemp*wtemp;
        wpi=sin(theta);
        wr=1.0;
        wi=0.0;
        for (m=1; m<mmax; m+=2) {
            for (i=m; i<=n; i+=istep) {
                j=i+mmax;
                tempr=wr*data[j]-wi*data[j+1];
                tempi=wr*data[j+1]+wi*data[j];
                data[j]=data[i]-tempr;
                data[j+1]=data[i+1]-tempi;
                data[i] += tempr;
                data[i+1] += tempi;
            }
        }
    }
}

```


Appendix C: Data and Software for Fractal Dimension Verification

```
    }  
    wr=(wtemp=wr)*wpr-wi*wpi+wr;  
    wi=wi*wpr+wtemp*wpi+wi;  
  }  
  mmax=istep;  
}  
return;  
}
```

APPENDIX D

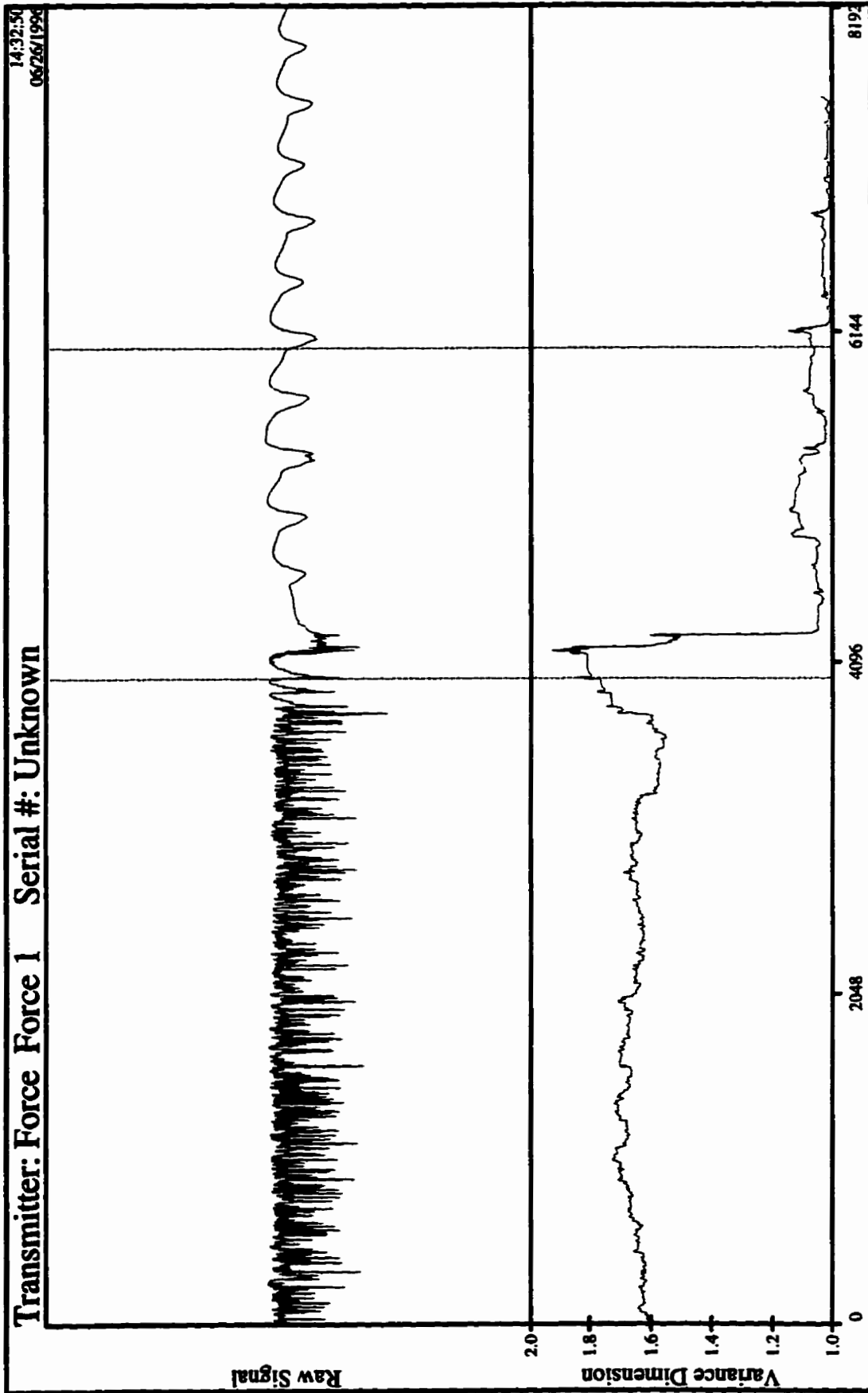
TRANSIENTS IN THE THESIS TEST SET

This appendix contains images of one raw recording from each different transmitter used in the thesis test set. The images are printed using TAC-MM with the following parameter settings:

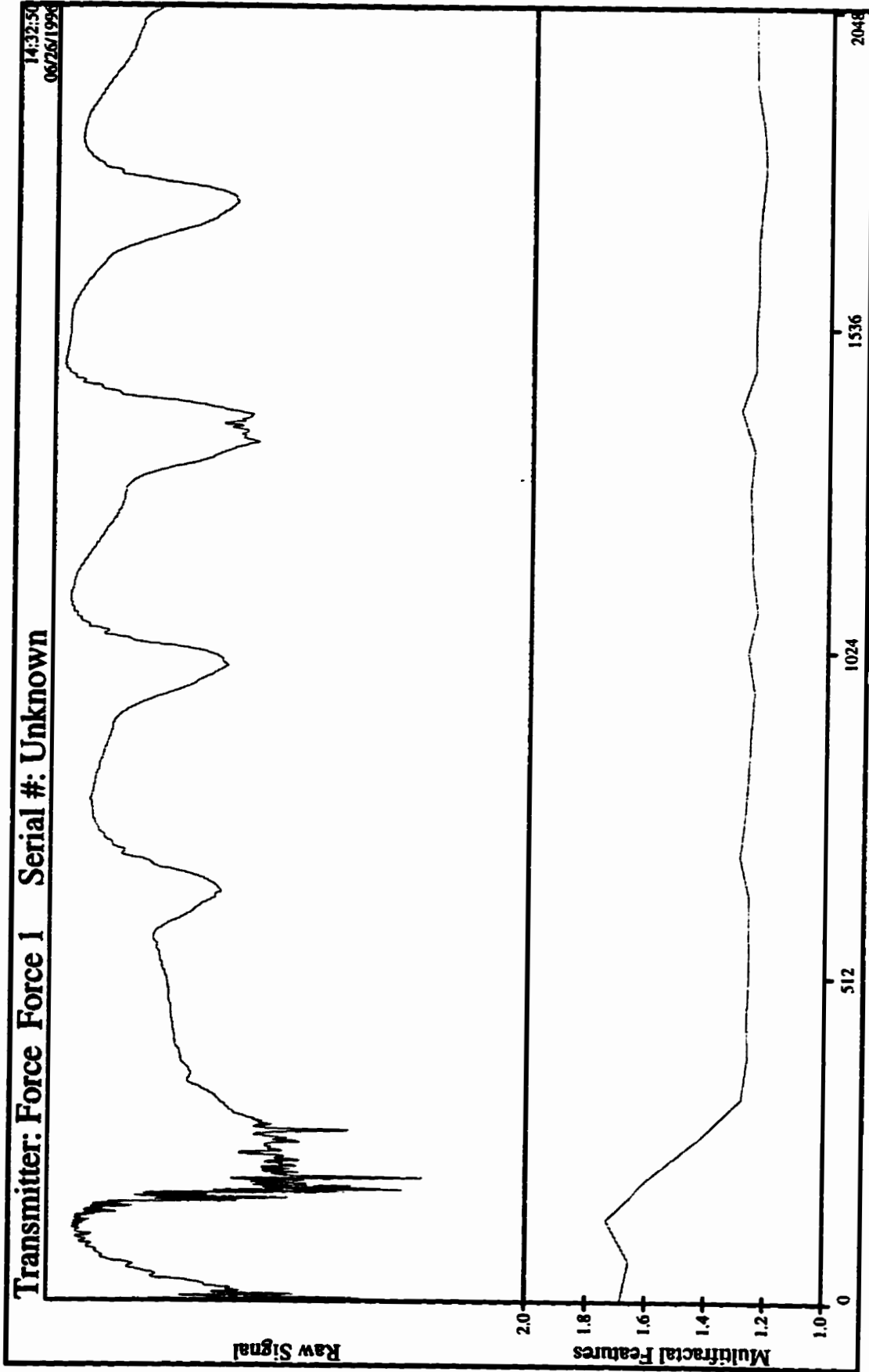
Raw File Size	8192
Transient Size	2048
Segmentation Window Size	512
Segmentation Variance Pairs	10
Segmentation Threshold	8
Feature Extraction Window Size	768
Feature Extraction Variance Pairs	8
Feature Extraction Window Shift	64

The images are arranged in the order of Classes 0 through 7 as they were assigned in Chapter 6 of this thesis. An image of the fractal dimension trajectory calculated using both segmentation parameters and feature extraction parameters is provided for comparison purposes.

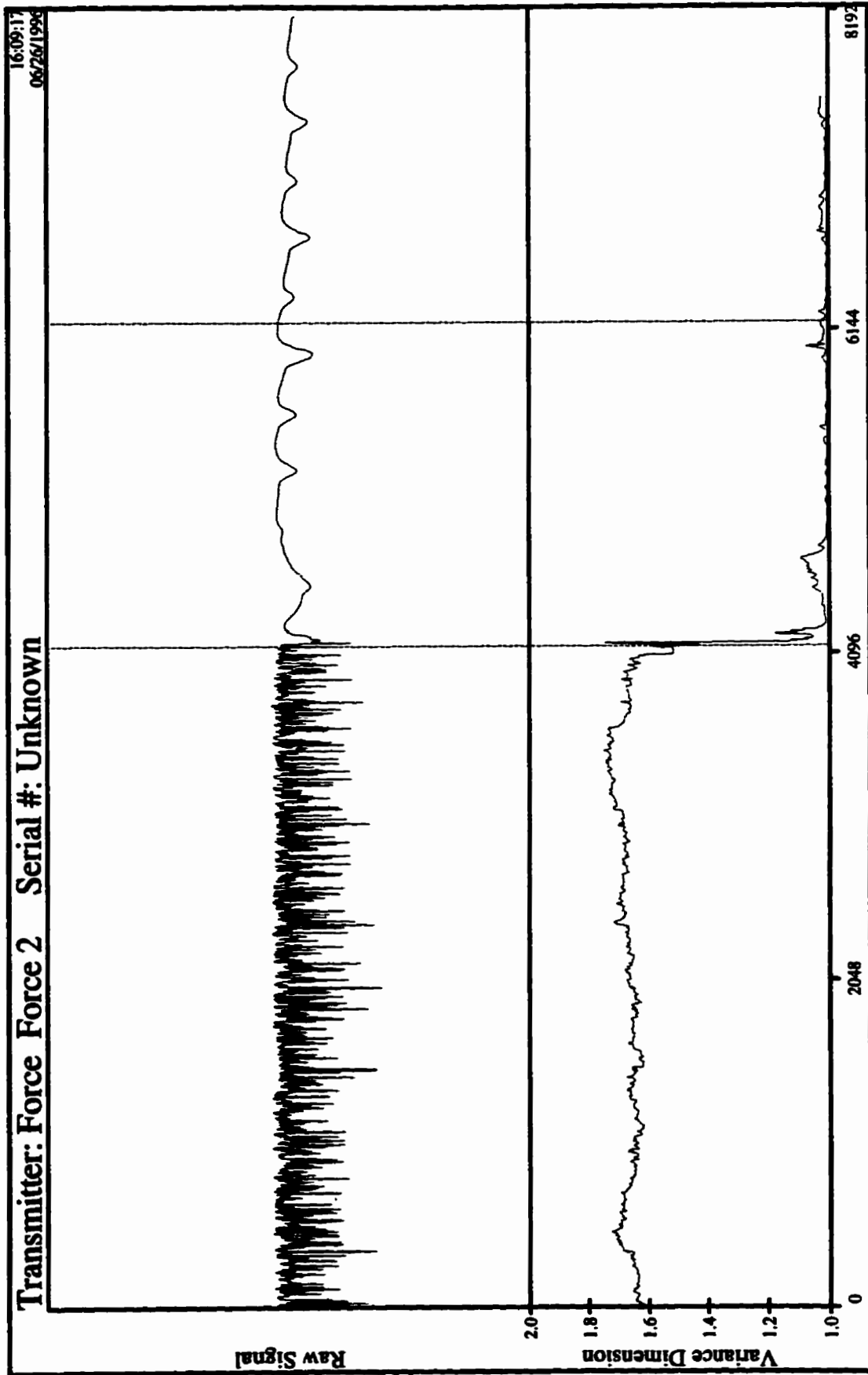
Segmentation, Class 0



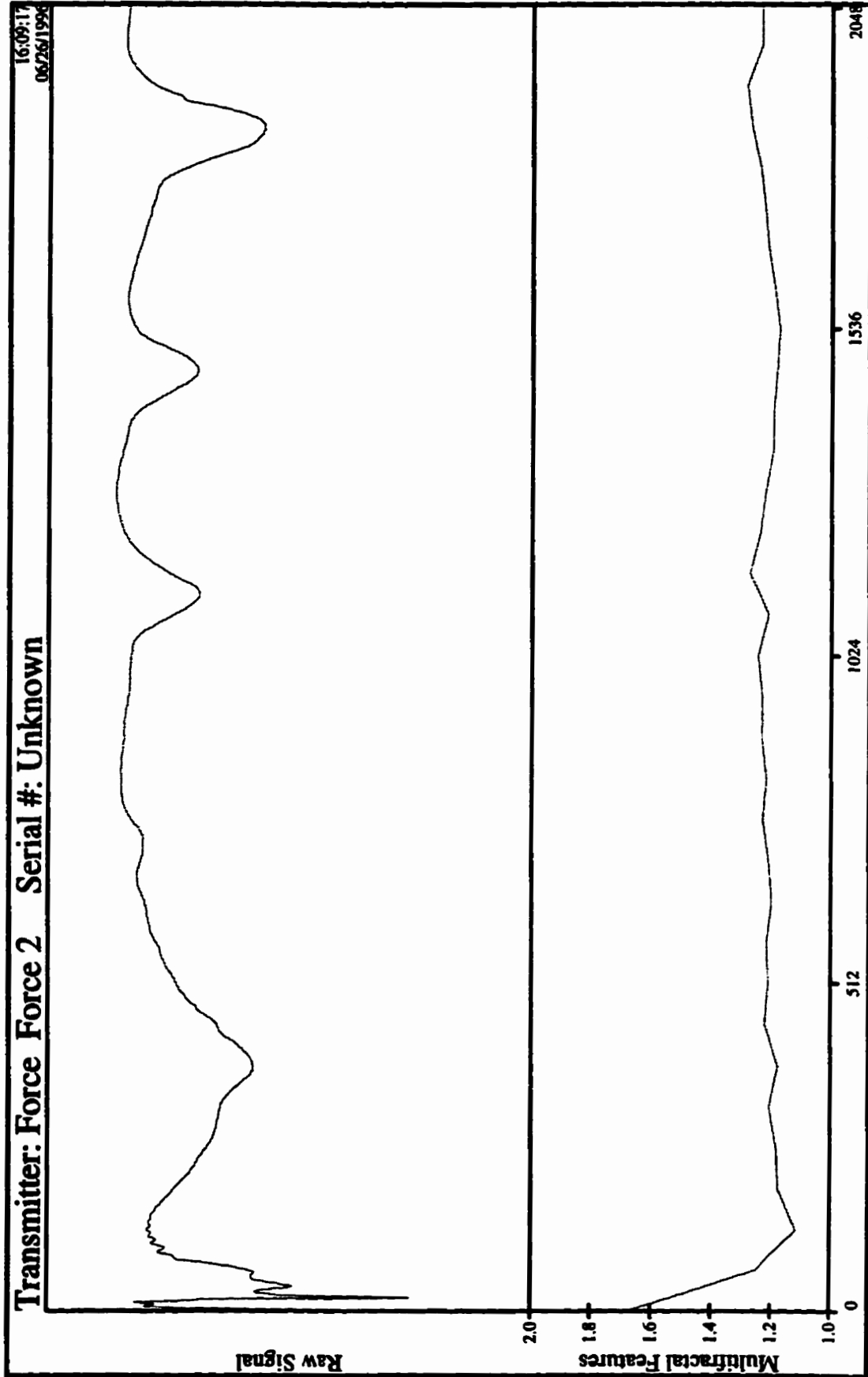
Feature Extraction, Class 0



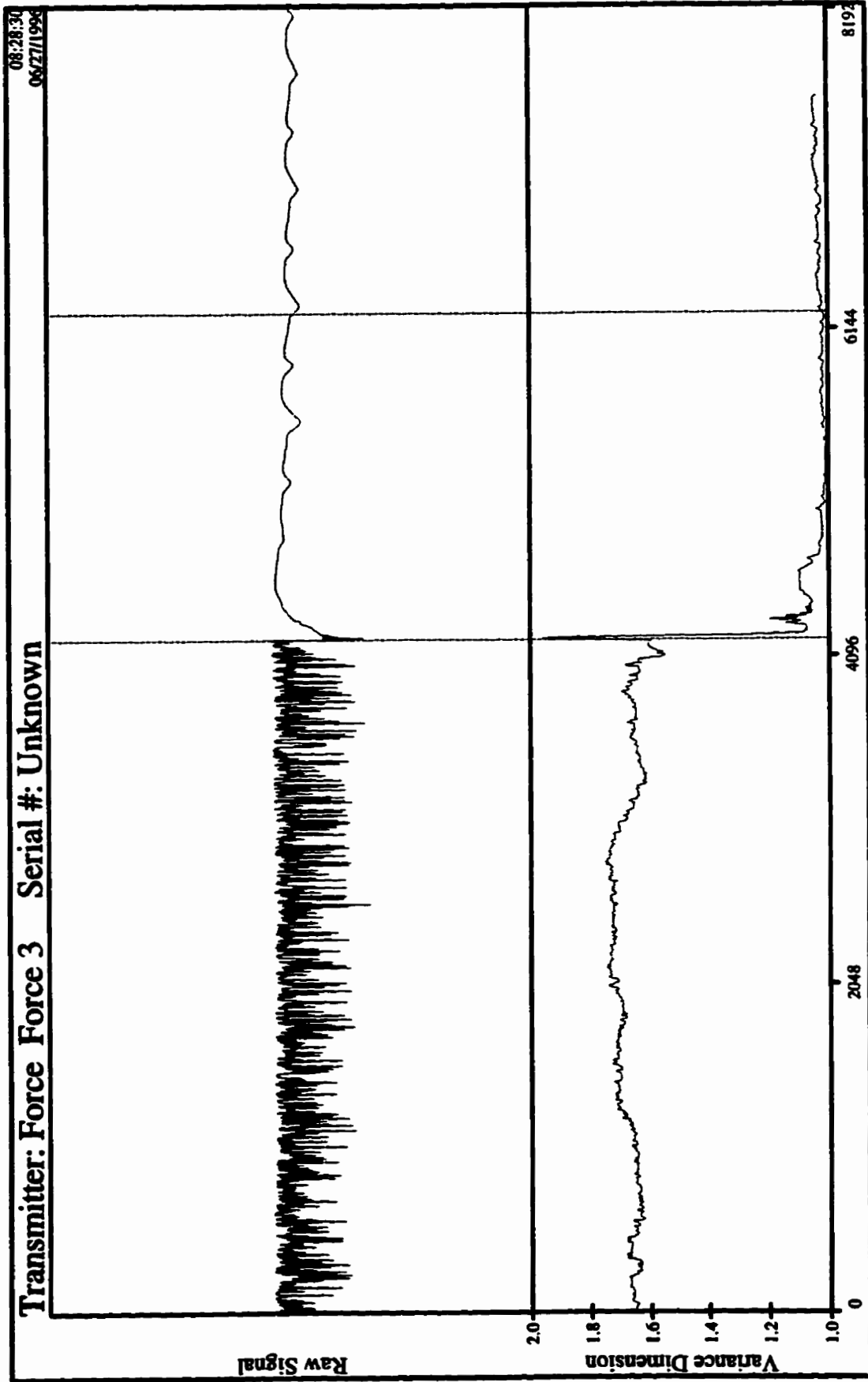
Segmentation, Class 1



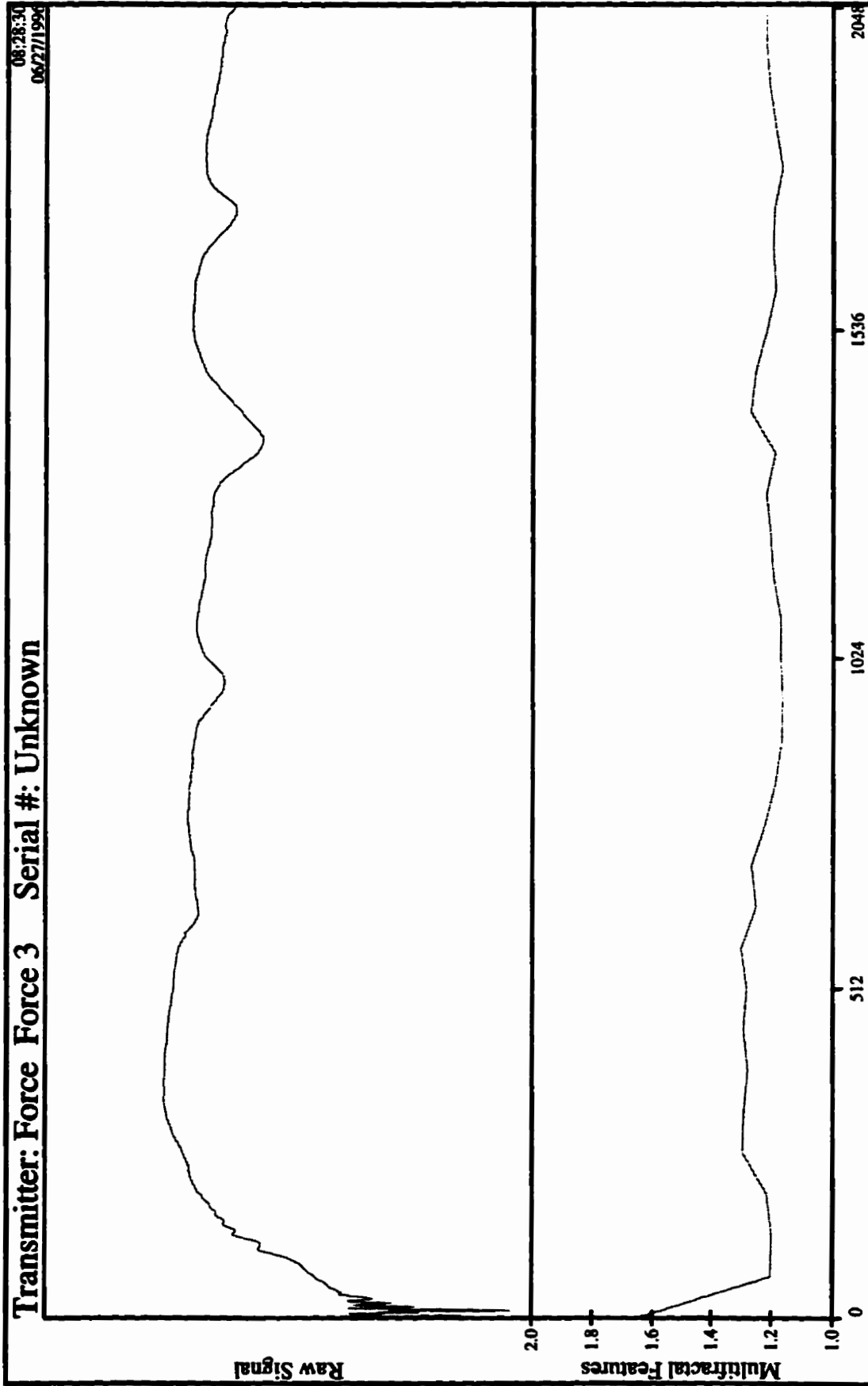
Feature Extraction, Class 1



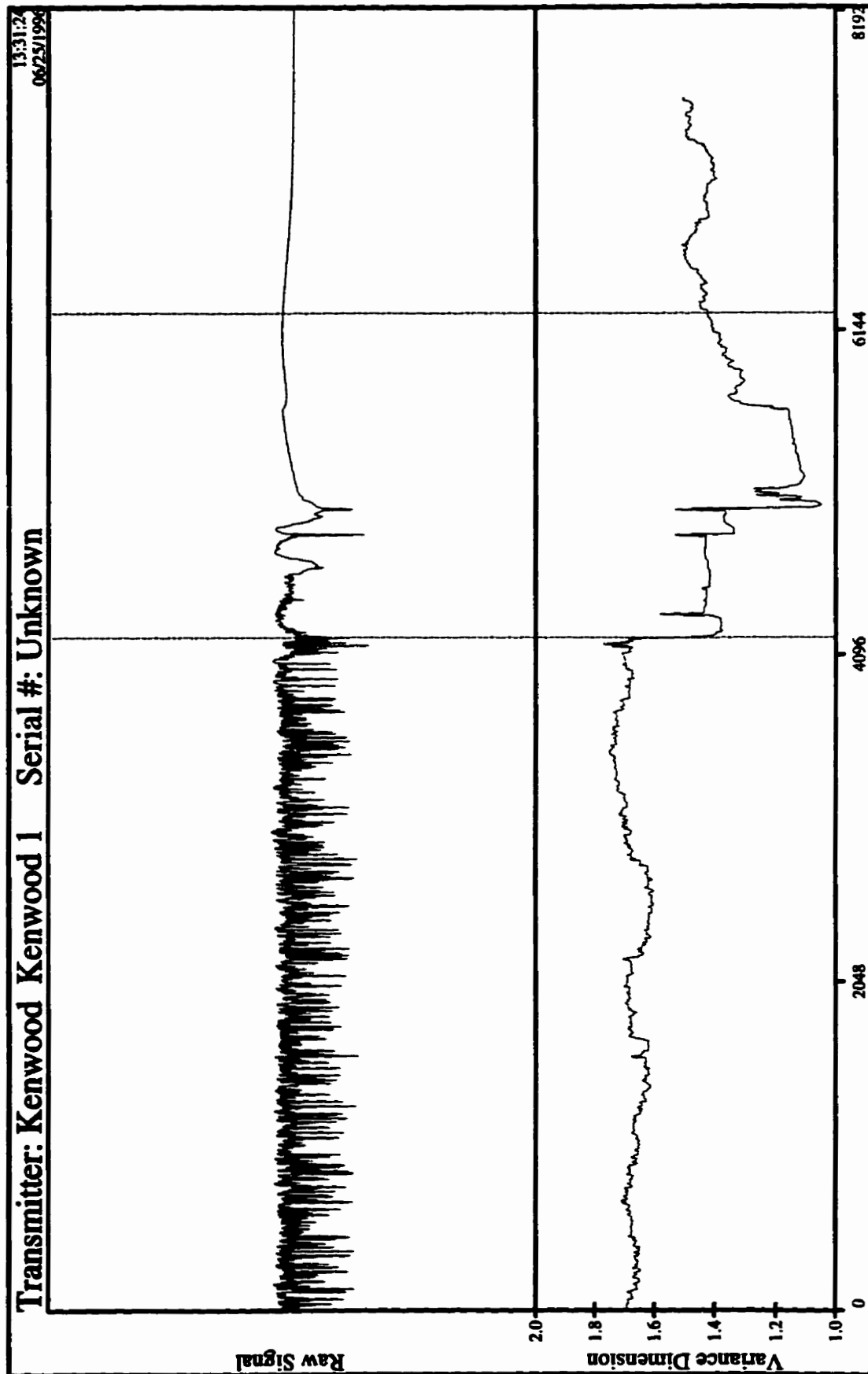
Segmentation, Class 2



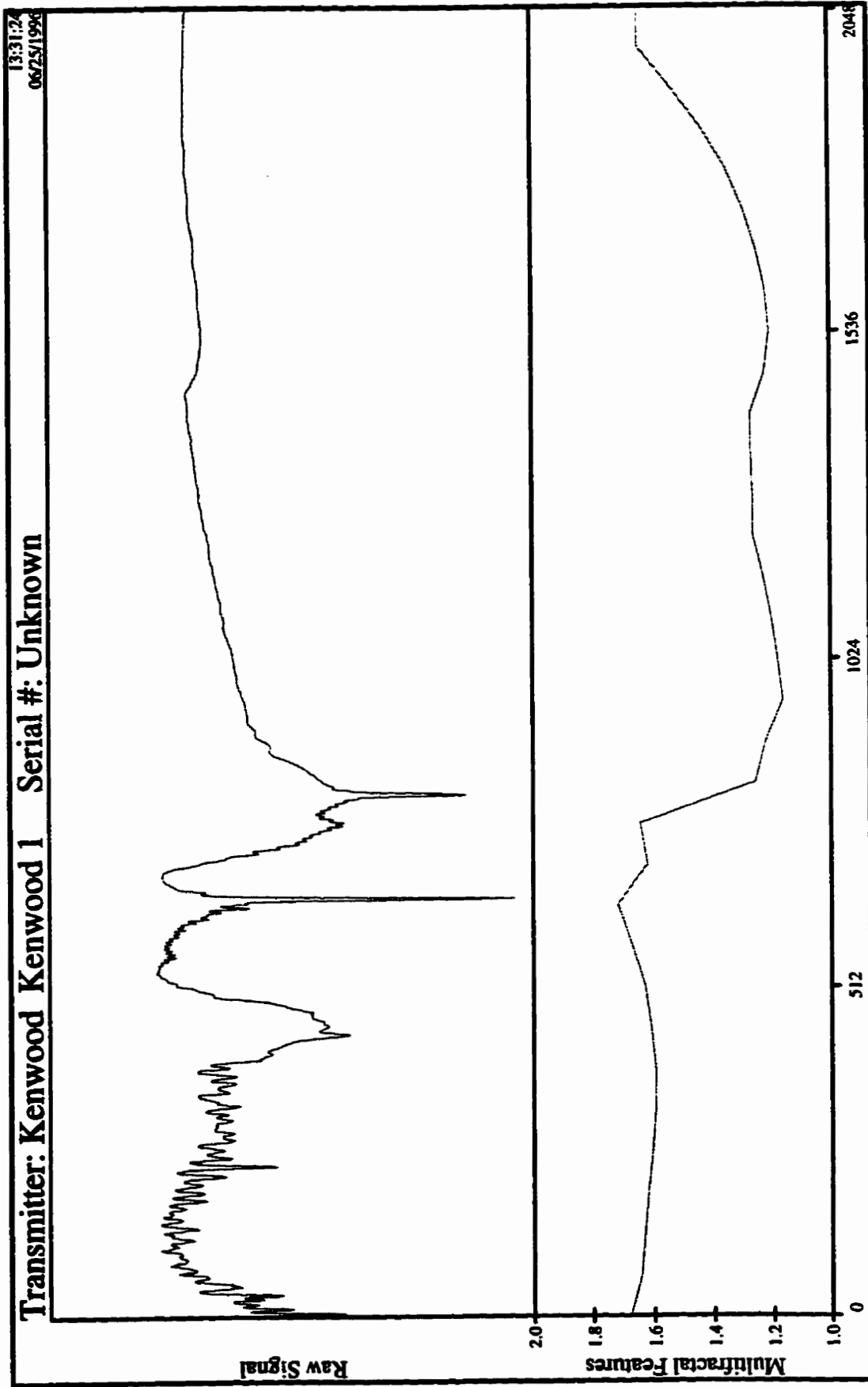
Feature Extraction, Class 2



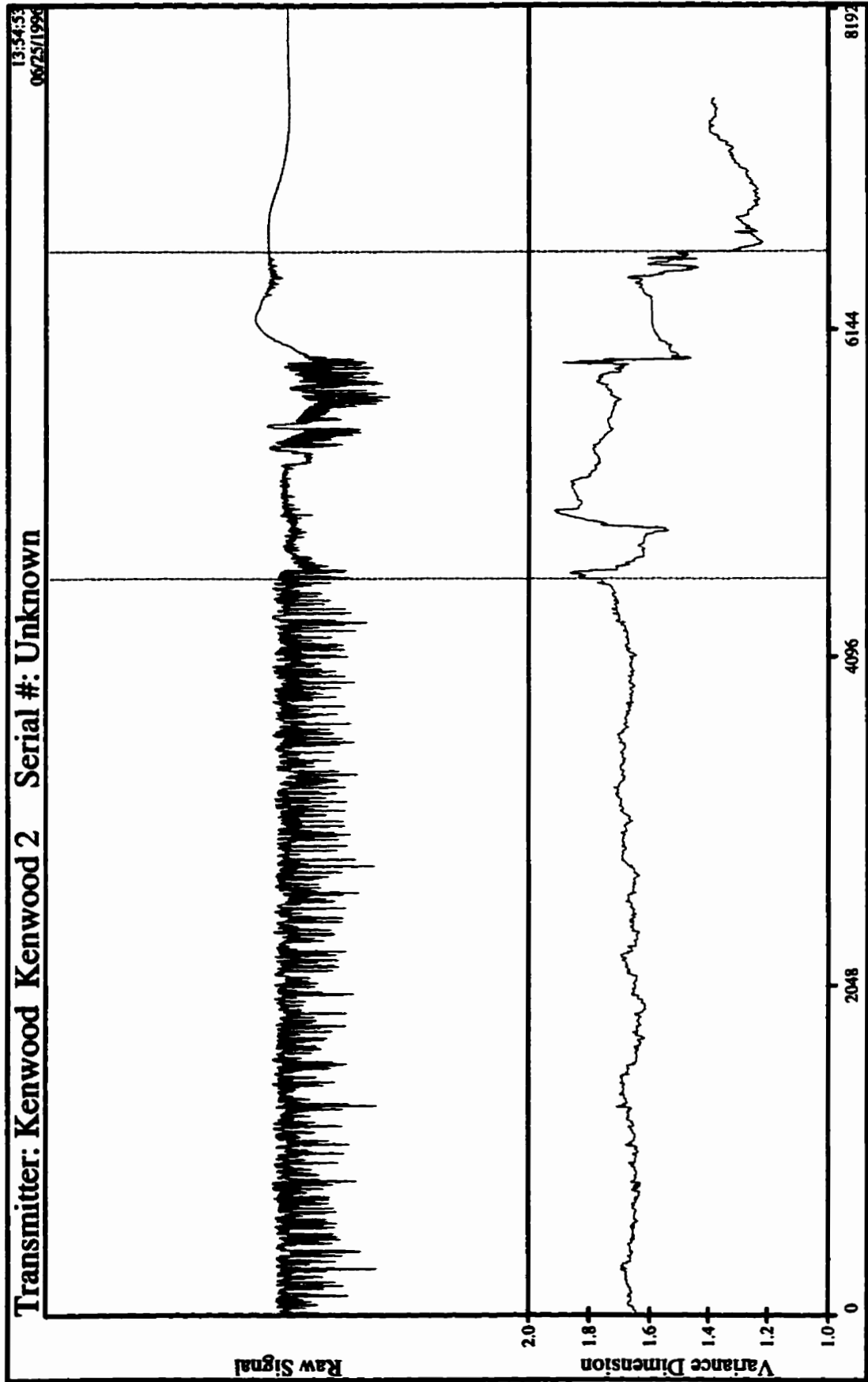
Segmentation, Class 3



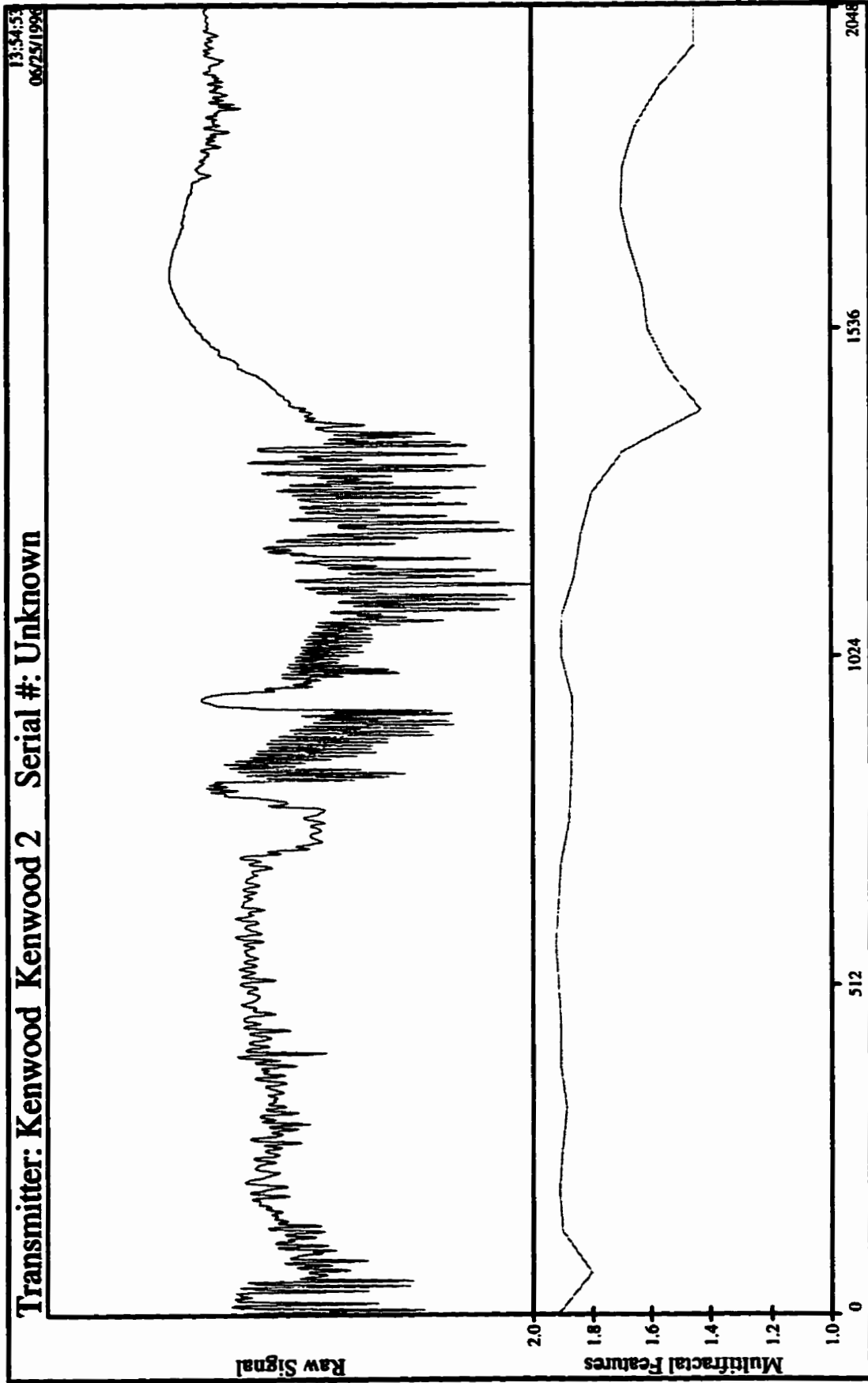
Feature Extraction, Class 3



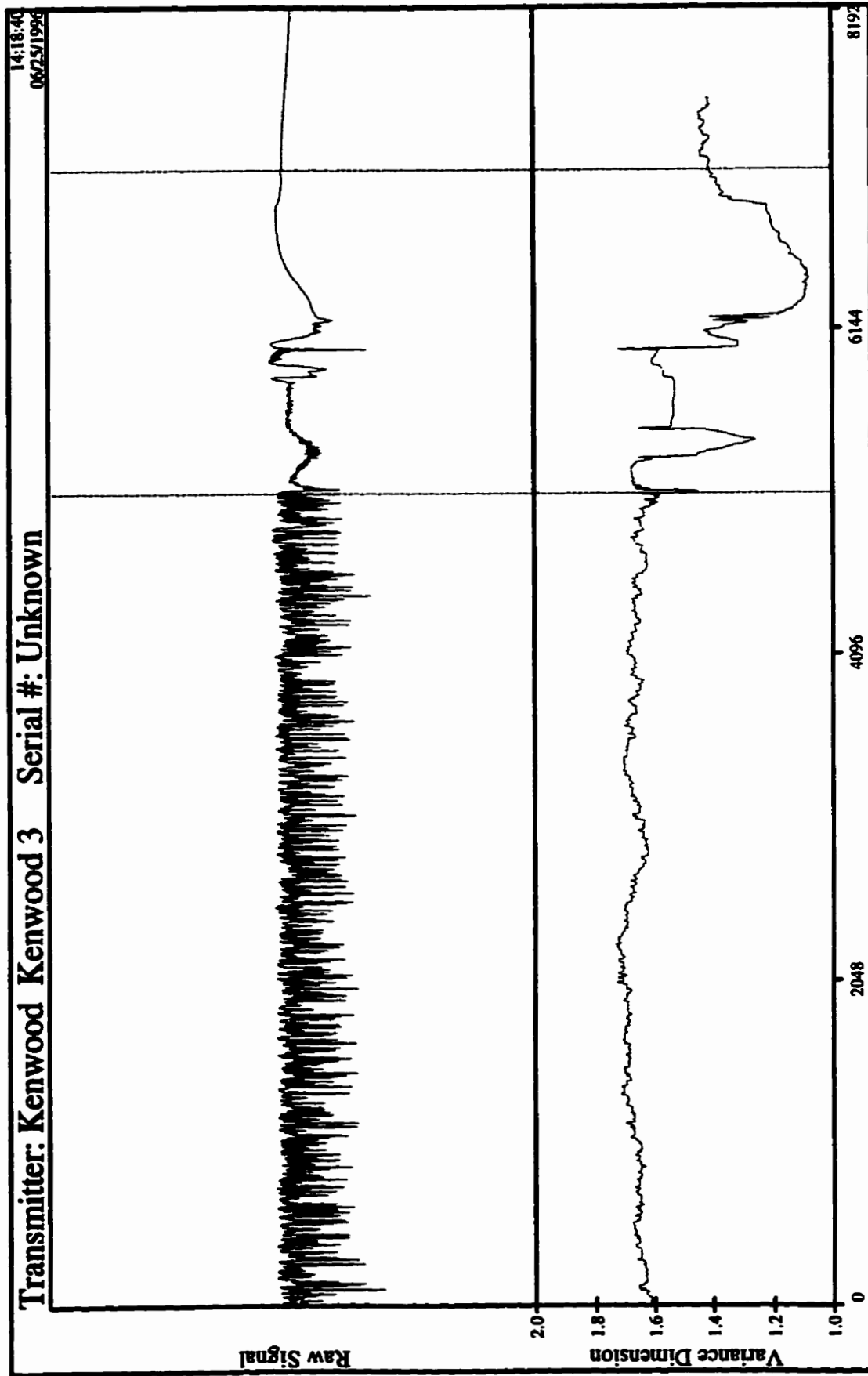
Segmentation, Class 4



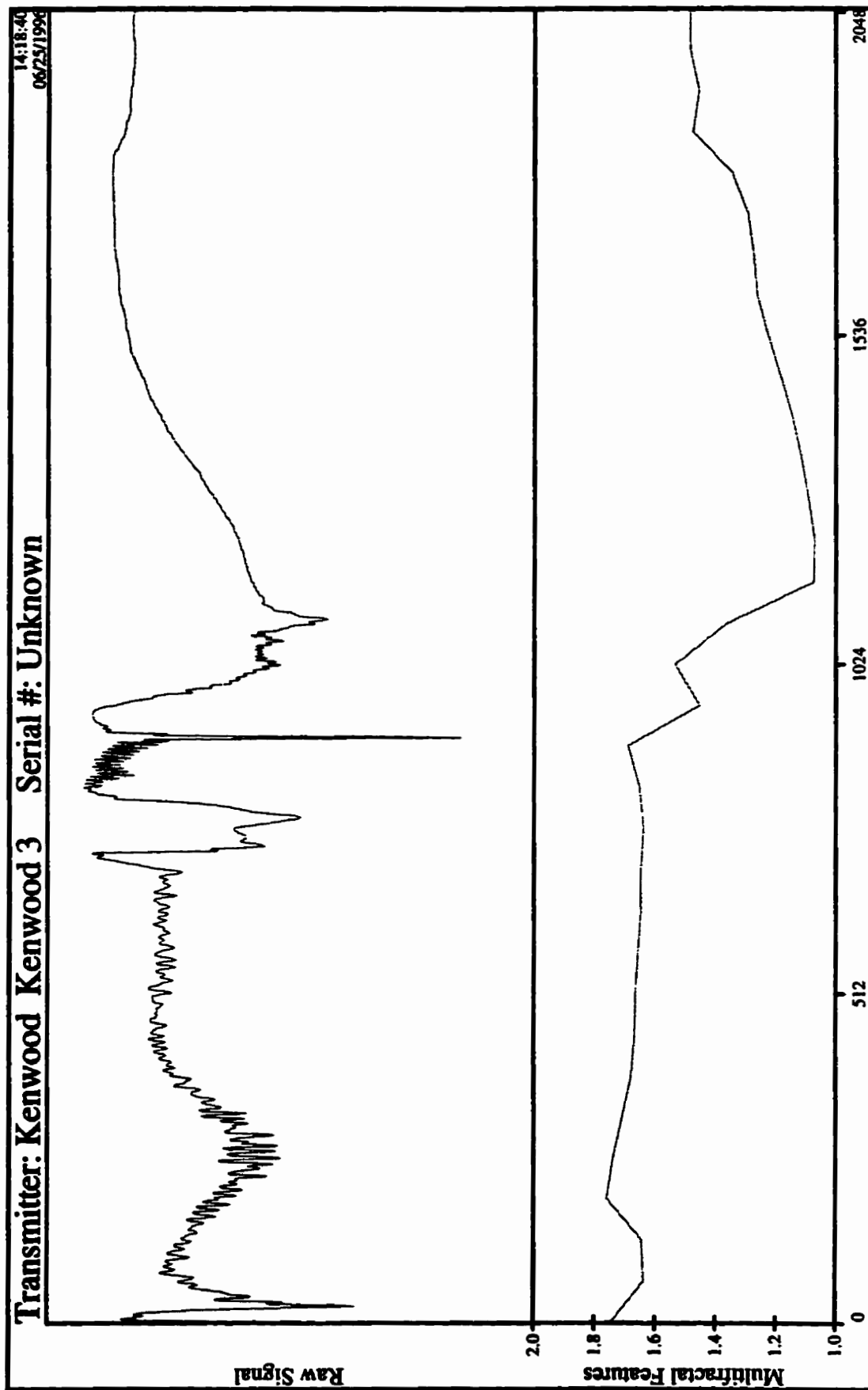
Feature Extraction, Class 4



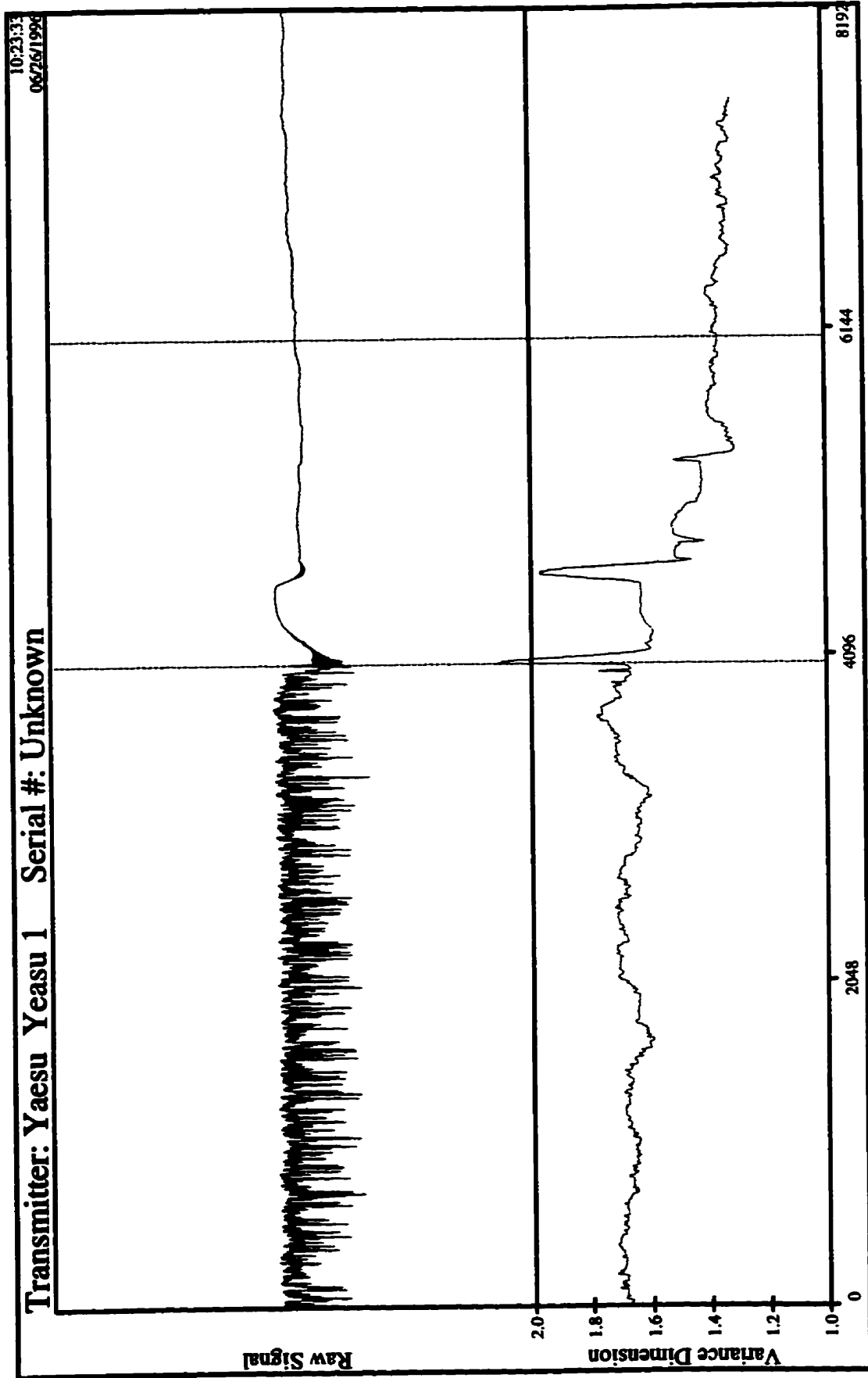
Segmentation, Class 5



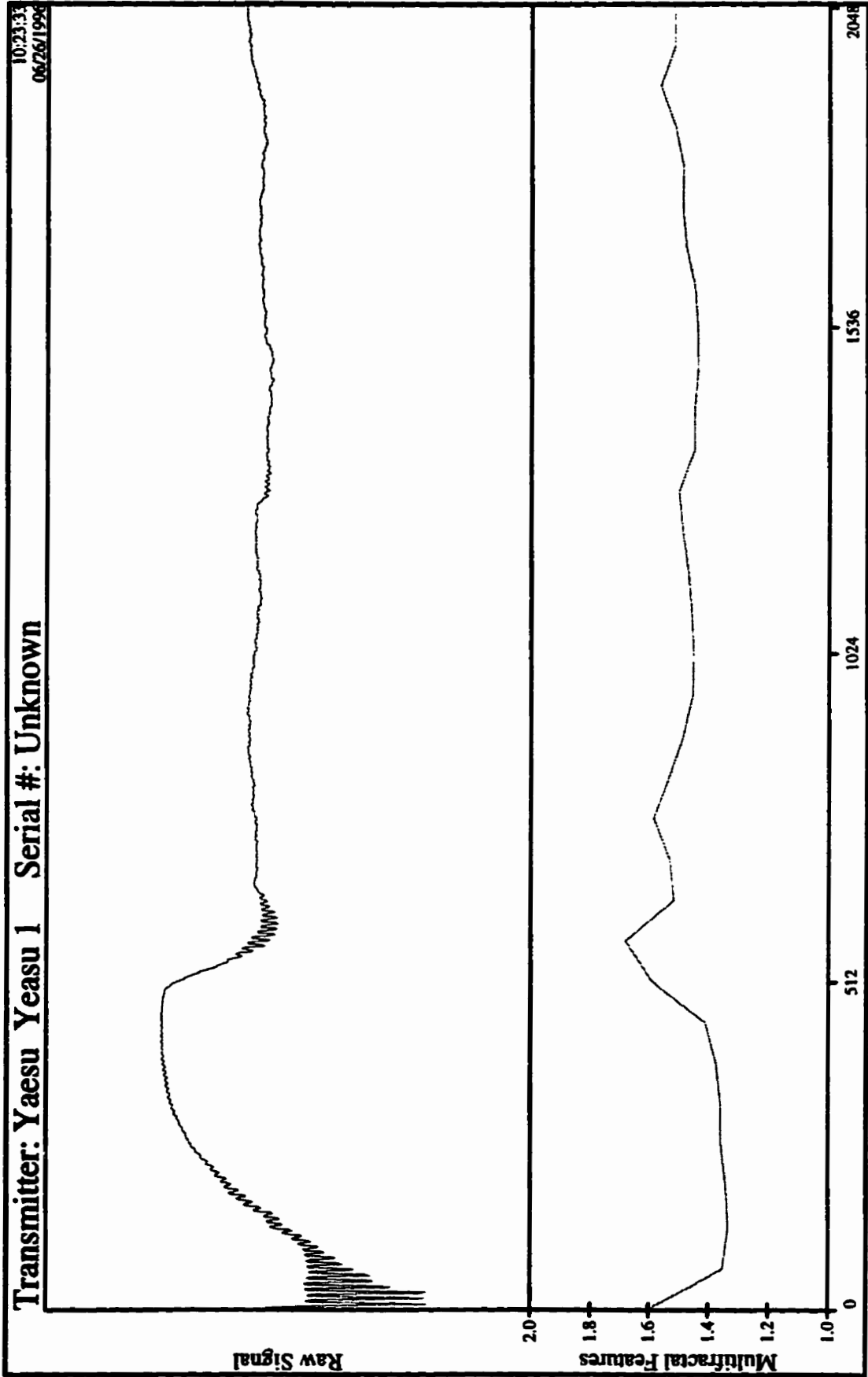
Feature Extraction, Class 5



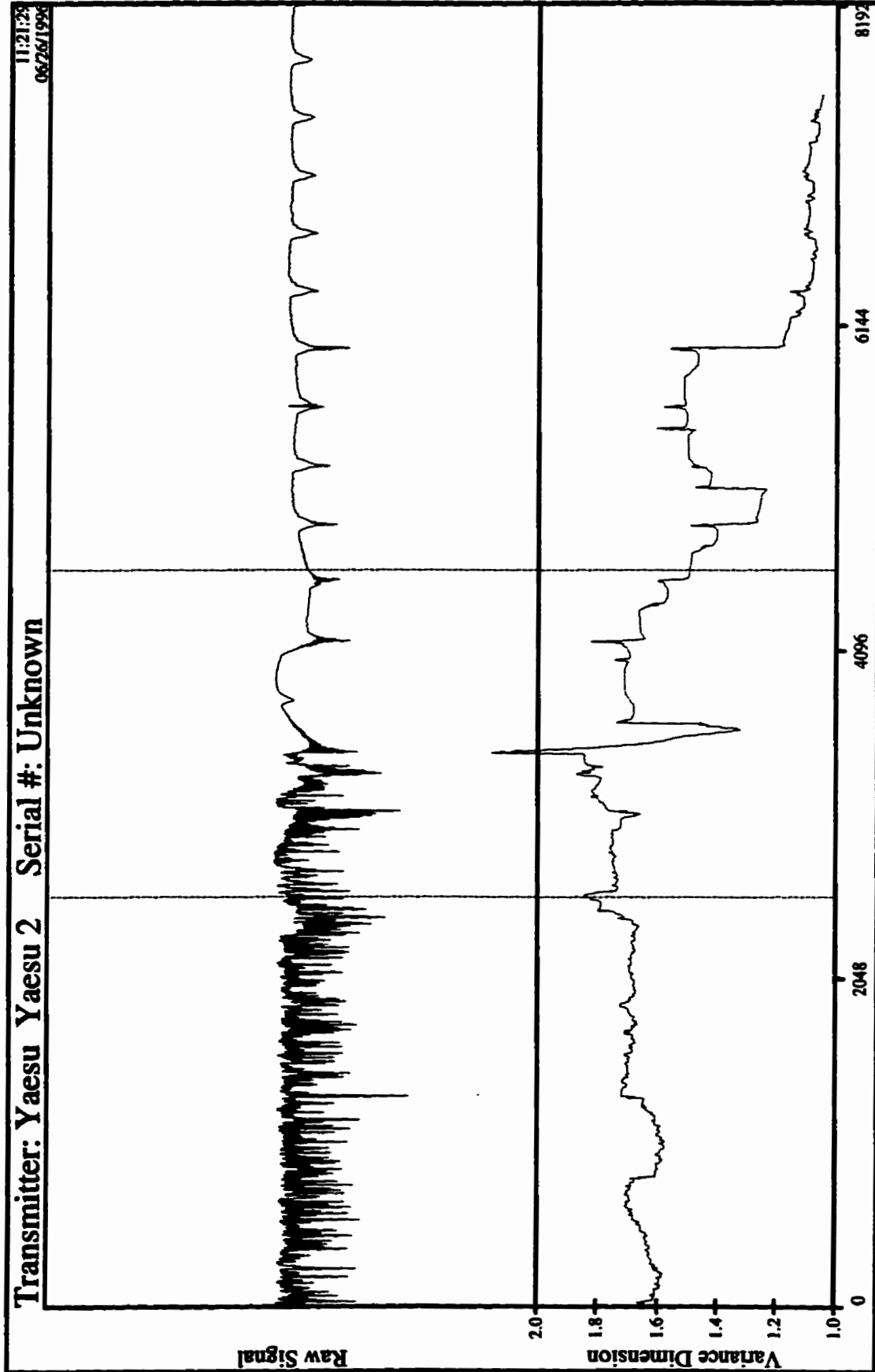
Segmentation, Class 6



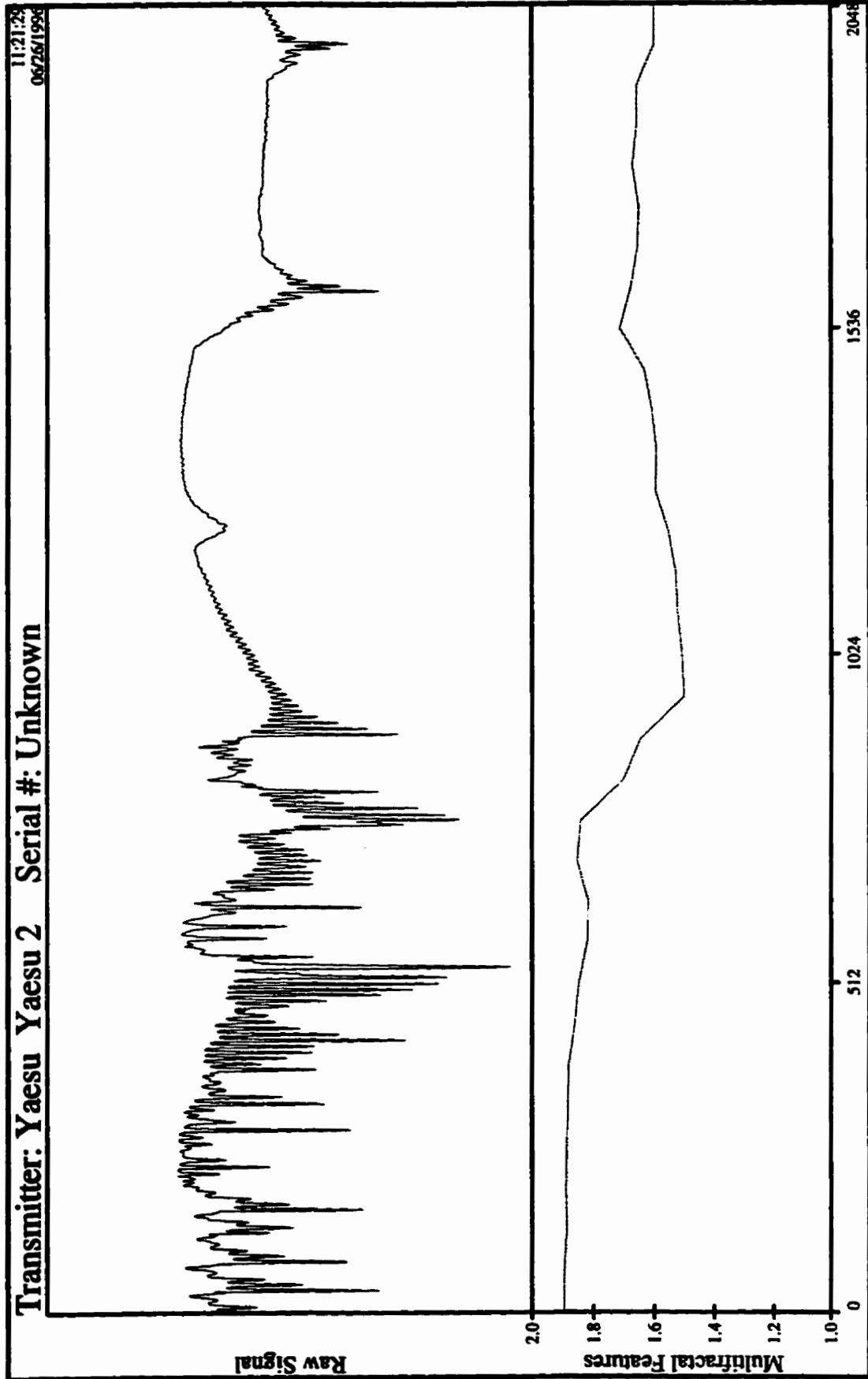
Feature Extraction, Class 6



Segmentation, Class 7



Feature Extraction, Class 7



APPENDIX E

EXPERIMENTAL CLASSIFICATION RESULTS

This appendix contains the batch classification reports for the test sets used in this thesis. Specifically, results are listed for the validation sets associated with the three different training sets used. Also, results are listed for the test of the PNN's rejection ability and for each of the test sets using multimodal segmentation. The reports are arranged as follows:

1.	Validation Set # 1 (First 20 Transients)	E-2
2.	Validation Set # 2 (Random Selection of 10)	E-7
3.	Validation Set # 3 (Random Selection of 30)	E-14
4.	Validation Set #3 with Extended Training	E-18
5.	Test For PNN's Rejection Ability	E-22
6.	Validation Set # 1 with Multimodal Segmentation	E-27
7.	Validation Set # 2 with Multimodal Segmentation	E-32
8.	Validation Set # 3 with Multimodal Segmentation	E-39

VALIDATION SET # 1 (FIRST 20 TRANSIENTS)

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
1	d:\Tran\force1\26fwo415.raw	0	0	97.5455	6.292e-002
2	d:\Tran\force1\26fwo422.raw	0	0	96.0108	5.478e-002
3	d:\Tran\force1\26fwo434.raw	0	0	100.000	2.234e-005
4	d:\Tran\force1\26fwo440.raw	0	0	97.3064	5.795e-002
5	d:\Tran\force1\26fwo443.raw	0	0	97.0582	5.882e-002
6	d:\Tran\force1\26fwo445.raw	0	0	98.9565	6.550e-002
7	d:\Tran\force1\26fwo451.raw	0	0	100.000	2.268e-004
8	d:\Tran\force1\26fwo453.raw	0	1	59.5014	1.093e-005
9	d:\Tran\force1\26fwo455.raw	0	0	97.4368	3.134e-002
10	d:\Tran\force1\26fwo461.raw	0	0	98.3771	4.476e-002
11	d:\Tran\force1\26fwo463.raw	0	0	100.000	1.889e-003
12	d:\Tran\force1\26fwo465.raw	0	0	100.000	6.553e-004
13	d:\Tran\force1\26fwo475.raw	0	0	100.000	1.474e-006
14	d:\Tran\force1\26fwo481.raw	0	0	99.6387	4.149e-002
15	d:\Tran\force1\26fwo484.raw	0	0	100.000	2.488e-004
16	d:\Tran\force1\26fwo492.raw	0	0	100.000	1.393e-004
17	d:\Tran\force1\26fwo494.raw	0	0	99.9906	1.561e-003
18	d:\Tran\force1\26fwo500.raw	0	0	100.000	2.782e-004
19	d:\Tran\force1\26fwo502.raw	0	0	99.4314	1.816e-002
20	d:\Tran\force1\26fwo504.raw	0	0	98.8113	3.556e-002
21	d:\Tran\force1\26fwo513.raw	0	0	99.1415	5.873e-002
22	d:\Tran\force1\26fwo515.raw	0	6	99.9819	1.173e-019
23	d:\Tran\force1\26fwo521.raw	0	0	98.9977	2.659e-002
24	d:\Tran\force1\26fwo524.raw	0	0	98.0198	4.928e-002
25	d:\Tran\force1\26fwo530.raw	0	0	100.000	2.479e-008
26	d:\Tran\force1\26fwo532.raw	0	0	98.5447	4.689e-002
27	d:\Tran\force1\26fwo543.raw	0	0	99.9082	7.096e-003
28	d:\Tran\force1\26fwo545.raw	0	6	84.3601	4.816e-019
29	d:\Tran\force1\26fwo552.raw	0	0	100.000	8.988e-004
30	d:\Tran\force1\26fwo554.raw	0	0	97.4367	4.223e-002
31	d:\Tran\force2\26fwq123.raw	1	1	93.7816	3.649e-002
32	d:\Tran\force2\26fwq125.raw	1	2	54.5802	3.042e-004
33	d:\Tran\force2\26fwq132.raw	1	1	70.8215	2.720e-006
34	d:\Tran\force2\26fwq134.raw	1	1	97.1558	3.014e-002
35	d:\Tran\force2\26fwq142.raw	1	1	50.4998	1.472e-005
36	d:\Tran\force2\26fwq144.raw	1	1	77.9331	2.684e-002
37	d:\Tran\force2\26fwq150.raw	1	1	96.9706	9.776e-003
38	d:\Tran\force2\26fwq152.raw	1	1	98.4948	2.084e-002
39	d:\Tran\force2\26fwq154.raw	1	1	98.6893	1.768e-002
40	d:\Tran\force2\26fwq160.raw	1	1	96.4970	4.354e-006
41	d:\Tran\force2\26fwq162.raw	1	1	90.0572	5.163e-003
42	d:\Tran\force2\26fwq173.raw	1	1	88.2648	1.889e-003
43	d:\Tran\force2\26fwq182.raw	1	1	87.3053	1.561e-003
44	d:\Tran\force2\26fwq184.raw	1	1	97.3213	3.034e-002
45	d:\Tran\force2\26fwq193.raw	1	1	98.9907	4.891e-002
46	d:\Tran\force2\26fwq202.raw	1	1	60.2620	3.929e-006
47	d:\Tran\force2\26fwq230.raw	1	1	65.6436	3.219e-002
48	d:\Tran\force2\26fwq235.raw	1	1	87.8696	1.122e-003
49	d:\Tran\force2\26fwq243.raw	1	1	95.4198	6.140e-002
50	d:\Tran\force2\26fwq253.raw	1	1	99.3206	1.282e-003

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
51	d:\Tran\force2\26fwq255.raw	1	1	98.4794	2.501e-002
52	d:\Tran\force2\26fwq262.raw	1	1	75.9941	4.882e-004
53	d:\Tran\force2\26fwq264.raw	1	1	80.0492	5.125e-004
54	d:\Tran\force2\26fwq265.raw	1	1	99.7436	2.763e-003
55	d:\Tran\force2\26fwq271.raw	1	1	94.3652	8.334e-003
56	d:\Tran\force2\26fwq273.raw	1	1	76.9584	2.017e-003
57	d:\Tran\force2\26fwq284.raw	1	1	89.1198	4.653e-002
58	d:\Tran\force2\26fwq290.raw	1	1	95.1700	1.192e-003
59	d:\Tran\force2\26fwq314.raw	1	1	75.6900	4.478e-003
60	d:\Tran\force2\26fwq322.raw	1	0	99.9396	9.284e-007
61	d:\Tran\force2\26fwq324.raw	1	1	66.0060	4.250e-006
62	d:\Tran\force2\26fwq330.raw	1	1	97.7851	6.145e-005
63	d:\Tran\force3\27fwi291.raw	2	2	99.8735	4.155e-002
64	d:\Tran\force3\27fwi311.raw	2	2	97.9403	3.605e-002
65	d:\Tran\force3\27fwi313.raw	2	2	98.3528	1.354e-002
66	d:\Tran\force3\27fwi315.raw	2	2	99.8733	2.843e-002
67	d:\Tran\force3\27fwi321.raw	2	2	55.0667	6.466e-003
68	d:\Tran\force3\27fwi324.raw	2	2	97.5357	3.653e-002
69	d:\Tran\force3\27fwi330.raw	2	2	99.6683	7.879e-003
70	d:\Tran\force3\27fwi332.raw	2	2	70.3202	4.262e-002
71	d:\Tran\force3\27fwi333.raw	2	2	99.4435	4.903e-003
72	d:\Tran\force3\27fwi335.raw	2	2	99.8922	2.909e-002
73	d:\Tran\force3\27fwi341.raw	2	2	97.9207	4.100e-003
74	d:\Tran\force3\27fwi344.raw	2	2	51.8769	2.094e-002
75	d:\Tran\force3\27fwi353.raw	2	2	83.7075	4.204e-002
76	d:\Tran\force3\27fwi355.raw	2	1	58.1257	2.451e-002
77	d:\Tran\force3\27fwi361.raw	2	2	94.7398	6.466e-002
78	d:\Tran\force3\27fwi381.raw	2	2	80.7281	1.249e-004
79	d:\Tran\force3\27fwi383.raw	2	2	98.8289	1.424e-001
80	d:\Tran\force3\27fwi394.raw	2	1	88.6832	2.825e-004
81	d:\Tran\force3\27fwi423.raw	2	2	86.4589	6.577e-003
82	d:\Tran\force3\27fwi425.raw	2	2	97.2984	5.494e-005
83	d:\Tran\force3\27fwi431.raw	2	2	87.0113	4.270e-003
84	d:\Tran\force3\27fwi433.raw	2	2	94.7765	7.152e-004
85	d:\Tran\force3\27fwi444.raw	2	2	99.3521	2.835e-003
86	d:\Tran\force3\27fwi450.raw	2	2	60.1327	1.463e-005
87	d:\Tran\force3\27fwi452.raw	2	2	99.1982	8.340e-002
88	d:\Tran\force3\27fwi454.raw	2	2	82.3168	3.107e-002
89	d:\Tran\force3\27fwi460.raw	2	2	98.0156	2.418e-002
90	d:\Tran\force3\27fwi461.raw	2	2	90.3091	4.711e-003
91	d:\Tran\force3\27fwi463.raw	2	2	99.7187	4.323e-002
92	d:\Tran\force3\27fwi465.raw	2	2	99.1123	1.175e-001
93	d:\Tran\force3\27fwi471.raw	2	2	89.3181	1.972e-002
94	d:\Tran\force3\27fwi473.raw	2	2	58.4462	1.262e-002
95	d:\Tran\ken1\25fwn380.raw	3	3	99.8513	7.770e-011
96	d:\Tran\ken1\25fwn382.raw	3	3	100.000	5.930e-007
97	d:\Tran\ken1\25fwn384.raw	3	3	100.000	8.797e-010
98	d:\Tran\ken1\25fwn392.raw	3	3	100.000	4.027e-009
99	d:\Tran\ken1\25fwn393.raw	3	3	100.000	1.583e-006
100	d:\Tran\ken1\25fwn395.raw	3	3	100.000	2.155e-004
101	d:\Tran\ken1\25fwn401.raw	3	3	100.000	8.418e-010
102	d:\Tran\ken1\25fwn402.raw	3	3	100.000	1.602e-012
103	d:\Tran\ken1\25fwn404.raw	3	3	100.000	6.241e-011
104	d:\Tran\ken1\25fwn405.raw	3	3	99.9944	3.620e-008

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
105	d:\Tran\ken1\25fwn411.raw	3	3	100.000	3.301e-008
106	d:\Tran\ken1\25fwn413.raw	3	3	100.000	2.611e-005
107	d:\Tran\ken1\25fwn414.raw	3	3	100.000	7.295e-008
108	d:\Tran\ken1\25fwn415.raw	3	3	100.000	1.795e-012
109	d:\Tran\ken1\25fwn421.raw	3	3	99.9998	7.141e-007
110	d:\Tran\ken1\25fwn423.raw	3	3	100.000	7.377e-004
111	d:\Tran\ken1\25fwn424.raw	3	3	100.000	5.336e-007
112	d:\Tran\ken1\25fwn430.raw	3	5	100.000	1.611e-009
113	d:\Tran\ken1\25fwn432.raw	3	3	100.000	1.218e-004
114	d:\Tran\ken1\25fwn433.raw	3	3	100.000	6.361e-006
115	d:\Tran\ken1\25fwn435.raw	3	3	100.000	1.059e-007
116	d:\Tran\ken1\25fwn441.raw	3	3	100.000	1.569e-006
117	d:\Tran\ken1\25fwn443.raw	3	3	100.000	9.368e-005
118	d:\Tran\ken1\25fwn445.raw	3	3	100.000	6.126e-005
119	d:\Tran\ken1\25fwn450.raw	3	3	100.000	2.283e-003
120	d:\Tran\ken1\25fwn452.raw	3	3	100.000	3.805e-005
121	d:\Tran\ken1\25fwn453.raw	3	3	83.6710	1.479e-018
122	d:\Tran\ken1\25fwn455.raw	3	3	100.000	4.172e-006
123	d:\Tran\ken1\25fwn461.raw	3	3	100.000	8.907e-006
124	d:\Tran\ken1\25fwn462.raw	3	3	83.2510	1.005e-015
125	d:\Tran\ken2\25fwn583.raw	4	4	100.000	2.923e-003
126	d:\Tran\ken2\25fwn584.raw	4	4	100.000	4.466e-002
127	d:\Tran\ken2\25fwn593.raw	4	4	100.000	3.323e-003
128	d:\Tran\ken2\25fwn594.raw	4	4	100.000	1.611e-003
129	d:\Tran\ken2\25fwo000.raw	4	4	100.000	2.034e-002
130	d:\Tran\ken2\25fwo002.raw	4	4	100.000	5.841e-002
131	d:\Tran\ken2\25fwo003.raw	4	4	100.000	5.336e-002
132	d:\Tran\ken2\25fwo005.raw	4	4	100.000	3.198e-003
133	d:\Tran\ken2\25fwo010.raw	4	4	100.000	2.096e-007
134	d:\Tran\ken2\25fwo015.raw	4	4	100.000	3.716e-006
135	d:\Tran\ken2\25fwo021.raw	4	4	100.000	8.465e-002
136	d:\Tran\ken2\25fwo022.raw	4	4	100.000	1.048e-006
137	d:\Tran\ken2\25fwo024.raw	4	4	100.000	4.451e-002
138	d:\Tran\ken2\25fwo030.raw	4	4	100.000	2.461e-003
139	d:\Tran\ken2\25fwo031.raw	4	4	100.000	5.747e-016
140	d:\Tran\ken2\25fwo033.raw	4	4	100.000	8.419e-002
141	d:\Tran\ken2\25fwo034.raw	4	4	100.000	1.700e-004
142	d:\Tran\ken2\25fwo042.raw	4	4	100.000	3.734e-014
143	d:\Tran\ken2\25fwo044.raw	4	4	100.000	1.036e-003
144	d:\Tran\ken2\25fwo045.raw	4	4	100.000	5.397e-002
145	d:\Tran\ken2\25fwo054.raw	4	4	100.000	1.550e-003
146	d:\Tran\ken2\25fwo062.raw	4	4	100.000	4.940e-002
147	d:\Tran\ken2\25fwo064.raw	4	4	100.000	3.243e-002
148	d:\Tran\ken2\25fwo065.raw	4	4	100.000	6.051e-003
149	d:\Tran\ken2\25fwo074.raw	4	4	100.000	4.753e-003
150	d:\Tran\ken2\25fwo075.raw	4	4	100.000	2.018e-003
151	d:\Tran\ken2\25fwo080.raw	4	4	100.000	3.155e-004
152	d:\Tran\ken2\25fwo082.raw	4	4	100.000	1.875e-002
153	d:\Tran\ken2\25fwo083.raw	4	4	100.000	3.110e-002
154	d:\Tran\ken2\25fwo085.raw	4	4	100.000	2.164e-003
155	d:\Tran\ken2\25fwo090.raw	4	4	100.000	4.478e-002
156	d:\Tran\ken3\25fwo225.raw	5	5	100.000	3.444e-005
157	d:\Tran\ken3\25fwo231.raw	5	5	100.000	5.064e-006
158	d:\Tran\ken3\25fwo232.raw	5	5	100.000	6.202e-011

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
159	d:\Tran\ken3\25fwo234.raw	5	5	100.000	2.145e-003
160	d:\Tran\ken3\25fwo235.raw	5	5	100.000	2.562e-015
161	d:\Tran\ken3\25fwo241.raw	5	5	100.000	5.362e-005
162	d:\Tran\ken3\25fwo251.raw	5	3	100.000	3.044e-021
163	d:\Tran\ken3\25fwo253.raw	5	5	100.000	1.270e-006
164	d:\Tran\ken3\25fwo260.raw	5	5	100.000	1.295e-002
165	d:\Tran\ken3\25fwo265.raw	5	5	100.000	1.328e-004
166	d:\Tran\ken3\25fwo271.raw	5	5	100.000	4.189e-004
167	d:\Tran\ken3\25fwo282.raw	5	5	100.000	1.077e-004
168	d:\Tran\ken3\25fwo283.raw	5	5	100.000	3.345e-005
169	d:\Tran\ken3\25fwo301.raw	5	5	100.000	9.946e-004
170	d:\Tran\ken3\25fwo303.raw	5	5	100.000	8.078e-004
171	d:\Tran\ken3\25fwo305.raw	5	3	100.000	2.187e-008
172	d:\Tran\ken3\25fwo312.raw	5	3	99.9941	2.259e-008
173	d:\Tran\ken3\25fwo314.raw	5	5	100.000	1.376e-004
174	d:\Tran\ken3\25fwo315.raw	5	5	100.000	1.566e-003
175	d:\Tran\ken3\25fwo321.raw	5	5	100.000	2.248e-004
176	d:\Tran\ken3\25fwo325.raw	5	5	99.9993	1.098e-005
177	d:\Tran\ken3\25fwo332.raw	5	5	99.9994	2.011e-005
178	d:\Tran\ken3\25fwo334.raw	5	5	100.000	3.038e-005
179	d:\Tran\ken3\25fwo335.raw	5	5	100.000	3.627e-012
180	d:\Tran\ken3\25fwo341.raw	5	5	100.000	7.055e-004
181	d:\Tran\ken3\25fwo342.raw	5	5	100.000	2.429e-003
182	d:\Tran\ken3\25fwo344.raw	5	5	100.000	1.650e-004
183	d:\Tran\ken3\25fwo352.raw	5	5	99.0165	1.188e-009
184	d:\Tran\ken3\25fwo354.raw	5	5	99.9998	2.664e-014
185	d:\Tran\ken3\25fwp063.raw	5	5	100.000	1.073e-015
186	d:\Tran\ken3\25fwp065.raw	5	5	100.000	3.235e-007
187	d:\Tran\ken3\25fwp075.raw	5	5	99.9971	3.413e-008
188	d:\Tran\Yaesu1\26fwk312.raw	6	6	99.9982	3.050e-004
189	d:\Tran\Yaesu1\26fwk314.raw	6	6	100.000	4.435e-005
190	d:\Tran\Yaesu1\26fwk332.raw	6	6	100.000	1.612e-004
191	d:\Tran\Yaesu1\26fwk334.raw	6	6	100.000	3.124e-004
192	d:\Tran\Yaesu1\26fwk335.raw	6	6	100.000	1.441e-005
193	d:\Tran\Yaesu1\26fwk341.raw	6	6	100.000	6.435e-006
194	d:\Tran\Yaesu1\26fwk343.raw	6	6	100.000	2.653e-003
195	d:\Tran\Yaesu1\26fwk345.raw	6	6	99.9999	1.844e-002
196	d:\Tran\Yaesu1\26fwk355.raw	6	6	100.000	7.836e-004
197	d:\Tran\Yaesu1\26fwk361.raw	6	6	100.000	5.492e-004
198	d:\Tran\Yaesu1\26fwk363.raw	6	6	99.9998	8.231e-008
199	d:\Tran\Yaesu1\26fwk365.raw	6	6	100.000	2.017e-006
200	d:\Tran\Yaesu1\26fwk370.raw	6	6	100.000	7.204e-004
201	d:\Tran\Yaesu1\26fwk372.raw	6	6	99.9997	7.638e-003
202	d:\Tran\Yaesu1\26fwk435.raw	6	6	100.000	3.058e-003
203	d:\Tran\Yaesu1\26fwk450.raw	6	6	93.1387	1.700e-011
204	d:\Tran\Yaesu1\26fwk452.raw	6	6	100.000	4.636e-003
205	d:\Tran\Yaesu1\26fwk453.raw	6	6	100.000	2.564e-003
206	d:\Tran\Yaesu1\26fwk455.raw	6	6	100.000	6.301e-003
207	d:\Tran\Yaesu1\26fwk483.raw	6	6	100.000	3.959e-004
208	d:\Tran\Yaesu1\26fwk502.raw	6	6	100.000	2.938e-004
209	d:\Tran\Yaesu1\26fwk505.raw	6	6	100.000	4.010e-006
210	d:\Tran\Yaesu1\26fwk520.raw	6	6	100.000	2.860e-004
211	d:\Tran\Yaesu1\26fwk522.raw	6	6	100.000	7.374e-004
212	d:\Tran\Yaesu1\26fwk524.raw	6	6	100.000	1.314e-006

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
213	d:\Tran\Yaesu1\26fwk532.raw	6	6	100.000	2.075e-002
214	d:\Tran\Yaesu1\26fwk541.raw	6	6	100.000	7.823e-003
215	d:\Tran\Yaesu1\26fwk544.raw	6	6	100.000	3.016e-005
216	d:\Tran\Yaesu1\26fwk545.raw	6	6	99.9997	3.523e-003
217	d:\Tran\Yaesu1\26fwk551.raw	6	6	100.000	1.992e-003
218	d:\Tran\Yaesu1\26fwk564.raw	6	6	100.000	4.860e-003
219	d:\Tran\Yaesu1\26fwk570.raw	6	6	100.000	1.707e-006
220	d:\Tran\Yaesu1\26fwk571.raw	6	6	99.9946	1.253e-003
221	d:\Tran\Yaesu1\26fwk573.raw	6	6	100.000	9.270e-004
222	d:\Tran\Yaesu1\26fwk575.raw	6	6	100.000	2.199e-007
223	d:\Tran\Yaesu1\26fwk585.raw	6	6	100.000	2.704e-002
224	d:\Tran\Yaesu1\26fwl002.raw	6	6	100.000	3.789e-006
225	d:\Tran\Yaesu1\26fwl003.raw	6	6	100.000	1.748e-005
226	d:\Tran\Yaesu2\26fwl224.raw	7	6	100.000	1.964e-010
227	d:\Tran\Yaesu2\26fwl225.raw	7	7	100.000	2.642e-011
228	d:\Tran\Yaesu2\26fwl231.raw	7	7	100.000	5.377e-011
229	d:\Tran\Yaesu2\26fwl233.raw	7	7	95.9343	1.008e-010
230	d:\Tran\Yaesu2\26fwl235.raw	7	7	100.000	2.508e-005
231	d:\Tran\Yaesu2\26fwl241.raw	7	7	100.000	6.866e-004
232	d:\Tran\Yaesu2\26fwl243.raw	7	7	100.000	6.224e-004
233	d:\Tran\Yaesu2\26fwl245.raw	7	7	100.000	3.684e-005
234	d:\Tran\Yaesu2\26fwl251.raw	7	7	100.000	2.807e-006
235	d:\Tran\Yaesu2\26fwl254.raw	7	7	99.9998	2.338e-003
236	d:\Tran\Yaesu2\26fwl260.raw	7	7	100.000	1.216e-005
237	d:\Tran\Yaesu2\26fwl262.raw	7	7	100.000	3.769e-005
238	d:\Tran\Yaesu2\26fwl264.raw	7	7	100.000	6.236e-008
239	d:\Tran\Yaesu2\26fwl270.raw	7	7	100.000	9.548e-006
240	d:\Tran\Yaesu2\26fwl272.raw	7	7	100.000	1.508e-005
241	d:\Tran\Yaesu2\26fwl274.raw	7	7	100.000	2.607e-004
242	d:\Tran\Yaesu2\26fwl280.raw	7	7	100.000	8.913e-006
243	d:\Tran\Yaesu2\26fwl281.raw	7	7	99.9998	1.744e-013
244	d:\Tran\Yaesu2\26fwl283.raw	7	7	100.000	2.440e-006
245	d:\Tran\Yaesu2\26fwl285.raw	7	7	100.000	4.551e-004
246	d:\Tran\Yaesu2\26fwl293.raw	7	7	100.000	9.886e-004
247	d:\Tran\Yaesu2\26fwl300.raw	7	7	100.000	1.443e-005
248	d:\Tran\Yaesu2\26fwl304.raw	7	7	100.000	9.845e-011
249	d:\Tran\Yaesu2\26fwl310.raw	7	7	100.000	2.479e-003
250	d:\Tran\Yaesu2\26fwl312.raw	7	7	100.000	9.227e-007
251	d:\Tran\Yaesu2\26fwl314.raw	7	7	100.000	1.391e-003
252	d:\Tran\Yaesu2\26fwl320.raw	7	7	99.9998	8.977e-007
253	d:\Tran\Yaesu2\26fwl322.raw	7	7	100.000	5.329e-009
254	d:\Tran\Yaesu2\26fwl330.raw	7	7	99.9999	1.915e-007
255	d:\Tran\Yaesu2\26fwl332.raw	7	7	100.000	4.046e-005

Number of Correct Classifications: 243/255.

VALIDATION SET # 2 (RANDOM SELECTION OF 10)

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
1	c:\transients\force1\26fwo322.raw	0	0	99.5993	4.106e-002
2	c:\transients\force1\26fwo325.raw	0	0	99.9856	1.248e-002
3	c:\transients\force1\26fwo332.raw	0	0	97.2578	1.105e-001
4	c:\transients\force1\26fwo341.raw	0	0	99.8806	1.492e-002
5	c:\transients\force1\26fwo363.raw	0	0	99.4528	1.506e-002
6	c:\transients\force1\26fwo365.raw	0	0	62.3419	4.719e-003
7	c:\transients\force1\26fwo374.raw	0	0	99.2105	3.067e-002
8	c:\transients\force1\26fwo380.raw	0	0	99.2496	6.329e-002
9	c:\transients\force1\26fwo393.raw	0	0	100.000	1.075e-003
10	c:\transients\force1\26fwo395.raw	0	0	100.000	6.655e-006
11	c:\transients\force1\26fwo401.raw	0	0	99.9995	2.247e-003
12	c:\transients\force1\26fwo403.raw	0	0	99.7854	2.874e-013
13	c:\transients\force1\26fwo415.raw	0	0	97.1734	1.144e-001
14	c:\transients\force1\26fwo422.raw	0	0	95.8953	1.169e-001
15	c:\transients\force1\26fwo434.raw	0	0	100.000	1.008e-006
16	c:\transients\force1\26fwo440.raw	0	0	96.5931	1.107e-001
17	c:\transients\force1\26fwo451.raw	0	0	100.000	2.498e-002
18	c:\transients\force1\26fwo453.raw	0	0	96.0408	1.560e-002
19	c:\transients\force1\26fwo455.raw	0	0	97.7187	5.350e-002
20	c:\transients\force1\26fwo461.raw	0	0	95.5406	6.647e-002
21	c:\transients\force1\26fwo475.raw	0	0	99.9930	1.365e-003
22	c:\transients\force1\26fwo481.raw	0	0	98.4709	6.577e-002
23	c:\transients\force1\26fwo484.raw	0	0	100.000	3.909e-003
24	c:\transients\force1\26fwo492.raw	0	0	99.6364	6.436e-003
25	c:\transients\force1\26fwo502.raw	0	0	99.5622	3.829e-002
26	c:\transients\force1\26fwo504.raw	0	0	63.6798	8.648e-003
27	c:\transients\force1\26fwo513.raw	0	0	98.3593	1.154e-001
28	c:\transients\force1\26fwo515.raw	0	6	99.9968	7.913e-012
29	c:\transients\force1\26fwo530.raw	0	0	100.000	6.121e-005
30	c:\transients\force1\26fwo532.raw	0	0	92.3155	2.965e-002
31	c:\transients\force1\26fwo543.raw	0	0	98.5232	1.346e-002
32	c:\transients\force1\26fwo545.raw	0	6	99.9892	2.036e-011
33	c:\transients\force1\26fwo362.raw	0	0	99.1118	9.686e-002
34	c:\transients\force1\26fwo390.raw	0	0	100.000	1.715e-005
35	c:\transients\force1\26fwo413.raw	0	0	100.000	2.871e-002
36	c:\transients\force1\26fwo445.raw	0	0	95.1080	5.990e-002
37	c:\transients\force1\26fwo465.raw	0	0	100.000	2.185e-003
38	c:\transients\force1\26fwo500.raw	0	0	100.000	4.181e-003
39	c:\transients\force1\26fwo524.raw	0	0	96.5528	1.102e-001
40	c:\transients\force1\26fwo554.raw	0	0	96.3245	1.058e-001
41	c:\transients\force2\26fwp573.raw	1	1	93.4344	3.230e-003
42	c:\transients\force2\26fwp592.raw	1	1	75.4360	1.823e-003
43	c:\transients\force2\26fwq012.raw	1	1	85.9246	6.608e-003
44	c:\transients\force2\26fwq015.raw	1	1	95.3070	2.947e-002
45	c:\transients\force2\26fwq044.raw	1	1	70.0518	9.695e-003
46	c:\transients\force2\26fwq050.raw	1	1	93.7401	5.660e-002
47	c:\transients\force2\26fwq054.raw	1	2	53.9532	3.792e-004
48	c:\transients\force2\26fwq061.raw	1	2	75.2830	2.171e-004
49	c:\transients\force2\26fwq073.raw	1	1	87.9549	8.193e-003
50	c:\transients\force2\26fwq084.raw	1	1	95.9500	5.008e-003

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
51	c:\transients\force2\26fwq091.raw	1	1	94.5352	3.821e-002
52	c:\transients\force2\26fwq094.raw	1	1	90.3657	6.413e-002
53	c:\transients\force2\26fwq123.raw	1	1	95.6488	7.187e-003
54	c:\transients\force2\26fwq125.raw	1	1	98.6605	1.285e-003
55	c:\transients\force2\26fwq132.raw	1	1	99.8969	3.252e-004
56	c:\transients\force2\26fwq134.raw	1	1	84.6183	1.590e-002
57	c:\transients\force2\26fwq150.raw	1	2	50.4876	8.770e-003
58	c:\transients\force2\26fwq152.raw	1	1	86.3929	1.093e-002
59	c:\transients\force2\26fwq154.raw	1	1	89.9631	4.444e-003
60	c:\transients\force2\26fwq160.raw	1	1	99.5781	4.243e-004
61	c:\transients\force2\26fwq182.raw	1	1	92.9179	2.108e-003
62	c:\transients\force2\26fwq184.raw	1	1	97.4205	6.027e-002
63	c:\transients\force2\26fwq193.raw	1	1	94.4100	1.498e-002
64	c:\transients\force2\26fwq202.raw	1	1	99.9998	2.692e-003
65	c:\transients\force2\26fwq243.raw	1	1	97.3066	3.569e-002
66	c:\transients\force2\26fwq253.raw	1	1	50.9541	7.507e-004
67	c:\transients\force2\26fwq255.raw	1	1	99.3600	5.241e-002
68	c:\transients\force2\26fwq262.raw	1	1	99.9473	1.102e-002
69	c:\transients\force2\26fwq271.raw	1	1	99.5470	7.600e-003
70	c:\transients\force2\26fwq273.raw	1	2	76.0431	9.171e-003
71	c:\transients\force2\26fwq284.raw	1	1	93.4600	3.420e-002
72	c:\transients\force2\26fwq290.raw	1	1	95.6536	1.553e-003
73	c:\transients\force2\26fwq324.raw	1	1	99.6085	4.851e-004
74	c:\transients\force2\26fwq330.raw	1	1	83.5676	3.956e-004
75	c:\transients\force2\26fwq042.raw	1	1	72.2736	4.485e-003
76	c:\transients\force2\26fwq071.raw	1	1	99.9944	3.955e-004
77	c:\transients\force2\26fwq121.raw	1	1	84.4409	2.755e-002
78	c:\transients\force2\26fwq144.raw	1	1	52.8526	5.735e-003
79	c:\transients\force2\26fwq173.raw	1	1	53.7142	4.939e-003
80	c:\transients\force2\26fwq235.raw	1	1	98.4216	4.011e-003
81	c:\transients\force2\26fwq265.raw	1	1	98.2088	2.621e-003
82	c:\transients\force2\26fwq322.raw	1	0	83.9099	2.006e-004
83	c:\transients\force3\27fwi155.raw	2	2	90.8114	6.017e-002
84	c:\transients\force3\27fwi190.raw	2	1	94.4994	2.222e-005
85	c:\transients\force3\27fwi191.raw	2	2	96.6484	4.239e-002
86	c:\transients\force3\27fwi194.raw	2	1	83.1471	1.387e-002
87	c:\transients\force3\27fwi212.raw	2	2	92.1551	1.062e-001
88	c:\transients\force3\27fwi214.raw	2	2	81.9448	8.768e-003
89	c:\transients\force3\27fwi242.raw	2	2	51.2032	5.247e-003
90	c:\transients\force3\27fwi250.raw	2	2	86.0776	3.698e-002
91	c:\transients\force3\27fwi262.raw	2	1	83.8342	1.090e-010
92	c:\transients\force3\27fwi264.raw	2	2	99.6059	4.875e-004
93	c:\transients\force3\27fwi274.raw	2	2	87.5742	9.519e-003
94	c:\transients\force3\27fwi281.raw	2	1	65.2878	1.466e-002
95	c:\transients\force3\27fwi291.raw	2	2	98.6932	8.359e-002
96	c:\transients\force3\27fwi311.raw	2	2	99.4391	1.039e-003
97	c:\transients\force3\27fwi313.raw	2	2	92.0100	1.450e-002
98	c:\transients\force3\27fwi315.raw	2	2	98.3186	4.872e-002
99	c:\transients\force3\27fwi330.raw	2	2	97.8981	1.691e-002
100	c:\transients\force3\27fwi332.raw	2	2	74.3965	1.371e-002
101	c:\transients\force3\27fwi333.raw	2	2	96.5273	9.698e-003
102	c:\transients\force3\27fwi335.raw	2	2	97.8078	1.689e-002
103	c:\transients\force3\27fwi353.raw	2	1	68.8878	9.254e-003
104	c:\transients\force3\27fwi355.raw	2	2	74.5886	2.085e-002

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
105	c:\transients\force3\27fwi361.raw	2	2	95.9898	9.471e-002
106	c:\transients\force3\27fwi381.raw	2	2	98.4778	3.738e-003
107	c:\transients\force3\27fwi423.raw	2	2	69.6740	6.325e-003
108	c:\transients\force3\27fwi425.raw	2	1	98.4867	2.624e-004
109	c:\transients\force3\27fwi431.raw	2	1	96.1711	7.147e-004
110	c:\transients\force3\27fwi433.raw	2	2	77.5299	1.126e-005
111	c:\transients\force3\27fwi452.raw	2	2	99.0158	1.921e-001
112	c:\transients\force3\27fwi454.raw	2	2	93.2207	5.056e-002
113	c:\transients\force3\27fwi460.raw	2	2	95.8986	3.006e-002
114	c:\transients\force3\27fwi461.raw	2	2	84.0327	6.313e-003
115	c:\transients\force3\27fwi471.raw	2	2	62.9300	5.969e-003
116	c:\transients\force3\27fwi473.raw	2	2	96.9376	8.743e-002
117	c:\transients\force3\27fwi210.raw	2	2	88.6928	1.536e-002
118	c:\transients\force3\27fwi253.raw	2	2	89.0389	4.591e-003
119	c:\transients\force3\27fwi285.raw	2	1	73.7959	8.651e-005
120	c:\transients\force3\27fwi324.raw	2	2	67.7494	1.119e-002
121	c:\transients\force3\27fwi344.raw	2	2	78.7721	1.981e-002
122	c:\transients\force3\27fwi394.raw	2	1	99.4473	6.464e-004
123	c:\transients\force3\27fwi450.raw	2	1	99.9549	4.670e-004
124	c:\transients\force3\27fwi465.raw	2	2	97.8687	2.611e-001
125	c:\transients\ken1\25fwn251.raw	3	3	100.000	4.097e-003
126	c:\transients\ken1\25fwn252.raw	3	0	99.9977	4.086e-009
127	c:\transients\ken1\25fwn254.raw	3	3	100.000	4.306e-007
128	c:\transients\ken1\25fwn255.raw	3	3	100.000	5.446e-007
129	c:\transients\ken1\25fwn264.raw	3	3	100.000	8.185e-007
130	c:\transients\ken1\25fwn273.raw	3	0	99.8053	1.354e-015
131	c:\transients\ken1\25fwn280.raw	3	3	100.000	1.327e-008
132	c:\transients\ken1\25fwn283.raw	3	3	100.000	7.423e-004
133	c:\transients\ken1\25fwn315.raw	3	3	85.8209	1.174e-016
134	c:\transients\ken1\25fwn332.raw	3	3	100.000	2.569e-012
135	c:\transients\ken1\25fwn341.raw	3	0	98.4275	1.187e-009
136	c:\transients\ken1\25fwn371.raw	3	3	100.000	3.830e-008
137	c:\transients\ken1\25fwn380.raw	3	3	51.2117	5.941e-013
138	c:\transients\ken1\25fwn382.raw	3	3	100.000	6.634e-006
139	c:\transients\ken1\25fwn384.raw	3	3	100.000	8.888e-006
140	c:\transients\ken1\25fwn392.raw	3	3	100.000	3.460e-006
141	c:\transients\ken1\25fwn401.raw	3	3	100.000	8.685e-007
142	c:\transients\ken1\25fwn402.raw	3	3	100.000	2.850e-009
143	c:\transients\ken1\25fwn404.raw	3	3	100.000	3.396e-008
144	c:\transients\ken1\25fwn405.raw	3	3	99.9968	4.040e-007
145	c:\transients\ken1\25fwn414.raw	3	3	100.000	3.665e-007
146	c:\transients\ken1\25fwn415.raw	3	3	100.000	1.448e-006
147	c:\transients\ken1\25fwn421.raw	3	3	100.000	2.476e-005
148	c:\transients\ken1\25fwn423.raw	3	0	93.8259	6.880e-011
149	c:\transients\ken1\25fwn432.raw	3	3	100.000	1.802e-004
150	c:\transients\ken1\25fwn433.raw	3	3	100.000	1.245e-004
151	c:\transients\ken1\25fwn435.raw	3	3	100.000	4.826e-006
152	c:\transients\ken1\25fwn441.raw	3	3	100.000	1.834e-003
153	c:\transients\ken1\25fwn450.raw	3	3	100.000	1.406e-006
154	c:\transients\ken1\25fwn452.raw	3	3	100.000	4.593e-013
155	c:\transients\ken1\25fwn453.raw	3	3	100.000	1.730e-009
156	c:\transients\ken1\25fwn455.raw	3	3	100.000	3.620e-009
157	c:\transients\ken1\25fwn263.raw	3	3	100.000	4.631e-009
158	c:\transients\ken1\25fwn312.raw	3	5	99.9889	4.077e-007

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
159	c:\transients\ken1\25fwn375.raw	3	3	100.000	9.029e-008
160	c:\transients\ken1\25fwn395.raw	3	3	100.000	2.937e-009
161	c:\transients\ken1\25fwn413.raw	3	0	88.1444	8.033e-012
162	c:\transients\ken1\25fwn430.raw	3	5	99.9978	3.381e-009
163	c:\transients\ken1\25fwn445.raw	3	3	100.000	1.006e-001
164	c:\transients\ken1\25fwn462.raw	3	3	100.000	1.000e-001
165	c:\transients\ken2\25fwn523.raw	4	4	100.000	2.630e-002
166	c:\transients\ken2\25fwn524.raw	4	4	100.000	5.076e-002
167	c:\transients\ken2\25fwn525.raw	4	4	100.000	5.867e-004
168	c:\transients\ken2\25fwn531.raw	4	4	100.000	8.510e-003
169	c:\transients\ken2\25fwn540.raw	4	4	100.000	6.136e-002
170	c:\transients\ken2\25fwn542.raw	4	4	100.000	4.104e-002
171	c:\transients\ken2\25fwn543.raw	4	4	100.000	1.326e-002
172	c:\transients\ken2\25fwn545.raw	4	4	100.000	4.119e-020
173	c:\transients\ken2\25fwn560.raw	4	4	100.000	8.928e-002
174	c:\transients\ken2\25fwn562.raw	4	4	100.000	5.545e-002
175	c:\transients\ken2\25fwn563.raw	4	4	100.000	2.189e-024
176	c:\transients\ken2\25fwn573.raw	4	4	100.000	2.650e-002
177	c:\transients\ken2\25fwn583.raw	4	4	100.000	5.079e-003
178	c:\transients\ken2\25fwn584.raw	4	4	100.000	2.891e-002
179	c:\transients\ken2\25fwn593.raw	4	4	100.000	3.265e-003
180	c:\transients\ken2\25fwn594.raw	4	4	100.000	1.354e-004
181	c:\transients\ken2\25fwo003.raw	4	4	100.000	3.402e-002
182	c:\transients\ken2\25fwo005.raw	4	4	100.000	1.481e-004
183	c:\transients\ken2\25fwo010.raw	4	4	100.000	7.779e-007
184	c:\transients\ken2\25fwo015.raw	4	4	100.000	1.222e-005
185	c:\transients\ken2\25fwo024.raw	4	4	100.000	7.913e-003
186	c:\transients\ken2\25fwo030.raw	4	4	100.000	3.046e-002
187	c:\transients\ken2\25fwo031.raw	4	4	100.000	6.089e-009
188	c:\transients\ken2\25fwo033.raw	4	4	100.000	4.375e-002
189	c:\transients\ken2\25fwo044.raw	4	4	100.000	2.253e-003
190	c:\transients\ken2\25fwo045.raw	4	4	100.000	1.468e-002
191	c:\transients\ken2\25fwo054.raw	4	4	100.000	5.851e-004
192	c:\transients\ken2\25fwo062.raw	4	4	100.000	1.825e-002
193	c:\transients\ken2\25fwo074.raw	4	4	100.000	4.699e-002
194	c:\transients\ken2\25fwo075.raw	4	4	100.000	4.087e-016
195	c:\transients\ken2\25fwo080.raw	4	4	100.000	8.502e-004
196	c:\transients\ken2\25fwo082.raw	4	4	100.000	3.229e-002
197	c:\transients\ken2\25fwo090.raw	4	4	100.000	2.101e-002
198	c:\transients\ken2\25fwn534.raw	4	4	100.000	3.550e-019
199	c:\transients\ken2\25fwn552.raw	4	4	100.000	1.568e-002
200	c:\transients\ken2\25fwn580.raw	4	4	100.000	1.942e-003
201	c:\transients\ken2\25fwo002.raw	4	4	100.000	1.543e-002
202	c:\transients\ken2\25fwo022.raw	4	4	100.000	1.773e-004
203	c:\transients\ken2\25fwo042.raw	4	4	100.000	1.875e-008
204	c:\transients\ken2\25fwo065.raw	4	4	100.000	5.071e-003
205	c:\transients\ken2\25fwo085.raw	4	4	100.000	1.567e-002
206	c:\transients\ken3\25fwo165.raw	5	5	100.000	2.266e-006
207	c:\transients\ken3\25fwo170.raw	5	5	100.000	6.856e-006
208	c:\transients\ken3\25fwo180.raw	5	5	100.000	4.872e-004
209	c:\transients\ken3\25fwo182.raw	5	5	100.000	3.275e-003
210	c:\transients\ken3\25fwo191.raw	5	5	100.000	7.395e-004
211	c:\transients\ken3\25fwo192.raw	5	5	100.000	1.128e-003
212	c:\transients\ken3\25fwo194.raw	5	3	94.9081	4.229e-022

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
213	c:\transients\ken3\25fwo195.raw	5	5	100.000	1.021e-015
214	c:\transients\ken3\25fwo212.raw	5	5	100.000	9.374e-007
215	c:\transients\ken3\25fwo213.raw	5	5	100.000	2.781e-003
216	c:\transients\ken3\25fwo215.raw	5	5	99.3902	1.081e-013
217	c:\transients\ken3\25fwo220.raw	5	5	100.000	1.251e-004
218	c:\transients\ken3\25fwo225.raw	5	5	100.000	3.273e-004
219	c:\transients\ken3\25fwo231.raw	5	5	100.000	3.835e-011
220	c:\transients\ken3\25fwo232.raw	5	5	100.000	2.300e-010
221	c:\transients\ken3\25fwo234.raw	5	5	100.000	6.114e-004
222	c:\transients\ken3\25fwo251.raw	5	3	100.000	5.218e-018
223	c:\transients\ken3\25fwo253.raw	5	5	100.000	3.909e-005
224	c:\transients\ken3\25fwo260.raw	5	5	100.000	1.602e-014
225	c:\transients\ken3\25fwo265.raw	5	5	100.000	3.032e-002
226	c:\transients\ken3\25fwo283.raw	5	5	100.000	4.451e-006
227	c:\transients\ken3\25fwo301.raw	5	5	100.000	1.752e-003
228	c:\transients\ken3\25fwo303.raw	5	5	100.000	2.837e-003
229	c:\transients\ken3\25fwo305.raw	5	3	98.3424	2.315e-014
230	c:\transients\ken3\25fwo315.raw	5	5	100.000	6.160e-003
231	c:\transients\ken3\25fwo321.raw	5	5	100.000	4.285e-006
232	c:\transients\ken3\25fwo325.raw	5	5	100.000	9.691e-003
233	c:\transients\ken3\25fwo332.raw	5	5	100.000	1.817e-002
234	c:\transients\ken3\25fwo341.raw	5	5	100.000	1.257e-003
235	c:\transients\ken3\25fwo342.raw	5	5	100.000	2.401e-003
236	c:\transients\ken3\25fwo344.raw	5	5	100.000	3.170e-004
237	c:\transients\ken3\25fwo352.raw	5	5	99.9999	2.724e-009
238	c:\transients\ken3\25fwp065.raw	5	5	100.000	8.101e-006
239	c:\transients\ken3\25fwp075.raw	5	5	100.000	2.319e-006
240	c:\transients\ken3\25fwo190.raw	5	3	100.000	8.911e-017
241	c:\transients\ken3\25fwo204.raw	5	5	100.000	1.619e-006
242	c:\transients\ken3\25fwo223.raw	5	5	100.000	2.677e-002
243	c:\transients\ken3\25fwo241.raw	5	5	100.000	1.697e-004
244	c:\transients\ken3\25fwo282.raw	5	5	100.000	4.656e-004
245	c:\transients\ken3\25fwo314.raw	5	5	100.000	1.704e-003
246	c:\transients\ken3\25fwo335.raw	5	5	100.000	4.398e-009
247	c:\transients\ken3\25fwp063.raw	5	5	100.000	3.420e-011
248	c:\transients\Yaesu1\25fwp323.raw	6	6	99.9999	3.225e-004
249	c:\transients\Yaesu1\25fwp325.raw	6	6	100.000	3.740e-007
250	c:\transients\Yaesu1\25fwp331.raw	6	6	100.000	1.957e-005
251	c:\transients\Yaesu1\26fwk212.raw	6	6	100.000	4.595e-004
252	c:\transients\Yaesu1\26fwk235.raw	6	6	100.000	2.606e-005
253	c:\transients\Yaesu1\26fwk240.raw	6	6	71.8568	4.890e-014
254	c:\transients\Yaesu1\26fwk242.raw	6	6	100.000	2.513e-003
255	c:\transients\Yaesu1\26fwk251.raw	6	7	90.5326	3.159e-021
256	c:\transients\Yaesu1\26fwk272.raw	6	6	100.000	1.491e-002
257	c:\transients\Yaesu1\26fwk274.raw	6	6	100.000	1.327e-004
258	c:\transients\Yaesu1\26fwk280.raw	6	6	100.000	6.984e-004
259	c:\transients\Yaesu1\26fwk281.raw	6	6	100.000	3.935e-004
260	c:\transients\Yaesu1\26fwk312.raw	6	6	100.000	1.764e-004
261	c:\transients\Yaesu1\26fwk314.raw	6	6	100.000	1.113e-002
262	c:\transients\Yaesu1\26fwk332.raw	6	6	100.000	4.630e-008
263	c:\transients\Yaesu1\26fwk334.raw	6	6	100.000	4.299e-005
264	c:\transients\Yaesu1\26fwk343.raw	6	6	100.000	9.004e-005
265	c:\transients\Yaesu1\26fwk345.raw	6	6	100.000	2.901e-004
266	c:\transients\Yaesu1\26fwk355.raw	6	6	100.000	1.615e-004

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
267	c:\transients\Yaesu1\26fwk361.raw	6	6	100.000	1.202e-004
268	c:\transients\Yaesu1\26fwk370.raw	6	6	100.000	1.471e-003
269	c:\transients\Yaesu1\26fwk372.raw	6	6	100.000	2.156e-003
270	c:\transients\Yaesu1\26fwk435.raw	6	6	99.5968	5.027e-011
271	c:\transients\Yaesu1\26fwk450.raw	6	6	100.000	2.357e-008
272	c:\transients\Yaesu1\26fwk455.raw	6	6	100.000	6.752e-005
273	c:\transients\Yaesu1\26fwk483.raw	6	6	100.000	1.081e-003
274	c:\transients\Yaesu1\26fwk502.raw	6	6	100.000	1.158e-005
275	c:\transients\Yaesu1\26fwk505.raw	6	6	100.000	1.108e-004
276	c:\transients\Yaesu1\26fwk524.raw	6	6	100.000	3.185e-002
277	c:\transients\Yaesu1\26fwk532.raw	6	6	100.000	3.461e-002
278	c:\transients\Yaesu1\26fwk541.raw	6	6	100.000	3.973e-004
279	c:\transients\Yaesu1\26fwk544.raw	6	6	100.000	2.918e-004
280	c:\transients\Yaesu1\26fwk564.raw	6	6	100.000	3.809e-005
281	c:\transients\Yaesu1\26fwk570.raw	6	6	100.000	4.696e-007
282	c:\transients\Yaesu1\26fwk571.raw	6	6	100.000	3.965e-002
283	c:\transients\Yaesu1\26fwk573.raw	6	6	100.000	2.196e-004
284	c:\transients\Yaesu1\26fwk575.raw	6	6	100.000	9.776e-004
285	c:\transients\Yaesu1\26fwk585.raw	6	6	100.000	1.544e-004
286	c:\transients\Yaesu1\26fwl002.raw	6	6	100.000	3.724e-003
287	c:\transients\Yaesu1\26fwl003.raw	6	6	100.000	3.512e-006
288	c:\transients\Yaesu1\26fwk233.raw	6	6	100.000	3.645e-005
289	c:\transients\Yaesu1\26fwk264.raw	6	6	100.000	2.086e-003
290	c:\transients\Yaesu1\26fwk285.raw	6	6	100.000	7.528e-003
291	c:\transients\Yaesu1\26fwk341.raw	6	6	100.000	2.734e-004
292	c:\transients\Yaesu1\26fwk365.raw	6	6	100.000	1.077e-005
293	c:\transients\Yaesu1\26fwk453.raw	6	6	100.000	2.865e-004
294	c:\transients\Yaesu1\26fwk522.raw	6	6	100.000	2.885e-005
295	c:\transients\Yaesu1\26fwk551.raw	6	6	100.000	4.182e-003
296	c:\transients\Yaesu2\26fwl125.raw	7	7	100.000	3.606e-008
297	c:\transients\Yaesu2\26fwl132.raw	7	7	100.000	9.171e-005
298	c:\transients\Yaesu2\26fwl133.raw	7	7	100.000	1.551e-004
299	c:\transients\Yaesu2\26fwl140.raw	7	7	100.000	2.718e-014
300	c:\transients\Yaesu2\26fwl173.raw	7	7	99.9851	3.218e-013
301	c:\transients\Yaesu2\26fwl180.raw	7	7	100.000	1.349e-013
302	c:\transients\Yaesu2\26fwl181.raw	7	7	99.9669	2.694e-005
303	c:\transients\Yaesu2\26fwl190.raw	7	7	100.000	1.506e-007
304	c:\transients\Yaesu2\26fwl200.raw	7	7	99.9651	1.739e-005
305	c:\transients\Yaesu2\26fwl202.raw	7	7	87.5346	1.275e-008
306	c:\transients\Yaesu2\26fwl204.raw	7	7	99.9993	1.524e-005
307	c:\transients\Yaesu2\26fwl212.raw	7	7	100.000	1.051e-004
308	c:\transients\Yaesu2\26fwl224.raw	7	7	100.000	6.982e-019
309	c:\transients\Yaesu2\26fwl225.raw	7	7	99.6486	1.388e-016
310	c:\transients\Yaesu2\26fwl231.raw	7	7	100.000	2.076e-002
311	c:\transients\Yaesu2\26fwl233.raw	7	7	100.000	3.475e-011
312	c:\transients\Yaesu2\26fwl243.raw	7	7	99.7349	2.920e-008
313	c:\transients\Yaesu2\26fwl245.raw	7	7	100.000	8.871e-004
314	c:\transients\Yaesu2\26fwl251.raw	7	7	100.000	2.990e-005
315	c:\transients\Yaesu2\26fwl254.raw	7	7	99.9448	1.296e-006
316	c:\transients\Yaesu2\26fwl264.raw	7	7	100.000	2.073e-007
317	c:\transients\Yaesu2\26fwl270.raw	7	7	100.000	1.936e-005
318	c:\transients\Yaesu2\26fwl272.raw	7	7	100.000	9.947e-004
319	c:\transients\Yaesu2\26fwl274.raw	7	7	99.9994	3.212e-005
320	c:\transients\Yaesu2\26fwl283.raw	7	7	100.000	1.762e-005

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
321	c:\transients\Yaesu2\26fw\285.raw	7	7	100.000	3.444e-003
322	c:\transients\Yaesu2\26fw\293.raw	7	7	100.000	8.811e-003
323	c:\transients\Yaesu2\26fw\300.raw	7	7	100.000	7.420e-008
324	c:\transients\Yaesu2\26fw\312.raw	7	7	100.000	1.038e-004
325	c:\transients\Yaesu2\26fw\314.raw	7	7	100.000	1.250e-002
326	c:\transients\Yaesu2\26fw\320.raw	7	7	60.5590	1.612e-008
327	c:\transients\Yaesu2\26fw\322.raw	7	7	100.000	1.611e-006
328	c:\transients\Yaesu2\26fw\153.raw	7	7	100.000	4.914e-004
329	c:\transients\Yaesu2\26fw\194.raw	7	7	100.000	1.990e-003
330	c:\transients\Yaesu2\26fw\220.raw	7	7	100.000	2.313e-004
331	c:\transients\Yaesu2\26fw\241.raw	7	7	99.9997	3.951e-005
332	c:\transients\Yaesu2\26fw\262.raw	7	7	100.000	1.904e-006
333	c:\transients\Yaesu2\26fw\281.raw	7	7	100.000	1.149e-008
334	c:\transients\Yaesu2\26fw\310.raw	7	7	85.8617	1.557e-009
335	c:\transients\Yaesu2\26fw\332.raw	7	7	100.000	1.213e-003

Number of Correct Classifications: 306/335.

VALIDATION SET # 3 (RANDOM SELECTION OF 30)

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
1	d:\Tran\force1\26fwo325.raw	0	0	98.0059	1.450e-003
2	d:\Tran\force1\26fwo341.raw	0	0	99.9906	2.497e-003
3	d:\Tran\force1\26fwo362.raw	0	0	99.9023	6.307e-002
4	d:\Tran\force1\26fwo380.raw	0	0	99.9286	3.158e-002
5	d:\Tran\force1\26fwo390.raw	0	0	100.000	2.208e-003
6	d:\Tran\force1\26fwo395.raw	0	0	100.000	2.560e-006
7	d:\Tran\force1\26fwo413.raw	0	0	100.000	5.837e-003
8	d:\Tran\force1\26fwo440.raw	0	0	99.6506	1.596e-001
9	d:\Tran\force1\26fwo453.raw	0	0	99.9666	2.464e-003
10	d:\Tran\force1\26fwo461.raw	0	0	99.3619	7.276e-002
11	d:\Tran\force1\26fwo465.raw	0	0	100.000	9.332e-004
12	d:\Tran\force1\26fwo481.raw	0	0	99.7808	7.438e-002
13	d:\Tran\force1\26fwo492.raw	0	0	99.9973	1.149e-003
14	d:\Tran\force1\26fwo500.raw	0	0	100.000	4.064e-004
15	d:\Tran\force1\26fwo504.raw	0	0	96.1232	1.086e-002
16	d:\Tran\force1\26fwo515.raw	0	6	100.000	1.853e-014
17	d:\Tran\force1\26fwo524.raw	0	0	99.6216	1.482e-001
18	d:\Tran\force1\26fwo532.raw	0	0	99.2166	3.432e-002
19	d:\Tran\force1\26fwo545.raw	0	6	99.8118	5.238e-014
20	d:\Tran\force1\26fwo554.raw	0	0	99.6416	1.390e-001
21	d:\Tran\force2\26fwp571.raw	1	0	93.6722	1.321e-005
22	d:\Tran\force2\26fwp592.raw	1	1	96.8041	2.016e-003
23	d:\Tran\force2\26fwq042.raw	1	1	79.2701	1.648e-003
24	d:\Tran\force2\26fwq061.raw	1	1	92.7370	9.562e-005
25	d:\Tran\force2\26fwq071.raw	1	1	76.5071	3.113e-005
26	d:\Tran\force2\26fwq084.raw	1	1	99.8018	1.054e-002
27	d:\Tran\force2\26fwq094.raw	1	1	91.1793	2.227e-002
28	d:\Tran\force2\26fwq125.raw	1	1	76.2983	2.415e-004
29	d:\Tran\force2\26fwq134.raw	1	1	95.8942	1.831e-002
30	d:\Tran\force2\26fwq144.raw	1	1	70.8681	3.615e-003
31	d:\Tran\force2\26fwq152.raw	1	1	96.8296	1.707e-002
32	d:\Tran\force2\26fwq160.raw	1	1	99.9161	3.054e-003
33	d:\Tran\force2\26fwq184.raw	1	1	99.5685	2.995e-002
34	d:\Tran\force2\26fwq202.raw	1	1	84.4025	2.181e-004
35	d:\Tran\force2\26fwq235.raw	1	1	99.5454	2.629e-003
36	d:\Tran\force2\26fwq253.raw	1	1	59.4970	1.216e-004
37	d:\Tran\force2\26fwq262.raw	1	1	98.2795	1.594e-003
38	d:\Tran\force2\26fwq265.raw	1	1	98.5558	8.645e-004
39	d:\Tran\force2\26fwq265.raw	1	1	98.5558	8.645e-004
40	d:\Tran\force2\26fwq273.raw	1	2	81.2337	1.285e-003
41	d:\Tran\force2\26fwq290.raw	1	1	92.1501	8.328e-004
42	d:\Tran\force2\26fwq330.raw	1	0	52.2138	2.031e-005
43	d:\Tran\force3\27fwi153.raw	2	2	98.3941	4.284e-002
44	d:\Tran\force3\27fwi190.raw	2	2	84.7330	3.516e-008
45	d:\Tran\force3\27fwi194.raw	2	2	88.6738	2.107e-002
46	d:\Tran\force3\27fwi210.raw	2	2	87.4535	2.407e-003
47	d:\Tran\force3\27fwi214.raw	2	2	73.8855	2.964e-003
48	d:\Tran\force3\27fwi250.raw	2	2	81.5460	6.213e-003
49	d:\Tran\force3\27fwi253.raw	2	2	99.7210	1.917e-002
50	d:\Tran\force3\27fwi264.raw	2	2	68.9010	1.919e-005

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
51	d:\Tran\force3\27fwi311.raw	2	2	72.1911	5.476e-005
52	d:\Tran\force3\27fwi315.raw	2	2	99.7148	3.805e-002
53	d:\Tran\force3\27fwi324.raw	2	2	95.2048	1.297e-002
54	d:\Tran\force3\27fwi332.raw	2	2	95.2416	1.334e-002
55	d:\Tran\force3\27fwi344.raw	2	2	77.7709	3.254e-003
56	d:\Tran\force3\27fwi355.raw	2	2	57.9581	2.811e-003
57	d:\Tran\force3\27fwi381.raw	2	2	99.8265	5.343e-003
58	d:\Tran\force3\27fwi394.raw	2	1	97.0187	1.136e-004
59	d:\Tran\force3\27fwi425.raw	2	2	75.7757	5.302e-005
60	d:\Tran\force3\27fwi433.raw	2	2	68.6129	5.687e-006
61	d:\Tran\force3\27fwi454.raw	2	2	95.7428	1.166e-002
62	d:\Tran\force3\27fwi461.raw	2	2	98.9174	1.285e-002
63	d:\Tran\force3\27fwi465.raw	2	2	98.8306	9.202e-002
64	d:\Tran\force3\27fwi473.raw	2	2	98.7733	2.531e-002
65	d:\Tran\ken1\25fwn245.raw	3	3	100.000	3.715e-009
66	d:\Tran\ken1\25fwn255.raw	3	3	99.9992	2.697e-011
67	d:\Tran\ken1\25fwn263.raw	3	3	100.000	1.152e-006
68	d:\Tran\ken1\25fwn273.raw	3	5	80.2422	2.325e-018
69	d:\Tran\ken1\25fwn332.raw	3	3	100.000	5.884e-007
70	d:\Tran\ken1\25fwn371.raw	3	3	100.000	9.249e-006
71	d:\Tran\ken1\25fwn375.raw	3	3	100.000	1.559e-005
72	d:\Tran\ken1\25fwn382.raw	3	3	100.000	9.153e-005
73	d:\Tran\ken1\25fwn392.raw	3	3	100.000	1.318e-003
74	d:\Tran\ken1\25fwn395.raw	3	3	100.000	1.282e-010
75	d:\Tran\ken1\25fwn405.raw	3	3	99.4762	1.083e-006
76	d:\Tran\ken1\25fwn413.raw	3	3	100.000	2.866e-007
77	d:\Tran\ken1\25fwn415.raw	3	3	100.000	4.421e-004
78	d:\Tran\ken1\25fwn423.raw	3	3	100.000	3.538e-004
79	d:\Tran\ken1\25fwn430.raw	3	5	100.000	1.237e-012
80	d:\Tran\ken1\25fwn433.raw	3	3	100.000	3.311e-008
81	d:\Tran\ken1\25fwn445.raw	3	3	100.000	2.558e-005
82	d:\Tran\ken1\25fwn452.raw	3	3	100.000	1.607e-003
83	d:\Tran\ken1\25fwn455.raw	3	3	100.000	2.637e-010
84	d:\Tran\ken1\25fwn462.raw	3	3	100.000	1.470e-012
85	d:\Tran\ken2\25fwn524.raw	4	4	100.000	3.597e-002
86	d:\Tran\ken2\25fwn531.raw	4	4	100.000	4.785e-003
87	d:\Tran\ken2\25fwn534.raw	4	4	100.000	1.516e-010
88	d:\Tran\ken2\25fwn545.raw	4	7	100.000	1.306e-018
89	d:\Tran\ken2\25fwn552.raw	4	4	100.000	2.019e-002
90	d:\Tran\ken2\25fwn562.raw	4	4	100.000	1.617e-002
91	d:\Tran\ken2\25fwn580.raw	4	4	100.000	4.909e-004
92	d:\Tran\ken2\25fwn584.raw	4	4	100.000	1.903e-002
93	d:\Tran\ken2\25fwn594.raw	4	4	100.000	4.413e-005
94	d:\Tran\ken2\25fwo002.raw	4	4	100.000	3.019e-002
95	d:\Tran\ken2\25fwo005.raw	4	4	100.000	1.554e-003
96	d:\Tran\ken2\25fwo015.raw	4	4	100.000	9.196e-003
97	d:\Tran\ken2\25fwo022.raw	4	4	100.000	1.158e-004
98	d:\Tran\ken2\25fwo030.raw	4	4	100.000	5.456e-003
99	d:\Tran\ken2\25fwo033.raw	4	4	100.000	4.919e-002
100	d:\Tran\ken2\25fwo045.raw	4	4	100.000	2.618e-002
101	d:\Tran\ken2\25fwo062.raw	4	4	100.000	1.138e-002
102	d:\Tran\ken2\25fwo065.raw	4	4	100.000	6.114e-004
103	d:\Tran\ken2\25fwo075.raw	4	4	100.000	1.125e-006
104	d:\Tran\ken2\25fwo082.raw	4	4	100.000	1.546e-002

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
105	d:\Tran\ken2\25fwo085.raw	4	4	100.000	4.629e-003
106	d:\Tran\ken3\25fwo160.raw	5	5	100.000	3.072e-005
107	d:\Tran\ken3\25fwo182.raw	5	5	100.000	5.783e-004
108	d:\Tran\ken3\25fwo190.raw	5	5	100.000	1.401e-006
109	d:\Tran\ken3\25fwo192.raw	5	5	100.000	1.006e-003
110	d:\Tran\ken3\25fwo195.raw	5	5	100.000	4.155e-003
111	d:\Tran\ken3\25fwo213.raw	5	5	100.000	2.414e-004
112	d:\Tran\ken3\25fwo220.raw	5	5	100.000	3.159e-006
113	d:\Tran\ken3\25fwo223.raw	5	5	100.000	5.825e-003
114	d:\Tran\ken3\25fwo231.raw	5	5	100.000	2.901e-005
115	d:\Tran\ken3\25fwo241.raw	5	5	100.000	4.933e-005
116	d:\Tran\ken3\25fwo253.raw	5	5	100.000	1.895e-004
117	d:\Tran\ken3\25fwo265.raw	5	5	100.000	1.040e-002
118	d:\Tran\ken3\25fwo282.raw	5	5	100.000	2.069e-006
119	d:\Tran\ken3\25fwo301.raw	5	5	100.000	2.834e-003
120	d:\Tran\ken3\25fwo305.raw	5	3	100.000	2.653e-012
121	d:\Tran\ken3\25fwo314.raw	5	5	100.000	1.063e-004
122	d:\Tran\ken3\25fwo321.raw	5	5	100.000	3.785e-004
123	d:\Tran\ken3\25fwo332.raw	5	5	100.000	1.185e-002
124	d:\Tran\ken3\25fwo342.raw	5	5	100.000	4.096e-004
125	d:\Tran\ken3\25fwo352.raw	5	5	99.9923	4.796e-010
126	d:\Tran\ken3\25fwp063.raw	5	5	100.000	2.526e-013
127	d:\Tran\ken3\25fwp075.raw	5	5	100.000	1.390e-008
128	d:\Tran\Yaesu1\25fwp321.raw	6	6	99.9999	2.215e-007
129	d:\Tran\Yaesu1\25fwp325.raw	6	6	100.000	6.238e-009
130	d:\Tran\Yaesu1\26fwk212.raw	6	6	100.000	1.696e-004
131	d:\Tran\Yaesu1\26fwk233.raw	6	6	100.000	3.910e-004
132	d:\Tran\Yaesu1\26fwk240.raw	6	6	100.000	5.379e-004
133	d:\Tran\Yaesu1\26fwk264.raw	6	6	100.000	2.776e-003
134	d:\Tran\Yaesu1\26fwk274.raw	6	6	99.9978	2.121e-005
135	d:\Tran\Yaesu1\26fwk281.raw	6	6	100.000	3.593e-005
136	d:\Tran\Yaesu1\26fwk285.raw	6	6	100.000	8.553e-004
137	d:\Tran\Yaesu1\26fwk314.raw	6	6	100.000	2.221e-003
138	d:\Tran\Yaesu1\26fwk334.raw	6	6	100.000	1.861e-003
139	d:\Tran\Yaesu1\26fwk341.raw	6	6	100.000	3.537e-003
140	d:\Tran\Yaesu1\26fwk345.raw	6	6	100.000	6.991e-003
141	d:\Tran\Yaesu1\26fwk361.raw	6	6	100.000	2.399e-003
142	d:\Tran\Yaesu1\26fwk365.raw	6	6	100.000	5.736e-005
143	d:\Tran\Yaesu1\26fwk372.raw	6	6	100.000	1.921e-004
144	d:\Tran\Yaesu1\26fwk450.raw	6	6	99.7338	1.679e-011
145	d:\Tran\Yaesu1\26fwk453.raw	6	6	100.000	1.047e-003
146	d:\Tran\Yaesu1\26fwk483.raw	6	6	100.000	1.911e-004
147	d:\Tran\Yaesu1\26fwk505.raw	6	6	100.000	2.060e-004
148	d:\Tran\Yaesu1\26fwk522.raw	6	6	99.9996	1.096e-004
149	d:\Tran\Yaesu1\26fwk532.raw	6	6	100.000	7.578e-003
150	d:\Tran\Yaesu1\26fwk544.raw	6	6	100.000	1.136e-005
151	d:\Tran\Yaesu1\26fwk551.raw	6	6	100.000	3.309e-004
152	d:\Tran\Yaesu1\26fwk570.raw	6	6	100.000	7.087e-004
153	d:\Tran\Yaesu1\26fwk573.raw	6	6	100.000	1.576e-003
154	d:\Tran\Yaesu1\26fwk585.raw	6	6	100.000	1.735e-005
155	d:\Tran\Yaesu1\26fwl003.raw	6	6	100.000	1.569e-007
156	d:\Tran\Yaesu2\26fwl132.raw	7	7	100.000	2.020e-005
157	d:\Tran\Yaesu2\26fwl140.raw	7	7	100.000	5.849e-010
158	d:\Tran\Yaesu2\26fwl153.raw	7	7	100.000	2.085e-003

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
159	d:\Tran\Yaesu2\26fwl190.raw	7	7	100.000	4.361e-004
160	d:\Tran\Yaesu2\26fwl194.raw	7	7	100.000	1.405e-004
161	d:\Tran\Yaesu2\26fwl202.raw	7	7	99.9997	2.052e-003
162	d:\Tran\Yaesu2\26fwl220.raw	7	7	100.000	7.483e-006
163	d:\Tran\Yaesu2\26fwl233.raw	7	7	99.9728	3.029e-011
164	d:\Tran\Yaesu2\26fwl241.raw	7	7	100.000	5.055e-004
165	d:\Tran\Yaesu2\26fwl245.raw	7	7	100.000	4.549e-005
166	d:\Tran\Yaesu2\26fwl254.raw	7	7	100.000	4.846e-004
167	d:\Tran\Yaesu2\26fwl262.raw	7	7	100.000	1.840e-005
168	d:\Tran\Yaesu2\26fwl270.raw	7	7	100.000	4.586e-006
169	d:\Tran\Yaesu2\26fwl274.raw	7	7	100.000	2.184e-004
170	d:\Tran\Yaesu2\26fwl281.raw	7	7	100.000	6.421e-008
171	d:\Tran\Yaesu2\26fwl285.raw	7	7	100.000	9.222e-004
172	d:\Tran\Yaesu2\26fwl300.raw	7	7	100.000	4.580e-008
173	d:\Tran\Yaesu2\26fwl310.raw	7	7	100.000	2.249e-003
174	d:\Tran\Yaesu2\26fwl314.raw	7	7	100.000	1.988e-003
175	d:\Tran\Yaesu2\26fwl332.raw	7	7	100.000	1.085e-004

Number of Correct Classifications: 165/175.

VALIDATION SET #3 WITH EXTENDED TRAINING

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
1	d:\Tran\force1\26fwo325.raw	0	0	100.000	1.490e-005
2	d:\Tran\force1\26fwo341.raw	0	0	99.9950	3.097e-005
3	d:\Tran\force1\26fwo362.raw	0	0	99.9979	2.104e-002
4	d:\Tran\force1\26fwo380.raw	0	0	99.9988	1.312e-002
5	d:\Tran\force1\26fwo390.raw	0	0	100.000	1.519e-003
6	d:\Tran\force1\26fwo395.raw	0	0	100.000	5.643e-010
7	d:\Tran\force1\26fwo413.raw	0	0	100.000	1.243e-003
8	d:\Tran\force1\26fwo440.raw	0	0	99.9930	1.234e-001
9	d:\Tran\force1\26fwo453.raw	0	0	99.9994	8.948e-004
10	d:\Tran\force1\26fwo461.raw	0	0	99.9478	3.106e-002
11	d:\Tran\force1\26fwo465.raw	0	0	100.000	1.077e-004
12	d:\Tran\force1\26fwo481.raw	0	0	99.9931	2.926e-002
13	d:\Tran\force1\26fwo492.raw	0	0	100.000	3.006e-004
14	d:\Tran\force1\26fwo500.raw	0	0	100.000	1.838e-007
15	d:\Tran\force1\26fwo504.raw	0	0	99.5734	3.478e-003
16	d:\Tran\force1\26fwo515.raw	0	6	100.000	2.877e-023
17	d:\Tran\force1\26fwo524.raw	0	0	99.9926	1.150e-001
18	d:\Tran\force1\26fwo532.raw	0	0	99.9741	2.848e-002
19	d:\Tran\force1\26fwo545.raw	0	6	98.3772	2.159e-022
20	d:\Tran\force1\26fwo554.raw	0	0	99.9952	1.166e-001
21	d:\Tran\force2\26fwq571.raw	1	0	79.0646	1.416e-006
22	d:\Tran\force2\26fwq592.raw	1	1	99.9953	1.866e-003
23	d:\Tran\force2\26fwq042.raw	1	1	99.9162	5.474e-003
24	d:\Tran\force2\26fwq061.raw	1	1	99.9946	4.652e-004
25	d:\Tran\force2\26fwq071.raw	1	1	99.9809	1.252e-004
26	d:\Tran\force2\26fwq084.raw	1	1	99.9972	9.991e-003
27	d:\Tran\force2\26fwq094.raw	1	1	93.5082	6.595e-003
28	d:\Tran\force2\26fwq125.raw	1	1	83.9723	2.869e-004
29	d:\Tran\force2\26fwq134.raw	1	1	99.9646	3.253e-002
30	d:\Tran\force2\26fwq144.raw	1	1	99.0410	2.185e-003
31	d:\Tran\force2\26fwq152.raw	1	1	99.9693	2.620e-002
32	d:\Tran\force2\26fwq160.raw	1	1	99.3685	5.340e-005
33	d:\Tran\force2\26fwq184.raw	1	1	99.3917	1.704e-002
34	d:\Tran\force2\26fwq202.raw	1	1	98.2697	4.353e-005
35	d:\Tran\force2\26fwq235.raw	1	1	99.9064	1.417e-003
36	d:\Tran\force2\26fwq253.raw	1	1	98.6045	6.049e-004
37	d:\Tran\force2\26fwq262.raw	1	1	99.9862	1.298e-004
38	d:\Tran\force2\26fwq265.raw	1	1	99.4406	2.576e-004
39	d:\Tran\force2\26fwq265.raw	1	1	99.4406	2.576e-004
40	d:\Tran\force2\26fwq273.raw	1	1	99.0124	2.360e-003
41	d:\Tran\force2\26fwq290.raw	1	1	99.9817	5.452e-003
42	d:\Tran\force2\26fwq330.raw	1	1	87.9275	3.697e-007
43	d:\Tran\force3\27fwi153.raw	2	2	99.9535	2.249e-002
44	d:\Tran\force3\27fwi190.raw	2	2	99.9904	1.739e-009
45	d:\Tran\force3\27fwi194.raw	2	2	99.5988	9.376e-003
46	d:\Tran\force3\27fwi210.raw	2	2	99.9519	4.567e-003
47	d:\Tran\force3\27fwi214.raw	2	2	80.2974	3.152e-004
48	d:\Tran\force3\27fwi250.raw	2	0	65.6177	2.861e-006
49	d:\Tran\force3\27fwi253.raw	2	2	99.9973	2.872e-002
50	d:\Tran\force3\27fwi264.raw	2	2	99.9998	3.568e-004

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
51	d:\Tran\force3\27fwi311.raw	2	2	99.9993	1.478e-004
52	d:\Tran\force3\27fwi315.raw	2	2	99.9868	2.901e-002
53	d:\Tran\force3\27fwi324.raw	2	2	99.4343	7.105e-003
54	d:\Tran\force3\27fwi332.raw	2	2	99.3552	6.830e-003
55	d:\Tran\force3\27fwi344.raw	2	2	96.9268	7.752e-004
56	d:\Tran\force3\27fwi355.raw	2	2	69.0149	2.870e-004
57	d:\Tran\force3\27fwi381.raw	2	2	99.9931	1.181e-002
58	d:\Tran\force3\27fwi394.raw	2	1	96.4071	3.875e-005
59	d:\Tran\force3\27fwi425.raw	2	2	99.5058	4.979e-005
60	d:\Tran\force3\27fwi433.raw	2	2	99.9995	1.506e-004
61	d:\Tran\force3\27fwi454.raw	2	2	99.9879	1.616e-002
62	d:\Tran\force3\27fwi461.raw	2	2	99.9969	8.075e-003
63	d:\Tran\force3\27fwi465.raw	2	2	99.9950	6.967e-002
64	d:\Tran\force3\27fwi473.raw	2	2	99.9922	2.125e-002
65	d:\Tran\ken1\25fwn245.raw	3	3	100.000	2.813e-018
66	d:\Tran\ken1\25fwn255.raw	3	3	100.000	2.499e-017
67	d:\Tran\ken1\25fwn263.raw	3	3	100.000	2.335e-019
68	d:\Tran\ken1\25fwn273.raw	3	3	100.000	2.641e-038
69	d:\Tran\ken1\25fwn332.raw	3	3	100.000	2.982e-019
70	d:\Tran\ken1\25fwn371.raw	3	3	100.000	6.703e-013
71	d:\Tran\ken1\25fwn375.raw	3	3	100.000	1.048e-006
72	d:\Tran\ken1\25fwn382.raw	3	3	100.000	6.747e-010
73	d:\Tran\ken1\25fwn392.raw	3	3	100.000	1.835e-003
74	d:\Tran\ken1\25fwn395.raw	3	3	100.000	5.795e-018
75	d:\Tran\ken1\25fwn405.raw	3	3	83.0458	3.201e-010
76	d:\Tran\ken1\25fwn413.raw	3	3	100.000	1.903e-009
77	d:\Tran\ken1\25fwn415.raw	3	3	100.000	1.208e-003
78	d:\Tran\ken1\25fwn423.raw	3	3	100.000	6.634e-006
79	d:\Tran\ken1\25fwn430.raw	3	5	100.000	1.872e-016
80	d:\Tran\ken1\25fwn433.raw	3	3	100.000	4.882e-017
81	d:\Tran\ken1\25fwn445.raw	3	3	100.000	4.985e-004
82	d:\Tran\ken1\25fwn452.raw	3	3	100.000	6.011e-005
83	d:\Tran\ken1\25fwn455.raw	3	3	100.000	4.863e-018
84	d:\Tran\ken1\25fwn462.raw	3	3	100.000	3.861e-014
85	d:\Tran\ken2\25fwn524.raw	4	4	100.000	1.426e-002
86	d:\Tran\ken2\25fwn531.raw	4	4	100.000	7.499e-004
87	d:\Tran\ken2\25fwn534.raw	4	4	100.000	1.606e-023
88	d:\Tran\ken2\25fwn545.raw	4	4	100.000	2.036e-028
89	d:\Tran\ken2\25fwn552.raw	4	4	100.000	8.873e-003
90	d:\Tran\ken2\25fwn562.raw	4	4	100.000	5.810e-003
91	d:\Tran\ken2\25fwn580.raw	4	4	100.000	1.037e-005
92	d:\Tran\ken2\25fwn584.raw	4	4	100.000	6.223e-003
93	d:\Tran\ken2\25fwn594.raw	4	4	100.000	6.954e-007
94	d:\Tran\ken2\25fwo002.raw	4	4	100.000	1.197e-002
95	d:\Tran\ken2\25fwo005.raw	4	4	100.000	1.849e-004
96	d:\Tran\ken2\25fwo015.raw	4	4	100.000	7.991e-004
97	d:\Tran\ken2\25fwo022.raw	4	4	100.000	1.680e-009
98	d:\Tran\ken2\25fwo030.raw	4	4	100.000	3.544e-004
99	d:\Tran\ken2\25fwo033.raw	4	4	100.000	2.084e-002
100	d:\Tran\ken2\25fwo045.raw	4	4	100.000	1.177e-002
101	d:\Tran\ken2\25fwo062.raw	4	4	100.000	3.995e-003
102	d:\Tran\ken2\25fwo065.raw	4	4	100.000	8.448e-005
103	d:\Tran\ken2\25fwo075.raw	4	4	100.000	1.204e-018
104	d:\Tran\ken2\25fwo082.raw	4	4	100.000	5.577e-003

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
105	d:\Tran\ken2\25fwo085.raw	4	4	100.000	2.583e-004
106	d:\Tran\ken3\25fwo160.raw	5	5	100.000	2.906e-015
107	d:\Tran\ken3\25fwo182.raw	5	5	100.000	7.628e-006
108	d:\Tran\ken3\25fwo190.raw	5	5	100.000	2.828e-016
109	d:\Tran\ken3\25fwo192.raw	5	5	100.000	5.478e-008
110	d:\Tran\ken3\25fwo195.raw	5	5	100.000	1.022e-003
111	d:\Tran\ken3\25fwo213.raw	5	5	100.000	8.576e-009
112	d:\Tran\ken3\25fwo220.raw	5	5	100.000	2.532e-008
113	d:\Tran\ken3\25fwo223.raw	5	5	100.000	1.904e-007
114	d:\Tran\ken3\25fwo231.raw	5	5	100.000	1.405e-015
115	d:\Tran\ken3\25fwo241.raw	5	5	100.000	6.713e-023
116	d:\Tran\ken3\25fwo253.raw	5	5	100.000	7.844e-005
117	d:\Tran\ken3\25fwo265.raw	5	5	100.000	2.371e-005
118	d:\Tran\ken3\25fwo282.raw	5	5	100.000	2.261e-014
119	d:\Tran\ken3\25fwo301.raw	5	5	100.000	4.100e-006
120	d:\Tran\ken3\25fwo305.raw	5	3	100.000	4.872e-017
121	d:\Tran\ken3\25fwo314.raw	5	5	100.000	1.374e-006
122	d:\Tran\ken3\25fwo321.raw	5	5	100.000	7.806e-004
123	d:\Tran\ken3\25fwo332.raw	5	5	100.000	1.690e-004
124	d:\Tran\ken3\25fwo342.raw	5	5	100.000	1.634e-004
125	d:\Tran\ken3\25fwo352.raw	5	5	100.000	3.142e-012
126	d:\Tran\ken3\25fwo063.raw	5	5	100.000	2.503e-018
127	d:\Tran\ken3\25fwo075.raw	5	5	100.000	1.269e-013
128	d:\Tran\Yaesu1\25fwo321.raw	6	6	100.000	7.979e-014
129	d:\Tran\Yaesu1\25fwo325.raw	6	6	100.000	3.135e-011
130	d:\Tran\Yaesu1\26fwo212.raw	6	6	100.000	6.675e-006
131	d:\Tran\Yaesu1\26fwo233.raw	6	6	100.000	6.059e-010
132	d:\Tran\Yaesu1\26fwo240.raw	6	6	100.000	4.104e-006
133	d:\Tran\Yaesu1\26fwo264.raw	6	6	100.000	1.273e-004
134	d:\Tran\Yaesu1\26fwo274.raw	6	6	100.000	6.214e-007
135	d:\Tran\Yaesu1\26fwo281.raw	6	6	100.000	6.978e-007
136	d:\Tran\Yaesu1\26fwo285.raw	6	6	100.000	3.136e-007
137	d:\Tran\Yaesu1\26fwo314.raw	6	6	100.000	8.322e-004
138	d:\Tran\Yaesu1\26fwo334.raw	6	6	100.000	1.988e-004
139	d:\Tran\Yaesu1\26fwo341.raw	6	6	100.000	1.250e-003
140	d:\Tran\Yaesu1\26fwo345.raw	6	6	100.000	5.844e-004
141	d:\Tran\Yaesu1\26fwo361.raw	6	6	100.000	3.168e-004
142	d:\Tran\Yaesu1\26fwo365.raw	6	6	100.000	5.397e-007
143	d:\Tran\Yaesu1\26fwo372.raw	6	6	100.000	2.587e-005
144	d:\Tran\Yaesu1\26fwo450.raw	6	6	99.9789	1.325e-027
145	d:\Tran\Yaesu1\26fwo453.raw	6	6	100.000	7.717e-005
146	d:\Tran\Yaesu1\26fwo483.raw	6	6	100.000	2.018e-009
147	d:\Tran\Yaesu1\26fwo505.raw	6	6	100.000	6.673e-006
148	d:\Tran\Yaesu1\26fwo522.raw	6	6	100.000	6.297e-009
149	d:\Tran\Yaesu1\26fwo532.raw	6	6	100.000	3.088e-004
150	d:\Tran\Yaesu1\26fwo544.raw	6	6	100.000	7.694e-012
151	d:\Tran\Yaesu1\26fwo551.raw	6	6	100.000	4.413e-008
152	d:\Tran\Yaesu1\26fwo570.raw	6	6	100.000	2.655e-005
153	d:\Tran\Yaesu1\26fwo573.raw	6	6	100.000	1.063e-004
154	d:\Tran\Yaesu1\26fwo585.raw	6	6	100.000	6.646e-008
155	d:\Tran\Yaesu1\26fwo003.raw	6	6	100.000	3.462e-012
156	d:\Tran\Yaesu2\26fwo1132.raw	7	7	100.000	3.910e-006
157	d:\Tran\Yaesu2\26fwo1140.raw	7	7	100.000	2.048e-020
158	d:\Tran\Yaesu2\26fwo1153.raw	7	7	100.000	1.861e-004

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
159	d:\Tran\Yaesu2\26fwl190.raw	7	7	100.000	7.130e-005
160	d:\Tran\Yaesu2\26fwl194.raw	7	7	100.000	3.753e-009
161	d:\Tran\Yaesu2\26fwl202.raw	7	7	100.000	8.574e-004
162	d:\Tran\Yaesu2\26fwl220.raw	7	7	100.000	1.797e-007
163	d:\Tran\Yaesu2\26fwl233.raw	7	7	100.000	3.379e-022
164	d:\Tran\Yaesu2\26fwl241.raw	7	7	100.000	4.476e-005
165	d:\Tran\Yaesu2\26fwl245.raw	7	7	100.000	1.681e-006
166	d:\Tran\Yaesu2\26fwl254.raw	7	7	100.000	1.636e-005
167	d:\Tran\Yaesu2\26fwl262.raw	7	7	100.000	2.424e-008
168	d:\Tran\Yaesu2\26fwl270.raw	7	7	100.000	1.489e-010
169	d:\Tran\Yaesu2\26fwl274.raw	7	7	100.000	4.781e-006
170	d:\Tran\Yaesu2\26fwl281.raw	7	7	100.000	2.541e-014
171	d:\Tran\Yaesu2\26fwl285.raw	7	7	100.000	2.452e-005
172	d:\Tran\Yaesu2\26fwl300.raw	7	7	100.000	1.816e-017
173	d:\Tran\Yaesu2\26fwl310.raw	7	7	100.000	3.508e-005
174	d:\Tran\Yaesu2\26fwl314.raw	7	7	100.000	1.585e-004
175	d:\Tran\Yaesu2\26fwl332.raw	7	7	100.000	1.751e-006

Number of Correct Classifications: 168/175.

TEST FOR PNN'S REJECTION ABILITY

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
1	d:\Tran\force1\26fwo415.raw	0	0	97.2873	6.390e-002
2	d:\Tran\force1\26fwo422.raw	0	0	95.5935	5.578e-002
3	d:\Tran\force1\26fwo434.raw	0	0	100.000	2.726e-005
4	d:\Tran\force1\26fwo440.raw	0	0	96.9941	5.883e-002
5	d:\Tran\force1\26fwo443.raw	0	0	96.7361	5.998e-002
6	d:\Tran\force1\26fwo445.raw	0	0	98.8080	6.752e-002
7	d:\Tran\force1\26fwo451.raw	0	0	100.000	2.625e-004
8	d:\Tran\force1\26fwo453.raw	0	1	58.5814	1.340e-005
9	d:\Tran\force1\26fwo455.raw	0	0	97.1530	3.269e-002
10	d:\Tran\force1\26fwo461.raw	0	0	98.1724	4.584e-002
11	d:\Tran\force1\26fwo463.raw	0	0	100.000	2.103e-003
12	d:\Tran\force1\26fwo465.raw	0	0	100.000	7.367e-004
13	d:\Tran\force1\26fwo475.raw	0	0	100.000	1.939e-006
14	d:\Tran\force1\26fwo481.raw	0	0	99.5820	4.343e-002
15	d:\Tran\force1\26fwo484.raw	0	0	100.000	2.847e-004
16	d:\Tran\force1\26fwo492.raw	0	0	100.000	1.647e-004
17	d:\Tran\force1\26fwo494.raw	0	0	99.9885	1.733e-003
18	d:\Tran\force1\26fwo500.raw	0	0	100.000	3.210e-004
19	d:\Tran\force1\26fwo502.raw	0	0	99.3589	1.919e-002
20	d:\Tran\force1\26fwo504.raw	0	0	98.6434	3.727e-002
21	d:\Tran\force1\26fwo513.raw	0	0	99.0354	6.036e-002
22	d:\Tran\force1\26fwo515.raw	0	-1	0.00000	2.884e-036
23	d:\Tran\force1\26fwo521.raw	0	0	98.8719	2.806e-002
24	d:\Tran\force1\26fwo524.raw	0	0	97.7546	5.038e-002
25	d:\Tran\force1\26fwo530.raw	0	0	100.000	3.627e-008
26	d:\Tran\force1\26fwo532.raw	0	0	98.3520	4.863e-002
27	d:\Tran\force1\26fwo543.raw	0	0	99.8918	7.516e-003
28	d:\Tran\force1\26fwo545.raw	0	-1	0.00000	7.207e-030
29	d:\Tran\force1\26fwo552.raw	0	0	100.000	9.953e-004
30	d:\Tran\force1\26fwo554.raw	0	0	97.1299	4.351e-002
31	d:\Tran\force2\26fwq123.raw	1	1	93.3121	3.778e-002
32	d:\Tran\force2\26fwq125.raw	1	2	54.0112	3.442e-004
33	d:\Tran\force2\26fwq132.raw	1	1	70.6416	3.497e-006
34	d:\Tran\force2\26fwq134.raw	1	1	96.8390	3.120e-002
35	d:\Tran\force2\26fwq142.raw	1	1	51.1857	1.844e-005
36	d:\Tran\force2\26fwq144.raw	1	1	77.4113	2.852e-002
37	d:\Tran\force2\26fwq150.raw	1	1	96.6430	1.055e-002
38	d:\Tran\force2\26fwq152.raw	1	1	98.2924	2.166e-002
39	d:\Tran\force2\26fwq154.raw	1	1	98.4920	1.826e-002
40	d:\Tran\force2\26fwq160.raw	1	1	96.2033	5.470e-006
41	d:\Tran\force2\26fwq162.raw	1	1	89.3796	5.582e-003
42	d:\Tran\force2\26fwq173.raw	1	1	87.7313	2.139e-003
43	d:\Tran\force2\26fwq182.raw	1	1	87.0111	1.752e-003
44	d:\Tran\force2\26fwq184.raw	1	1	97.0366	3.147e-002
45	d:\Tran\force2\26fwq193.raw	1	1	98.8504	5.059e-002
46	d:\Tran\force2\26fwq202.raw	1	1	60.0928	4.956e-006
47	d:\Tran\force2\26fwq230.raw	1	1	65.3669	3.371e-002
48	d:\Tran\force2\26fwq235.raw	1	1	87.3896	1.260e-003
49	d:\Tran\force2\26fwq243.raw	1	1	95.0479	6.357e-002
50	d:\Tran\force2\26fwq253.raw	1	1	99.2321	1.417e-003

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
51	d:\Tran\force2\26fwq255.raw	1	1	98.2943	2.621e-002
52	d:\Tran\force2\26fwq262.raw	1	1	75.7617	5.551e-004
53	d:\Tran\force2\26fwq264.raw	1	1	79.4855	5.900e-004
54	d:\Tran\force2\26fwq265.raw	1	1	99.6965	3.065e-003
55	d:\Tran\force2\26fwq271.raw	1	1	93.9386	9.105e-003
56	d:\Tran\force2\26fwq273.raw	1	1	76.4012	2.234e-003
57	d:\Tran\force2\26fwq284.raw	1	1	88.6393	4.864e-002
58	d:\Tran\force2\26fwq290.raw	1	1	94.9135	1.367e-003
59	d:\Tran\force2\26fwq314.raw	1	1	75.2602	4.997e-003
60	d:\Tran\force2\26fwq322.raw	1	0	99.9283	1.220e-006
61	d:\Tran\force2\26fwq324.raw	1	1	66.2036	5.471e-006
62	d:\Tran\force2\26fwq330.raw	1	1	97.5370	7.262e-005
63	d:\Tran\force3\27fwi291.raw	2	2	99.8463	4.266e-002
64	d:\Tran\force3\27fwi311.raw	2	2	97.7073	3.634e-002
65	d:\Tran\force3\27fwi313.raw	2	2	98.1803	1.454e-002
66	d:\Tran\force3\27fwi315.raw	2	2	99.8461	2.942e-002
67	d:\Tran\force3\27fwi321.raw	2	2	54.6509	6.991e-003
68	d:\Tran\force3\27fwi324.raw	2	2	97.2979	3.812e-002
69	d:\Tran\force3\27fwi330.raw	2	2	99.6142	8.474e-003
70	d:\Tran\force3\27fwi332.raw	2	2	69.7389	4.440e-002
71	d:\Tran\force3\27fwi333.raw	2	2	99.3601	5.405e-003
72	d:\Tran\force3\27fwi335.raw	2	2	99.8677	2.999e-002
73	d:\Tran\force3\27fwi341.raw	2	2	97.7036	4.548e-003
74	d:\Tran\force3\27fwi344.raw	2	2	51.7501	2.223e-002
75	d:\Tran\force3\27fwi353.raw	2	2	83.0433	4.389e-002
76	d:\Tran\force3\27fwi355.raw	2	1	58.0935	2.607e-002
77	d:\Tran\force3\27fwi361.raw	2	2	94.3323	6.679e-002
78	d:\Tran\force3\27fwi381.raw	2	2	80.5267	1.506e-004
79	d:\Tran\force3\27fwi383.raw	2	2	98.6914	1.447e-001
80	d:\Tran\force3\27fwi394.raw	2	1	88.2457	3.233e-004
81	d:\Tran\force3\27fwi423.raw	2	2	85.7706	7.114e-003
82	d:\Tran\force3\27fwi425.raw	2	2	97.0159	6.487e-005
83	d:\Tran\force3\27fwi431.raw	2	2	86.1520	4.534e-003
84	d:\Tran\force3\27fwi433.raw	2	2	94.3590	7.928e-004
85	d:\Tran\force3\27fwi444.raw	2	2	99.2675	3.127e-003
86	d:\Tran\force3\27fwi450.raw	2	2	59.6836	1.782e-005
87	d:\Tran\force3\27fwi452.raw	2	2	99.0882	8.558e-002
88	d:\Tran\force3\27fwi454.raw	2	2	81.3634	3.207e-002
89	d:\Tran\force3\27fwi460.raw	2	2	97.7939	2.551e-002
90	d:\Tran\force3\27fwi461.raw	2	2	89.6576	5.091e-003
91	d:\Tran\force3\27fwi463.raw	2	2	99.6656	4.444e-002
92	d:\Tran\force3\27fwi465.raw	2	2	98.9951	1.199e-001
93	d:\Tran\force3\27fwi471.raw	2	2	88.7906	2.077e-002
94	d:\Tran\force3\27fwi473.raw	2	2	57.8325	1.340e-002
95	d:\Tran\ken1\25fwn380.raw	3	3	99.8260	1.271e-010
96	d:\Tran\ken1\25fwn382.raw	3	3	100.000	7.938e-007
97	d:\Tran\ken1\25fwn384.raw	3	3	100.000	1.387e-009
98	d:\Tran\ken1\25fwn392.raw	3	3	100.000	6.136e-009
99	d:\Tran\ken1\25fwn393.raw	3	3	100.000	2.035e-006
100	d:\Tran\ken1\25fwn395.raw	3	3	100.000	2.475e-004
101	d:\Tran\ken1\25fwn401.raw	3	3	100.000	1.306e-009
102	d:\Tran\ken1\25fwn402.raw	3	3	100.000	2.878e-012
103	d:\Tran\ken1\25fwn404.raw	3	3	100.000	1.042e-010
104	d:\Tran\ken1\25fwn405.raw	3	3	99.9929	5.100e-008

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
105	d:\Tran\ken1\25fwn411.raw	3	3	100.000	4.661e-008
106	d:\Tran\ken1\25fwn413.raw	3	3	100.000	3.171e-005
107	d:\Tran\ken1\25fwn414.raw	3	3	100.000	1.012e-007
108	d:\Tran\ken1\25fwn415.raw	3	3	100.000	3.216e-012
109	d:\Tran\ken1\25fwn421.raw	3	3	99.9997	9.522e-007
110	d:\Tran\ken1\25fwn423.raw	3	3	100.000	8.245e-004
111	d:\Tran\ken1\25fwn424.raw	3	3	100.000	7.094e-007
112	d:\Tran\ken1\25fwn430.raw	3	5	100.000	2.451e-009
113	d:\Tran\ken1\25fwn432.raw	3	3	100.000	1.430e-004
114	d:\Tran\ken1\25fwn433.raw	3	3	100.000	8.040e-006
115	d:\Tran\ken1\25fwn435.raw	3	3	100.000	1.453e-007
116	d:\Tran\ken1\25fwn441.raw	3	3	100.000	2.017e-006
117	d:\Tran\ken1\25fwn443.raw	3	3	100.000	1.091e-004
118	d:\Tran\ken1\25fwn445.raw	3	3	100.000	7.207e-005
119	d:\Tran\ken1\25fwn450.raw	3	3	100.000	2.460e-003
120	d:\Tran\ken1\25fwn452.raw	3	3	100.000	4.550e-005
121	d:\Tran\ken1\25fwn453.raw	3	3	83.1878	3.738e-018
122	d:\Tran\ken1\25fwn455.raw	3	3	100.000	5.239e-006
123	d:\Tran\ken1\25fwn461.raw	3	3	100.000	1.104e-005
124	d:\Tran\ken1\25fwn462.raw	3	3	82.7030	2.160e-015
125	d:\Tran\ken2\25fwn583.raw	4	4	100.000	3.146e-003
126	d:\Tran\ken2\25fwn584.raw	4	4	100.000	4.617e-002
127	d:\Tran\ken2\25fwn593.raw	4	4	100.000	3.622e-003
128	d:\Tran\ken2\25fwn594.raw	4	4	100.000	1.752e-003
129	d:\Tran\ken2\25fwo000.raw	4	4	100.000	2.115e-002
130	d:\Tran\ken2\25fwo002.raw	4	4	100.000	5.960e-002
131	d:\Tran\ken2\25fwo003.raw	4	4	100.000	5.428e-002
132	d:\Tran\ken2\25fwo005.raw	4	4	100.000	3.457e-003
133	d:\Tran\ken2\25fwo010.raw	4	4	100.000	2.831e-007
134	d:\Tran\ken2\25fwo015.raw	4	4	100.000	4.680e-006
135	d:\Tran\ken2\25fwo021.raw	4	4	100.000	8.580e-002
136	d:\Tran\ken2\25fwo022.raw	4	4	100.000	1.391e-006
137	d:\Tran\ken2\25fwo024.raw	4	4	100.000	4.568e-002
138	d:\Tran\ken2\25fwo030.raw	4	4	100.000	2.712e-003
139	d:\Tran\ken2\25fwo031.raw	4	4	100.000	1.289e-015
140	d:\Tran\ken2\25fwo033.raw	4	4	100.000	8.537e-002
141	d:\Tran\ken2\25fwo034.raw	4	4	100.000	1.999e-004
142	d:\Tran\ken2\25fwo042.raw	4	4	100.000	7.504e-014
143	d:\Tran\ken2\25fwo044.raw	4	4	100.000	1.169e-003
144	d:\Tran\ken2\25fwo045.raw	4	4	100.000	5.509e-002
145	d:\Tran\ken2\25fwo054.raw	4	4	100.000	1.725e-003
146	d:\Tran\ken2\25fwo062.raw	4	4	100.000	5.056e-002
147	d:\Tran\ken2\25fwo064.raw	4	4	100.000	3.351e-002
148	d:\Tran\ken2\25fwo065.raw	4	4	100.000	6.374e-003
149	d:\Tran\ken2\25fwo074.raw	4	4	100.000	5.117e-003
150	d:\Tran\ken2\25fwo075.raw	4	4	100.000	2.182e-003
151	d:\Tran\ken2\25fwo080.raw	4	4	100.000	3.618e-004
152	d:\Tran\ken2\25fwo082.raw	4	4	100.000	1.970e-002
153	d:\Tran\ken2\25fwo083.raw	4	4	100.000	3.212e-002
154	d:\Tran\ken2\25fwo085.raw	4	4	100.000	2.346e-003
155	d:\Tran\ken2\25fwo090.raw	4	4	100.000	4.637e-002
156	d:\Tran\ken3\25fwo225.raw	5	5	100.000	4.109e-005
157	d:\Tran\ken3\25fwo231.raw	5	5	100.000	6.329e-006
158	d:\Tran\ken3\25fwo232.raw	5	5	100.000	1.044e-010

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
159	d:\Tran\ken3\25fwo234.raw	5	5	100.000	2.348e-003
160	d:\Tran\ken3\25fwo235.raw	5	5	100.000	5.380e-015
161	d:\Tran\ken3\25fwo241.raw	5	5	100.000	6.454e-005
162	d:\Tran\ken3\25fwo251.raw	5	-1	0.00000	8.898e-021
163	d:\Tran\ken3\25fwo253.raw	5	5	100.000	1.643e-006
164	d:\Tran\ken3\25fwo260.raw	5	5	100.000	1.339e-002
165	d:\Tran\ken3\25fwo265.raw	5	5	100.000	1.567e-004
166	d:\Tran\ken3\25fwo271.raw	5	5	100.000	4.714e-004
167	d:\Tran\ken3\25fwo282.raw	5	5	100.000	1.254e-004
168	d:\Tran\ken3\25fwo283.raw	5	5	100.000	3.994e-005
169	d:\Tran\ken3\25fwo301.raw	5	5	100.000	1.094e-003
170	d:\Tran\ken3\25fwo303.raw	5	5	100.000	8.928e-004
171	d:\Tran\ken3\25fwo305.raw	5	3	100.000	3.149e-008
172	d:\Tran\ken3\25fwo312.raw	5	3	99.9924	3.220e-008
173	d:\Tran\ken3\25fwo314.raw	5	5	100.000	1.604e-004
174	d:\Tran\ken3\25fwo315.raw	5	5	100.000	1.703e-003
175	d:\Tran\ken3\25fwo321.raw	5	5	100.000	2.563e-004
176	d:\Tran\ken3\25fwo325.raw	5	5	99.9991	1.354e-005
177	d:\Tran\ken3\25fwo332.raw	5	5	99.9992	2.477e-005
178	d:\Tran\ken3\25fwo334.raw	5	5	100.000	3.637e-005
179	d:\Tran\ken3\25fwo335.raw	5	5	100.000	6.532e-012
180	d:\Tran\ken3\25fwo341.raw	5	5	100.000	7.903e-004
181	d:\Tran\ken3\25fwo342.raw	5	5	100.000	2.614e-003
182	d:\Tran\ken3\25fwo344.raw	5	5	100.000	1.931e-004
183	d:\Tran\ken3\25fwo352.raw	5	5	98.8981	1.840e-009
184	d:\Tran\ken3\25fwo354.raw	5	5	99.9997	5.319e-014
185	d:\Tran\ken3\25fwp063.raw	5	5	100.000	2.300e-015
186	d:\Tran\ken3\25fwp065.raw	5	5	100.000	4.353e-007
187	d:\Tran\ken3\25fwp075.raw	5	5	99.9962	4.840e-008
188	d:\Tran\Yaesu1\26fwk312.raw	6	-1	0.00000	4.745e-033
189	d:\Tran\Yaesu1\26fwk314.raw	6	-1	0.00000	1.264e-024
190	d:\Tran\Yaesu1\26fwk332.raw	6	-1	0.00000	1.725e-033
191	d:\Tran\Yaesu1\26fwk334.raw	6	-1	0.00000	1.777e-026
192	d:\Tran\Yaesu1\26fwk335.raw	6	-1	0.00000	3.503e-027
193	d:\Tran\Yaesu1\26fwk341.raw	6	-1	0.00000	2.156e-029
194	d:\Tran\Yaesu1\26fwk343.raw	6	-1	0.00000	1.179e-028
195	d:\Tran\Yaesu1\26fwk345.raw	6	-1	0.00000	1.216e-033
196	d:\Tran\Yaesu1\26fwk355.raw	6	-1	0.00000	3.234e-027
197	d:\Tran\Yaesu1\26fwk361.raw	6	-1	0.00000	4.091e-028
198	d:\Tran\Yaesu1\26fwk363.raw	6	-1	0.00000	9.295e-033
199	d:\Tran\Yaesu1\26fwk365.raw	6	-1	0.00000	1.073e-026
200	d:\Tran\Yaesu1\26fwk370.raw	6	-1	0.00000	2.815e-026
201	d:\Tran\Yaesu1\26fwk372.raw	6	-1	0.00000	7.357e-037
202	d:\Tran\Yaesu1\26fwk435.raw	6	-1	0.00000	1.065e-028
203	d:\Tran\Yaesu1\26fwk450.raw	6	-1	0.00000	1.184e-043
204	d:\Tran\Yaesu1\26fwk452.raw	6	-1	0.00000	2.560e-034
205	d:\Tran\Yaesu1\26fwk453.raw	6	-1	0.00000	1.150e-028
206	d:\Tran\Yaesu1\26fwk455.raw	6	-1	0.00000	4.663e-031
207	d:\Tran\Yaesu1\26fwk483.raw	6	-1	0.00000	5.722e-032
208	d:\Tran\Yaesu1\26fwk502.raw	6	-1	0.00000	1.291e-027
209	d:\Tran\Yaesu1\26fwk505.raw	6	-1	0.00000	8.261e-032
210	d:\Tran\Yaesu1\26fwk520.raw	6	-1	0.00000	1.259e-027
211	d:\Tran\Yaesu1\26fwk522.raw	6	-1	0.00000	3.298e-028
212	d:\Tran\Yaesu1\26fwk524.raw	6	-1	0.00000	4.081e-028

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
213	d:\Tran\Yaesu1\26fwk532.raw	6	-1	0.00000	1.536e-025
214	d:\Tran\Yaesu1\26fwk541.raw	6	-1	0.00000	3.164e-029
215	d:\Tran\Yaesu1\26fwk544.raw	6	-1	0.00000	7.466e-028
216	d:\Tran\Yaesu1\26fwk545.raw	6	-1	0.00000	1.828e-037
217	d:\Tran\Yaesu1\26fwk551.raw	6	-1	0.00000	2.432e-031
218	d:\Tran\Yaesu1\26fwk564.raw	6	-1	0.00000	1.133e-029
219	d:\Tran\Yaesu1\26fwk570.raw	6	-1	0.00000	6.907e-024
220	d:\Tran\Yaesu1\26fwk571.raw	6	-1	0.00000	2.740e-039
221	d:\Tran\Yaesu1\26fwk573.raw	6	-1	0.00000	2.608e-028
222	d:\Tran\Yaesu1\26fwk575.raw	6	-1	0.00000	3.009e-026
223	d:\Tran\Yaesu1\26fwk585.raw	6	-1	0.00000	1.814e-034
224	d:\Tran\Yaesu1\26fwl002.raw	6	-1	0.00000	3.023e-025
225	d:\Tran\Yaesu1\26fwl003.raw	6	-1	0.00000	1.388e-025
226	d:\Tran\Yaesu2\26fwl224.raw	7	-1	0.00000	2.736e-046
227	d:\Tran\Yaesu2\26fwl225.raw	7	-1	0.00000	7.766e-021
228	d:\Tran\Yaesu2\26fwl231.raw	7	-1	0.00000	2.062e-061
229	d:\Tran\Yaesu2\26fwl233.raw	7	-1	0.00000	1.811e-048
230	d:\Tran\Yaesu2\26fwl235.raw	7	-1	0.00000	2.516e-054
231	d:\Tran\Yaesu2\26fwl241.raw	7	-1	0.00000	2.922e-046
232	d:\Tran\Yaesu2\26fwl243.raw	7	-1	0.00000	6.960e-054
233	d:\Tran\Yaesu2\26fwl245.raw	7	-1	0.00000	4.652e-047
234	d:\Tran\Yaesu2\26fwl251.raw	7	-1	0.00000	1.055e-074
235	d:\Tran\Yaesu2\26fwl254.raw	7	-1	0.00000	1.511e-045
236	d:\Tran\Yaesu2\26fwl260.raw	7	-1	0.00000	7.844e-069
237	d:\Tran\Yaesu2\26fwl262.raw	7	-1	0.00000	2.932e-039
238	d:\Tran\Yaesu2\26fwl264.raw	7	-1	0.00000	3.508e-056
239	d:\Tran\Yaesu2\26fwl270.raw	7	-1	0.00000	2.568e-040
240	d:\Tran\Yaesu2\26fwl272.raw	7	-1	0.00000	2.745e-051
241	d:\Tran\Yaesu2\26fwl274.raw	7	-1	0.00000	1.823e-046
242	d:\Tran\Yaesu2\26fwl280.raw	7	-1	0.00000	5.317e-056
243	d:\Tran\Yaesu2\26fwl281.raw	7	-1	0.00000	1.234e-060
244	d:\Tran\Yaesu2\26fwl283.raw	7	-1	0.00000	1.587e-044
245	d:\Tran\Yaesu2\26fwl285.raw	7	-1	0.00000	4.001e-049
246	d:\Tran\Yaesu2\26fwl293.raw	7	-1	0.00000	1.913e-061
247	d:\Tran\Yaesu2\26fwl300.raw	7	-1	0.00000	1.793e-054
248	d:\Tran\Yaesu2\26fwl304.raw	7	-1	0.00000	1.684e-059
249	d:\Tran\Yaesu2\26fwl310.raw	7	-1	0.00000	5.574e-050
250	d:\Tran\Yaesu2\26fwl312.raw	7	-1	0.00000	6.915e-072
251	d:\Tran\Yaesu2\26fwl314.raw	7	-1	0.00000	1.087e-052
252	d:\Tran\Yaesu2\26fwl320.raw	7	-1	0.00000	9.380e-040
253	d:\Tran\Yaesu2\26fwl322.raw	7	-1	0.00000	1.222e-059
254	d:\Tran\Yaesu2\26fwl330.raw	7	-1	0.00000	1.039e-059
255	d:\Tran\Yaesu2\26fwl332.raw	7	-1	0.00000	7.277e-052

Number of Correct Classifications: 176/255.

VALIDATION SET #1 WITH MULTIMODAL SEGMENTATION

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
1	d:\Tran\force1\26fwo415.raw	0	0	97.3767	6.296e-002
2	d:\Tran\force1\26fwo422.raw	0	0	96.0108	5.478e-002
3	d:\Tran\force1\26fwo434.raw	0	0	100.000	2.600e-007
4	d:\Tran\force1\26fwo440.raw	0	0	97.2889	5.797e-002
5	d:\Tran\force1\26fwo443.raw	0	0	97.3442	6.000e-002
6	d:\Tran\force1\26fwo445.raw	0	0	98.9565	6.550e-002
7	d:\Tran\force1\26fwo451.raw	0	0	99.8978	5.856e-003
8	d:\Tran\force1\26fwo453.raw	0	0	99.7483	1.737e-002
9	d:\Tran\force1\26fwo455.raw	0	0	98.8306	8.859e-003
10	d:\Tran\force1\26fwo461.raw	0	0	98.3512	4.273e-002
11	d:\Tran\force1\26fwo463.raw	0	0	100.000	2.181e-002
12	d:\Tran\force1\26fwo465.raw	0	0	99.9954	1.236e-003
13	d:\Tran\force1\26fwo475.raw	0	0	100.000	5.122e-005
14	d:\Tran\force1\26fwo481.raw	0	0	99.9331	1.728e-002
15	d:\Tran\force1\26fwo484.raw	0	0	99.9903	1.399e-003
16	d:\Tran\force1\26fwo492.raw	0	0	99.9977	7.281e-004
17	d:\Tran\force1\26fwo494.raw	0	0	99.9836	6.092e-003
18	d:\Tran\force1\26fwo500.raw	0	0	99.9992	1.174e-004
19	d:\Tran\force1\26fwo502.raw	0	0	99.4314	1.816e-002
20	d:\Tran\force1\26fwo504.raw	0	0	98.7598	3.660e-002
21	d:\Tran\force1\26fwo513.raw	0	0	99.1415	5.873e-002
22	d:\Tran\force1\26fwo515.raw	0	6	99.9893	7.327e-017
23	d:\Tran\force1\26fwo521.raw	0	0	98.9977	2.659e-002
24	d:\Tran\force1\26fwo524.raw	0	0	97.9924	5.179e-002
25	d:\Tran\force1\26fwo530.raw	0	0	99.9970	5.258e-006
26	d:\Tran\force1\26fwo532.raw	0	0	99.8340	4.642e-002
27	d:\Tran\force1\26fwo543.raw	0	0	99.8274	1.337e-002
28	d:\Tran\force1\26fwo545.raw	0	0	99.9140	1.358e-010
29	d:\Tran\force1\26fwo552.raw	0	0	99.7287	1.341e-002
30	d:\Tran\force1\26fwo554.raw	0	0	97.4367	4.223e-002
31	d:\Tran\force2\26fwq123.raw	1	1	93.7172	3.452e-002
32	d:\Tran\force2\26fwq125.raw	1	1	82.8185	1.038e-002
33	d:\Tran\force2\26fwq132.raw	1	1	88.6469	1.166e-003
34	d:\Tran\force2\26fwq134.raw	1	1	96.0653	4.338e-003
35	d:\Tran\force2\26fwq142.raw	1	1	81.8422	6.866e-003
36	d:\Tran\force2\26fwq144.raw	1	1	77.9331	2.684e-002
37	d:\Tran\force2\26fwq150.raw	1	1	98.9649	2.353e-002
38	d:\Tran\force2\26fwq152.raw	1	1	98.4948	2.084e-002
39	d:\Tran\force2\26fwq154.raw	1	1	98.6893	1.768e-002
40	d:\Tran\force2\26fwq160.raw	1	1	82.8846	2.533e-003
41	d:\Tran\force2\26fwq162.raw	1	1	90.0572	5.163e-003
42	d:\Tran\force2\26fwq173.raw	1	1	98.5623	3.856e-003
43	d:\Tran\force2\26fwq182.raw	1	1	97.6177	9.352e-003
44	d:\Tran\force2\26fwq184.raw	1	1	97.6320	2.677e-002
45	d:\Tran\force2\26fwq193.raw	1	1	99.4364	5.570e-002
46	d:\Tran\force2\26fwq202.raw	1	1	95.4127	4.960e-003
47	d:\Tran\force2\26fwq230.raw	1	1	65.6436	3.219e-002
48	d:\Tran\force2\26fwq235.raw	1	1	86.1831	3.221e-003
49	d:\Tran\force2\26fwq243.raw	1	1	96.2262	5.965e-002
50	d:\Tran\force2\26fwq253.raw	1	1	99.3206	1.282e-003

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
51	d:\Tran\force2\26fwq255.raw	1	1	99.0819	8.564e-003
52	d:\Tran\force2\26fwq262.raw	1	1	98.1633	6.225e-003
53	d:\Tran\force2\26fwq264.raw	1	1	50.3255	8.168e-004
54	d:\Tran\force2\26fwq265.raw	1	1	96.1280	3.644e-003
55	d:\Tran\force2\26fwq271.raw	1	1	94.6777	7.684e-003
56	d:\Tran\force2\26fwq273.raw	1	1	76.9584	2.017e-003
57	d:\Tran\force2\26fwq284.raw	1	1	89.1198	4.653e-002
58	d:\Tran\force2\26fwq290.raw	1	1	94.4642	1.153e-002
59	d:\Tran\force2\26fwq314.raw	1	1	75.6900	4.478e-003
60	d:\Tran\force2\26fwq322.raw	1	2	87.3339	3.292e-003
61	d:\Tran\force2\26fwq324.raw	1	1	99.2370	3.794e-004
62	d:\Tran\force2\26fwq330.raw	1	1	83.8004	8.394e-003
63	d:\Tran\force3\27fwi291.raw	2	2	99.8735	4.155e-002
64	d:\Tran\force3\27fwi311.raw	2	2	96.7625	2.750e-002
65	d:\Tran\force3\27fwi313.raw	2	2	98.3528	1.354e-002
66	d:\Tran\force3\27fwi315.raw	2	2	99.4034	4.488e-003
67	d:\Tran\force3\27fwi321.raw	2	2	99.0730	6.765e-003
68	d:\Tran\force3\27fwi324.raw	2	2	81.8157	1.404e-002
69	d:\Tran\force3\27fwi330.raw	2	2	99.6683	7.879e-003
70	d:\Tran\force3\27fwi332.raw	2	2	70.3202	4.262e-002
71	d:\Tran\force3\27fwi333.raw	2	2	99.5621	7.059e-003
72	d:\Tran\force3\27fwi335.raw	2	2	99.8930	2.997e-002
73	d:\Tran\force3\27fwi341.raw	2	2	98.3687	6.573e-003
74	d:\Tran\force3\27fwi344.raw	2	2	96.9713	3.251e-002
75	d:\Tran\force3\27fwi353.raw	2	2	83.4205	4.319e-002
76	d:\Tran\force3\27fwi355.raw	2	2	95.9594	2.845e-002
77	d:\Tran\force3\27fwi361.raw	2	2	94.9569	6.726e-002
78	d:\Tran\force3\27fwi381.raw	2	2	99.6616	1.352e-002
79	d:\Tran\force3\27fwi383.raw	2	2	98.8289	1.424e-001
80	d:\Tran\force3\27fwi394.raw	2	2	99.5529	1.514e-002
81	d:\Tran\force3\27fwi423.raw	2	2	86.4589	6.577e-003
82	d:\Tran\force3\27fwi425.raw	2	2	96.3532	7.307e-004
83	d:\Tran\force3\27fwi431.raw	2	1	67.7084	6.738e-003
84	d:\Tran\force3\27fwi433.raw	2	2	94.7765	7.152e-004
85	d:\Tran\force3\27fwi444.raw	2	2	99.8101	8.671e-003
86	d:\Tran\force3\27fwi450.raw	2	2	69.1159	1.224e-002
87	d:\Tran\force3\27fwi452.raw	2	2	99.1982	8.340e-002
88	d:\Tran\force3\27fwi454.raw	2	2	82.3168	3.107e-002
89	d:\Tran\force3\27fwi460.raw	2	2	98.2550	2.462e-002
90	d:\Tran\force3\27fwi461.raw	2	2	90.2984	4.493e-003
91	d:\Tran\force3\27fwi463.raw	2	2	99.7187	4.323e-002
92	d:\Tran\force3\27fwi465.raw	2	2	99.1123	1.175e-001
93	d:\Tran\force3\27fwi471.raw	2	2	89.2641	2.003e-002
94	d:\Tran\force3\27fwi473.raw	2	2	65.3848	2.902e-002
95	d:\Tran\ken1\25fwn380.raw	3	3	100.000	1.140e-005
96	d:\Tran\ken1\25fwn382.raw	3	3	100.000	2.355e-003
97	d:\Tran\ken1\25fwn384.raw	3	3	100.000	1.057e-003
98	d:\Tran\ken1\25fwn392.raw	3	3	100.000	5.809e-009
99	d:\Tran\ken1\25fwn393.raw	3	3	100.000	3.671e-003
100	d:\Tran\ken1\25fwn395.raw	3	3	100.000	2.920e-004
101	d:\Tran\ken1\25fwn401.raw	3	3	99.9995	1.872e-007
102	d:\Tran\ken1\25fwn402.raw	3	3	100.000	2.090e-005
103	d:\Tran\ken1\25fwn404.raw	3	3	100.000	6.860e-005
104	d:\Tran\ken1\25fwn405.raw	3	3	99.9996	1.077e-005

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
105	d:\Tran\ken1\25fwn411.raw	3	3	100.000	1.999e-003
106	d:\Tran\ken1\25fwn413.raw	3	3	100.000	2.882e-005
107	d:\Tran\ken1\25fwn414.raw	3	3	100.000	2.142e-007
108	d:\Tran\ken1\25fwn415.raw	3	3	100.000	7.290e-006
109	d:\Tran\ken1\25fwn421.raw	3	3	100.000	5.186e-003
110	d:\Tran\ken1\25fwn423.raw	3	3	100.000	2.081e-003
111	d:\Tran\ken1\25fwn424.raw	3	3	100.000	1.916e-004
112	d:\Tran\ken1\25fwn430.raw	3	3	100.000	2.460e-006
113	d:\Tran\ken1\25fwn432.raw	3	3	100.000	7.605e-004
114	d:\Tran\ken1\25fwn433.raw	3	3	100.000	7.346e-006
115	d:\Tran\ken1\25fwn435.raw	3	3	100.000	4.811e-005
116	d:\Tran\ken1\25fwn441.raw	3	3	100.000	1.297e-003
117	d:\Tran\ken1\25fwn443.raw	3	3	100.000	9.731e-009
118	d:\Tran\ken1\25fwn445.raw	3	3	100.000	1.503e-003
119	d:\Tran\ken1\25fwn450.raw	3	3	100.000	2.189e-003
120	d:\Tran\ken1\25fwn452.raw	3	3	100.000	9.778e-005
121	d:\Tran\ken1\25fwn453.raw	3	5	78.6396	1.037e-008
122	d:\Tran\ken1\25fwn455.raw	3	3	100.000	1.351e-006
123	d:\Tran\ken1\25fwn461.raw	3	3	100.000	3.189e-006
124	d:\Tran\ken1\25fwn462.raw	3	4	100.000	7.799e-007
125	d:\Tran\ken2\25fwn583.raw	4	4	100.000	4.768e-003
126	d:\Tran\ken2\25fwn584.raw	4	4	100.000	4.433e-002
127	d:\Tran\ken2\25fwn593.raw	4	4	100.000	1.719e-002
128	d:\Tran\ken2\25fwn594.raw	4	4	100.000	5.190e-002
129	d:\Tran\ken2\25fwo000.raw	4	4	100.000	4.226e-002
130	d:\Tran\ken2\25fwo002.raw	4	4	100.000	5.250e-002
131	d:\Tran\ken2\25fwo003.raw	4	4	100.000	4.872e-002
132	d:\Tran\ken2\25fwo005.raw	4	4	100.000	2.134e-002
133	d:\Tran\ken2\25fwo010.raw	4	4	100.000	6.513e-007
134	d:\Tran\ken2\25fwo015.raw	4	4	100.000	3.619e-006
135	d:\Tran\ken2\25fwo021.raw	4	4	100.000	7.970e-002
136	d:\Tran\ken2\25fwo022.raw	4	4	100.000	5.642e-003
137	d:\Tran\ken2\25fwo024.raw	4	4	100.000	1.934e-002
138	d:\Tran\ken2\25fwo030.raw	4	4	100.000	4.825e-002
139	d:\Tran\ken2\25fwo031.raw	4	4	100.000	9.537e-004
140	d:\Tran\ken2\25fwo033.raw	4	4	100.000	8.424e-002
141	d:\Tran\ken2\25fwo034.raw	4	4	100.000	1.074e-002
142	d:\Tran\ken2\25fwo042.raw	4	4	100.000	1.122e-002
143	d:\Tran\ken2\25fwo044.raw	4	4	100.000	5.986e-003
144	d:\Tran\ken2\25fwo045.raw	4	4	100.000	5.298e-002
145	d:\Tran\ken2\25fwo054.raw	4	4	100.000	5.293e-002
146	d:\Tran\ken2\25fwo062.raw	4	4	100.000	2.840e-002
147	d:\Tran\ken2\25fwo064.raw	4	4	100.000	7.028e-002
148	d:\Tran\ken2\25fwo065.raw	4	4	100.000	6.668e-003
149	d:\Tran\ken2\25fwo074.raw	4	4	100.000	1.355e-002
150	d:\Tran\ken2\25fwo075.raw	4	4	100.000	2.032e-002
151	d:\Tran\ken2\25fwo080.raw	4	4	100.000	7.335e-004
152	d:\Tran\ken2\25fwo082.raw	4	4	100.000	1.895e-002
153	d:\Tran\ken2\25fwo083.raw	4	4	100.000	5.624e-002
154	d:\Tran\ken2\25fwo085.raw	4	4	100.000	2.523e-003
155	d:\Tran\ken2\25fwo090.raw	4	4	100.000	4.893e-002
156	d:\Tran\ken3\25fwo225.raw	5	5	100.000	1.291e-004
157	d:\Tran\ken3\25fwo231.raw	5	5	100.000	1.456e-004
158	d:\Tran\ken3\25fwo232.raw	5	5	100.000	6.202e-011

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
159	d:\Tran\ken3\25fwo234.raw	5	5	100.000	2.079e-002
160	d:\Tran\ken3\25fwo235.raw	5	5	99.9937	5.218e-020
161	d:\Tran\ken3\25fwo241.raw	5	5	100.000	5.176e-005
162	d:\Tran\ken3\25fwo251.raw	5	3	100.000	2.854e-016
163	d:\Tran\ken3\25fwo253.raw	5	5	100.000	2.430e-006
164	d:\Tran\ken3\25fwo260.raw	5	5	100.000	1.246e-002
165	d:\Tran\ken3\25fwo265.raw	5	5	100.000	1.328e-004
166	d:\Tran\ken3\25fwo271.raw	5	5	100.000	3.402e-004
167	d:\Tran\ken3\25fwo282.raw	5	5	100.000	2.273e-004
168	d:\Tran\ken3\25fwo283.raw	5	5	100.000	8.488e-003
169	d:\Tran\ken3\25fwo301.raw	5	5	100.000	8.653e-004
170	d:\Tran\ken3\25fwo303.raw	5	5	100.000	8.279e-004
171	d:\Tran\ken3\25fwo305.raw	5	3	100.000	2.348e-007
172	d:\Tran\ken3\25fwo312.raw	5	5	100.000	2.785e-005
173	d:\Tran\ken3\25fwo314.raw	5	5	100.000	1.374e-004
174	d:\Tran\ken3\25fwo315.raw	5	5	100.000	9.413e-004
175	d:\Tran\ken3\25fwo321.raw	5	5	100.000	2.392e-004
176	d:\Tran\ken3\25fwo325.raw	5	5	99.9992	9.716e-006
177	d:\Tran\ken3\25fwo332.raw	5	5	99.9993	1.885e-005
178	d:\Tran\ken3\25fwo334.raw	5	5	100.000	3.397e-005
179	d:\Tran\ken3\25fwo335.raw	5	5	100.000	1.715e-012
180	d:\Tran\ken3\25fwo341.raw	5	5	100.000	7.040e-004
181	d:\Tran\ken3\25fwo342.raw	5	5	100.000	2.313e-003
182	d:\Tran\ken3\25fwo344.raw	5	5	100.000	1.407e-004
183	d:\Tran\ken3\25fwo352.raw	5	5	100.000	5.752e-004
184	d:\Tran\ken3\25fwo354.raw	5	3	99.8439	1.475e-012
185	d:\Tran\ken3\25fwp063.raw	5	5	100.000	8.276e-016
186	d:\Tran\ken3\25fwp065.raw	5	5	100.000	4.503e-006
187	d:\Tran\ken3\25fwp075.raw	5	5	99.9971	3.413e-008
188	d:\Tran\Yaesu1\26fwk312.raw	6	6	99.9996	2.419e-003
189	d:\Tran\Yaesu1\26fwk314.raw	6	6	100.000	9.506e-004
190	d:\Tran\Yaesu1\26fwk332.raw	6	6	100.000	1.397e-003
191	d:\Tran\Yaesu1\26fwk334.raw	6	6	100.000	3.293e-004
192	d:\Tran\Yaesu1\26fwk335.raw	6	6	100.000	1.866e-003
193	d:\Tran\Yaesu1\26fwk341.raw	6	6	100.000	5.908e-006
194	d:\Tran\Yaesu1\26fwk343.raw	6	6	100.000	5.420e-004
195	d:\Tran\Yaesu1\26fwk345.raw	6	6	99.9992	1.262e-002
196	d:\Tran\Yaesu1\26fwk355.raw	6	6	100.000	7.578e-004
197	d:\Tran\Yaesu1\26fwk361.raw	6	6	100.000	1.938e-003
198	d:\Tran\Yaesu1\26fwk363.raw	6	6	100.000	3.418e-004
199	d:\Tran\Yaesu1\26fwk365.raw	6	6	100.000	3.747e-005
200	d:\Tran\Yaesu1\26fwk370.raw	6	6	100.000	2.380e-003
201	d:\Tran\Yaesu1\26fwk372.raw	6	6	99.9835	1.045e-003
202	d:\Tran\Yaesu1\26fwk435.raw	6	6	100.000	1.287e-002
203	d:\Tran\Yaesu1\26fwk450.raw	6	6	99.9988	5.514e-010
204	d:\Tran\Yaesu1\26fwk452.raw	6	6	100.000	5.899e-003
205	d:\Tran\Yaesu1\26fwk453.raw	6	6	100.000	9.204e-003
206	d:\Tran\Yaesu1\26fwk455.raw	6	6	100.000	5.436e-003
207	d:\Tran\Yaesu1\26fwk483.raw	6	6	100.000	9.350e-004
208	d:\Tran\Yaesu1\26fwk502.raw	6	6	100.000	9.264e-004
209	d:\Tran\Yaesu1\26fwk505.raw	6	6	100.000	4.104e-006
210	d:\Tran\Yaesu1\26fwk520.raw	6	6	100.000	7.337e-003
211	d:\Tran\Yaesu1\26fwk522.raw	6	6	100.000	7.448e-003
212	d:\Tran\Yaesu1\26fwk524.raw	6	6	100.000	5.595e-003

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
213	d:\Tran\Yaesu1\26fwk532.raw	6	6	100.000	3.249e-004
214	d:\Tran\Yaesu1\26fwk541.raw	6	6	100.000	6.848e-003
215	d:\Tran\Yaesu1\26fwk544.raw	6	6	100.000	6.648e-004
216	d:\Tran\Yaesu1\26fwk545.raw	6	6	99.9998	1.017e-002
217	d:\Tran\Yaesu1\26fwk551.raw	6	6	100.000	1.319e-002
218	d:\Tran\Yaesu1\26fwk564.raw	6	6	100.000	5.702e-003
219	d:\Tran\Yaesu1\26fwk570.raw	6	6	100.000	2.764e-005
220	d:\Tran\Yaesu1\26fwk571.raw	6	6	100.000	2.506e-002
221	d:\Tran\Yaesu1\26fwk573.raw	6	6	100.000	3.889e-003
222	d:\Tran\Yaesu1\26fwk575.raw	6	6	100.000	3.207e-003
223	d:\Tran\Yaesu1\26fwk585.raw	6	6	100.000	2.195e-002
224	d:\Tran\Yaesu1\26fwl002.raw	6	6	100.000	3.438e-006
225	d:\Tran\Yaesu1\26fwl003.raw	6	6	100.000	4.341e-005
226	d:\Tran\Yaesu2\26fwl224.raw	7	7	100.000	6.171e-007
227	d:\Tran\Yaesu2\26fwl225.raw	7	7	100.000	5.815e-010
228	d:\Tran\Yaesu2\26fwl231.raw	7	7	100.000	5.662e-010
229	d:\Tran\Yaesu2\26fwl233.raw	7	7	99.6668	2.958e-010
230	d:\Tran\Yaesu2\26fwl235.raw	7	7	100.000	5.514e-005
231	d:\Tran\Yaesu2\26fwl241.raw	7	7	100.000	4.189e-004
232	d:\Tran\Yaesu2\26fwl243.raw	7	7	100.000	2.358e-006
233	d:\Tran\Yaesu2\26fwl245.raw	7	7	100.000	3.922e-005
234	d:\Tran\Yaesu2\26fwl251.raw	7	7	100.000	9.993e-007
235	d:\Tran\Yaesu2\26fwl254.raw	7	7	99.9999	2.456e-003
236	d:\Tran\Yaesu2\26fwl260.raw	7	7	100.000	3.962e-005
237	d:\Tran\Yaesu2\26fwl262.raw	7	7	100.000	3.255e-005
238	d:\Tran\Yaesu2\26fwl264.raw	7	7	100.000	1.169e-006
239	d:\Tran\Yaesu2\26fwl270.raw	7	7	100.000	7.114e-005
240	d:\Tran\Yaesu2\26fwl272.raw	7	7	100.000	3.403e-005
241	d:\Tran\Yaesu2\26fwl274.raw	7	7	100.000	6.421e-005
242	d:\Tran\Yaesu2\26fwl280.raw	7	7	100.000	7.932e-005
243	d:\Tran\Yaesu2\26fwl281.raw	7	7	99.9965	3.115e-013
244	d:\Tran\Yaesu2\26fwl283.raw	7	7	100.000	2.839e-003
245	d:\Tran\Yaesu2\26fwl285.raw	7	7	100.000	4.474e-004
246	d:\Tran\Yaesu2\26fwl293.raw	7	7	100.000	4.207e-004
247	d:\Tran\Yaesu2\26fwl300.raw	7	7	100.000	1.363e-005
248	d:\Tran\Yaesu2\26fwl304.raw	7	7	100.000	2.337e-007
249	d:\Tran\Yaesu2\26fwl310.raw	7	7	100.000	2.435e-003
250	d:\Tran\Yaesu2\26fwl312.raw	7	7	100.000	5.845e-005
251	d:\Tran\Yaesu2\26fwl314.raw	7	7	100.000	1.397e-003
252	d:\Tran\Yaesu2\26fwl320.raw	7	7	99.9996	4.544e-007
253	d:\Tran\Yaesu2\26fwl322.raw	7	7	100.000	5.329e-009
254	d:\Tran\Yaesu2\26fwl330.raw	7	7	100.000	2.173e-007
255	d:\Tran\Yaesu2\26fwl332.raw	7	7	100.000	3.491e-005

Number of Correct Classifications: 247/255.

VALIDATION SET #2 WITH MULTIMODAL SEGMENTATION

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
1	c:\transients\force1\26fwo322.raw	0	0	99.5857	4.151e-002
2	c:\transients\force1\26fwo325.raw	0	0	99.9962	1.112e-002
3	c:\transients\force1\26fwo332.raw	0	0	97.2578	1.105e-001
4	c:\transients\force1\26fwo341.raw	0	0	99.9065	1.866e-002
5	c:\transients\force1\26fwo363.raw	0	0	99.0977	5.836e-003
6	c:\transients\force1\26fwo365.raw	0	0	67.6109	7.954e-003
7	c:\transients\force1\26fwo374.raw	0	0	99.4551	3.639e-002
8	c:\transients\force1\26fwo380.raw	0	0	99.2834	6.596e-002
9	c:\transients\force1\26fwo393.raw	0	0	93.1092	1.645e-002
10	c:\transients\force1\26fwo395.raw	0	0	99.9607	1.625e-005
11	c:\transients\force1\26fwo401.raw	0	0	65.4435	1.111e-003
12	c:\transients\force1\26fwo403.raw	0	0	100.000	2.057e-013
13	c:\transients\force1\26fwo415.raw	0	0	97.1807	1.118e-001
14	c:\transients\force1\26fwo422.raw	0	0	95.8953	1.169e-001
15	c:\transients\force1\26fwo434.raw	0	0	100.000	3.523e-006
16	c:\transients\force1\26fwo440.raw	0	0	96.5365	1.093e-001
17	c:\transients\force1\26fwo451.raw	0	0	100.000	2.296e-002
18	c:\transients\force1\26fwo453.raw	0	0	95.9810	1.541e-002
19	c:\transients\force1\26fwo455.raw	0	0	99.9935	8.852e-003
20	c:\transients\force1\26fwo461.raw	0	0	95.2333	6.317e-002
21	c:\transients\force1\26fwo475.raw	0	0	99.9930	1.397e-003
22	c:\transients\force1\26fwo481.raw	0	0	91.3049	2.420e-003
23	c:\transients\force1\26fwo484.raw	0	0	98.7398	6.416e-003
24	c:\transients\force1\26fwo492.raw	0	0	99.5594	4.047e-003
25	c:\transients\force1\26fwo502.raw	0	0	99.5622	3.829e-002
26	c:\transients\force1\26fwo504.raw	0	0	66.2496	9.445e-003
27	c:\transients\force1\26fwo513.raw	0	0	98.3593	1.154e-001
28	c:\transients\force1\26fwo515.raw	0	6	99.9874	2.433e-010
29	c:\transients\force1\26fwo530.raw	0	0	99.9365	1.356e-004
30	c:\transients\force1\26fwo532.raw	0	0	49.2454	1.043e-003
31	c:\transients\force1\26fwo543.raw	0	0	98.9033	1.957e-002
32	c:\transients\force1\26fwo545.raw	0	0	93.3338	1.456e-008
33	c:\transients\force1\26fwo562.raw	0	0	99.1118	9.686e-002
34	c:\transients\force1\26fwo390.raw	0	1	85.9869	1.198e-003
35	c:\transients\force1\26fwo413.raw	0	0	100.000	2.564e-002
36	c:\transients\force1\26fwo445.raw	0	0	95.1080	5.990e-002
37	c:\transients\force1\26fwo465.raw	0	0	99.7135	2.417e-002
38	c:\transients\force1\26fwo500.raw	0	0	99.3918	2.164e-004
39	c:\transients\force1\26fwo524.raw	0	0	96.6037	1.126e-001
40	c:\transients\force1\26fwo554.raw	0	0	96.3245	1.058e-001
41	c:\transients\force2\26fwp573.raw	1	1	87.6446	2.713e-003
42	c:\transients\force2\26fwp592.raw	1	1	75.4360	1.823e-003
43	c:\transients\force2\26fwq012.raw	1	1	82.3508	1.173e-002
44	c:\transients\force2\26fwq015.raw	1	1	97.2615	4.142e-002
45	c:\transients\force2\26fwq044.raw	1	1	85.2259	1.238e-002
46	c:\transients\force2\26fwq050.raw	1	1	94.2297	5.519e-002
47	c:\transients\force2\26fwq054.raw	1	2	53.9532	3.792e-004
48	c:\transients\force2\26fwq061.raw	1	1	99.9115	6.500e-003
49	c:\transients\force2\26fwq073.raw	1	1	87.9549	8.193e-003
50	c:\transients\force2\26fwq084.raw	1	1	96.1094	4.869e-003

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
51	c:\transients\force2\26fwq091.raw	1	1	99.9993	3.446e-002
52	c:\transients\force2\26fwq094.raw	1	1	90.3657	6.413e-002
53	c:\transients\force2\26fwq123.raw	1	1	95.9417	3.307e-002
54	c:\transients\force2\26fwq125.raw	1	2	57.8419	8.710e-003
55	c:\transients\force2\26fwq132.raw	1	2	53.3315	1.105e-003
56	c:\transients\force2\26fwq134.raw	1	1	99.9974	1.353e-003
57	c:\transients\force2\26fwq150.raw	1	2	50.4876	8.770e-003
58	c:\transients\force2\26fwq152.raw	1	1	86.3929	1.093e-002
59	c:\transients\force2\26fwq154.raw	1	1	89.9631	4.444e-003
60	c:\transients\force2\26fwq160.raw	1	2	73.4509	3.157e-003
61	c:\transients\force2\26fwq182.raw	1	2	61.1689	3.130e-003
62	c:\transients\force2\26fwq184.raw	1	1	97.7964	5.624e-002
63	c:\transients\force2\26fwq193.raw	1	1	97.3412	1.354e-002
64	c:\transients\force2\26fwq202.raw	1	1	58.5839	2.394e-003
65	c:\transients\force2\26fwq243.raw	1	1	98.3050	3.593e-002
66	c:\transients\force2\26fwq253.raw	1	1	50.9541	7.507e-004
67	c:\transients\force2\26fwq255.raw	1	1	99.6149	2.858e-002
68	c:\transients\force2\26fwq262.raw	1	1	99.9519	1.053e-002
69	c:\transients\force2\26fwq271.raw	1	1	99.5118	7.062e-003
70	c:\transients\force2\26fwq273.raw	1	2	76.0431	9.171e-003
71	c:\transients\force2\26fwq284.raw	1	1	93.4600	3.420e-002
72	c:\transients\force2\26fwq290.raw	1	2	70.2496	1.713e-003
73	c:\transients\force2\26fwq324.raw	1	1	99.6085	4.851e-004
74	c:\transients\force2\26fwq330.raw	1	2	94.7630	2.333e-002
75	c:\transients\force2\26fwq042.raw	1	2	79.8125	4.246e-003
76	c:\transients\force2\26fwq071.raw	1	1	98.5288	2.964e-002
77	c:\transients\force2\26fwq121.raw	1	1	91.2788	4.248e-002
78	c:\transients\force2\26fwq144.raw	1	1	52.8526	5.735e-003
79	c:\transients\force2\26fwq173.raw	1	1	53.7142	4.939e-003
80	c:\transients\force2\26fwq235.raw	1	1	72.6063	4.816e-003
81	c:\transients\force2\26fwq265.raw	1	1	98.2088	2.621e-003
82	c:\transients\force2\26fwq322.raw	1	2	97.5596	2.669e-002
83	c:\transients\force3\27fwi155.raw	2	2	53.6708	1.198e-002
84	c:\transients\force3\27fwi190.raw	2	2	92.9105	1.709e-003
85	c:\transients\force3\27fwi191.raw	2	2	99.7668	5.174e-002
86	c:\transients\force3\27fwi194.raw	2	1	83.1471	1.387e-002
87	c:\transients\force3\27fwi212.raw	2	2	92.1551	1.062e-001
88	c:\transients\force3\27fwi214.raw	2	2	81.9448	8.768e-003
89	c:\transients\force3\27fwi242.raw	2	2	99.3860	8.213e-003
90	c:\transients\force3\27fwi250.raw	2	2	86.2770	3.751e-002
91	c:\transients\force3\27fwi262.raw	2	1	63.8375	4.699e-005
92	c:\transients\force3\27fwi264.raw	2	2	74.8707	1.264e-002
93	c:\transients\force3\27fwi274.raw	2	2	97.1926	1.383e-001
94	c:\transients\force3\27fwi281.raw	2	1	65.2878	1.466e-002
95	c:\transients\force3\27fwi291.raw	2	2	98.6932	8.359e-002
96	c:\transients\force3\27fwi311.raw	2	2	99.4059	1.860e-003
97	c:\transients\force3\27fwi313.raw	2	2	92.0100	1.450e-002
98	c:\transients\force3\27fwi315.raw	2	2	98.9055	2.322e-002
99	c:\transients\force3\27fwi330.raw	2	2	97.8981	1.691e-002
100	c:\transients\force3\27fwi332.raw	2	2	74.3965	1.371e-002
101	c:\transients\force3\27fwi333.raw	2	2	95.5811	9.097e-003
102	c:\transients\force3\27fwi335.raw	2	2	97.6963	1.605e-002
103	c:\transients\force3\27fwi353.raw	2	1	69.2258	1.015e-002
104	c:\transients\force3\27fwi355.raw	2	2	72.6336	2.037e-002

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
105	c:\transients\force3\27fwi361.raw	2	2	96.2419	1.067e-001
106	c:\transients\force3\27fwi381.raw	2	2	96.3664	1.322e-002
107	c:\transients\force3\27fwi423.raw	2	2	69.6740	6.325e-003
108	c:\transients\force3\27fwi425.raw	2	2	98.2265	5.513e-003
109	c:\transients\force3\27fwi431.raw	2	2	98.9744	6.673e-002
110	c:\transients\force3\27fwi433.raw	2	2	98.7850	1.072e-004
111	c:\transients\force3\27fwi452.raw	2	2	99.0158	1.921e-001
112	c:\transients\force3\27fwi454.raw	2	2	93.2207	5.056e-002
113	c:\transients\force3\27fwi460.raw	2	2	95.6897	2.959e-002
114	c:\transients\force3\27fwi461.raw	2	2	83.6438	6.110e-003
115	c:\transients\force3\27fwi471.raw	2	2	61.0641	5.566e-003
116	c:\transients\force3\27fwi473.raw	2	2	88.3940	4.327e-002
117	c:\transients\force3\27fwi210.raw	2	2	87.9319	1.549e-002
118	c:\transients\force3\27fwi253.raw	2	2	98.4638	8.445e-003
119	c:\transients\force3\27fwi285.raw	2	2	76.4611	1.465e-003
120	c:\transients\force3\27fwi324.raw	2	2	49.8650	9.183e-003
121	c:\transients\force3\27fwi344.raw	2	2	77.2408	1.767e-002
122	c:\transients\force3\27fwi394.raw	2	2	96.1339	8.294e-003
123	c:\transients\force3\27fwi450.raw	2	1	57.6019	2.136e-003
124	c:\transients\force3\27fwi465.raw	2	2	97.8687	2.611e-001
125	c:\transients\ken1\25fwn251.raw	3	3	100.000	5.460e-003
126	c:\transients\ken1\25fwn252.raw	3	3	99.9499	5.725e-012
127	c:\transients\ken1\25fwn254.raw	3	3	100.000	2.097e-006
128	c:\transients\ken1\25fwn255.raw	3	3	100.000	1.965e-006
129	c:\transients\ken1\25fwn264.raw	3	3	100.000	5.544e-006
130	c:\transients\ken1\25fwn273.raw	3	5	100.000	1.094e-009
131	c:\transients\ken1\25fwn280.raw	3	3	100.000	2.525e-002
132	c:\transients\ken1\25fwn283.raw	3	3	100.000	3.984e-003
133	c:\transients\ken1\25fwn315.raw	3	3	100.000	4.710e-005
134	c:\transients\ken1\25fwn332.raw	3	3	100.000	1.391e-003
135	c:\transients\ken1\25fwn341.raw	3	3	100.000	5.694e-004
136	c:\transients\ken1\25fwn371.raw	3	3	100.000	3.846e-008
137	c:\transients\ken1\25fwn380.raw	3	3	79.7066	8.111e-010
138	c:\transients\ken1\25fwn382.raw	3	3	100.000	3.919e-004
139	c:\transients\ken1\25fwn384.raw	3	3	100.000	7.752e-004
140	c:\transients\ken1\25fwn392.raw	3	3	100.000	1.760e-006
141	c:\transients\ken1\25fwn401.raw	3	3	100.000	9.167e-005
142	c:\transients\ken1\25fwn402.raw	3	3	100.000	6.976e-004
143	c:\transients\ken1\25fwn404.raw	3	3	100.000	2.486e-003
144	c:\transients\ken1\25fwn405.raw	3	3	99.9976	5.982e-007
145	c:\transients\ken1\25fwn414.raw	3	3	100.000	4.982e-003
146	c:\transients\ken1\25fwn415.raw	3	3	100.000	9.130e-005
147	c:\transients\ken1\25fwn421.raw	3	3	100.000	9.011e-003
148	c:\transients\ken1\25fwn423.raw	3	3	100.000	4.875e-005
149	c:\transients\ken1\25fwn432.raw	3	3	100.000	2.463e-003
150	c:\transients\ken1\25fwn433.raw	3	3	100.000	1.245e-004
151	c:\transients\ken1\25fwn435.raw	3	3	100.000	1.778e-004
152	c:\transients\ken1\25fwn441.raw	3	3	100.000	6.768e-004
153	c:\transients\ken1\25fwn450.raw	3	3	100.000	1.171e-004
154	c:\transients\ken1\25fwn452.raw	3	3	100.000	2.942e-005
155	c:\transients\ken1\25fwn453.raw	3	3	99.9997	6.506e-009
156	c:\transients\ken1\25fwn455.raw	3	3	100.000	8.671e-009
157	c:\transients\ken1\25fwn263.raw	3	3	100.000	1.541e-005
158	c:\transients\ken1\25fwn312.raw	3	5	99.9880	4.317e-007

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
159	c:\transients\ken1\25fwn375.raw	3	3	100.000	9.029e-008
160	c:\transients\ken1\25fwn395.raw	3	3	100.000	7.526e-003
161	c:\transients\ken1\25fwn413.raw	3	3	100.000	6.215e-003
162	c:\transients\ken1\25fwn430.raw	3	3	100.000	3.675e-006
163	c:\transients\ken1\25fwn445.raw	3	3	100.000	1.006e-001
164	c:\transients\ken1\25fwn462.raw	3	4	100.000	2.602e-006
165	c:\transients\ken2\25fwn523.raw	4	4	100.000	2.463e-002
166	c:\transients\ken2\25fwn524.raw	4	4	100.000	4.895e-002
167	c:\transients\ken2\25fwn525.raw	4	4	100.000	1.394e-002
168	c:\transients\ken2\25fwn531.raw	4	4	100.000	1.483e-004
169	c:\transients\ken2\25fwn540.raw	4	4	100.000	5.418e-002
170	c:\transients\ken2\25fwn542.raw	4	4	100.000	5.169e-002
171	c:\transients\ken2\25fwn543.raw	4	4	100.000	3.687e-002
172	c:\transients\ken2\25fwn545.raw	4	4	100.000	6.206e-017
173	c:\transients\ken2\25fwn560.raw	4	4	100.000	5.525e-002
174	c:\transients\ken2\25fwn562.raw	4	4	100.000	5.236e-002
175	c:\transients\ken2\25fwn563.raw	4	4	100.000	7.670e-008
176	c:\transients\ken2\25fwn573.raw	4	4	100.000	1.079e-001
177	c:\transients\ken2\25fwn583.raw	4	4	100.000	7.877e-003
178	c:\transients\ken2\25fwn584.raw	4	4	100.000	3.559e-002
179	c:\transients\ken2\25fwn593.raw	4	4	100.000	4.205e-002
180	c:\transients\ken2\25fwn594.raw	4	4	100.000	5.196e-002
181	c:\transients\ken2\25fwo003.raw	4	4	100.000	5.416e-002
182	c:\transients\ken2\25fwo005.raw	4	4	100.000	1.106e-002
183	c:\transients\ken2\25fwo010.raw	4	4	100.000	1.185e-006
184	c:\transients\ken2\25fwo015.raw	4	4	100.000	1.658e-005
185	c:\transients\ken2\25fwo024.raw	4	4	100.000	9.834e-002
186	c:\transients\ken2\25fwo030.raw	4	4	100.000	5.731e-002
187	c:\transients\ken2\25fwo031.raw	4	4	100.000	9.186e-004
188	c:\transients\ken2\25fwo033.raw	4	4	100.000	6.200e-002
189	c:\transients\ken2\25fwo044.raw	4	4	100.000	1.326e-002
190	c:\transients\ken2\25fwo045.raw	4	4	100.000	4.468e-002
191	c:\transients\ken2\25fwo054.raw	4	4	100.000	5.220e-002
192	c:\transients\ken2\25fwo062.raw	4	4	100.000	1.290e-003
193	c:\transients\ken2\25fwo074.raw	4	4	100.000	2.843e-002
194	c:\transients\ken2\25fwo075.raw	4	4	100.000	3.551e-016
195	c:\transients\ken2\25fwo080.raw	4	4	100.000	4.502e-003
196	c:\transients\ken2\25fwo082.raw	4	4	100.000	2.857e-002
197	c:\transients\ken2\25fwo090.raw	4	4	100.000	1.026e-001
198	c:\transients\ken2\25fwn534.raw	4	4	100.000	4.142e-019
199	c:\transients\ken2\25fwn552.raw	4	4	100.000	3.236e-002
200	c:\transients\ken2\25fwn580.raw	4	4	100.000	1.554e-002
201	c:\transients\ken2\25fwo002.raw	4	4	100.000	2.244e-002
202	c:\transients\ken2\25fwo022.raw	4	4	100.000	1.011e-002
203	c:\transients\ken2\25fwo042.raw	4	4	100.000	1.474e-003
204	c:\transients\ken2\25fwo065.raw	4	4	100.000	8.829e-003
205	c:\transients\ken2\25fwo085.raw	4	4	100.000	1.625e-002
206	c:\transients\ken3\25fwo165.raw	5	5	100.000	1.123e-004
207	c:\transients\ken3\25fwo170.raw	5	5	100.000	8.658e-005
208	c:\transients\ken3\25fwo180.raw	5	5	100.000	4.872e-004
209	c:\transients\ken3\25fwo182.raw	5	5	100.000	5.720e-004
210	c:\transients\ken3\25fwo191.raw	5	5	100.000	3.445e-004
211	c:\transients\ken3\25fwo192.raw	5	5	100.000	9.027e-004
212	c:\transients\ken3\25fwo194.raw	5	5	100.000	1.591e-010

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
213	c:\transients\ken3\25fwo195.raw	5	5	100.000	1.643e-015
214	c:\transients\ken3\25fwo212.raw	5	5	100.000	9.374e-007
215	c:\transients\ken3\25fwo213.raw	5	5	100.000	2.781e-003
216	c:\transients\ken3\25fwo215.raw	5	5	100.000	8.837e-006
217	c:\transients\ken3\25fwo220.raw	5	5	100.000	1.021e-004
218	c:\transients\ken3\25fwo225.raw	5	5	100.000	3.144e-004
219	c:\transients\ken3\25fwo231.raw	5	5	100.000	3.835e-011
220	c:\transients\ken3\25fwo232.raw	5	5	100.000	2.300e-010
221	c:\transients\ken3\25fwo234.raw	5	5	100.000	1.578e-004
222	c:\transients\ken3\25fwo251.raw	5	3	100.000	8.656e-010
223	c:\transients\ken3\25fwo253.raw	5	5	100.000	7.052e-005
224	c:\transients\ken3\25fwo260.raw	5	5	100.000	1.869e-014
225	c:\transients\ken3\25fwo265.raw	5	5	100.000	3.032e-002
226	c:\transients\ken3\25fwo283.raw	5	5	100.000	2.610e-004
227	c:\transients\ken3\25fwo301.raw	5	5	100.000	2.047e-003
228	c:\transients\ken3\25fwo303.raw	5	5	100.000	3.956e-003
229	c:\transients\ken3\25fwo305.raw	5	5	100.000	3.454e-006
230	c:\transients\ken3\25fwo315.raw	5	5	100.000	7.647e-004
231	c:\transients\ken3\25fwo321.raw	5	5	100.000	4.513e-006
232	c:\transients\ken3\25fwo325.raw	5	5	100.000	9.925e-003
233	c:\transients\ken3\25fwo332.raw	5	5	100.000	1.781e-002
234	c:\transients\ken3\25fwo341.raw	5	5	100.000	1.282e-003
235	c:\transients\ken3\25fwo342.raw	5	5	100.000	2.557e-003
236	c:\transients\ken3\25fwo344.raw	5	5	100.000	3.657e-006
237	c:\transients\ken3\25fwo352.raw	5	5	100.000	1.751e-003
238	c:\transients\ken3\25fwp065.raw	5	5	100.000	5.003e-006
239	c:\transients\ken3\25fwp075.raw	5	5	100.000	2.319e-006
240	c:\transients\ken3\25fwo190.raw	5	3	100.000	8.911e-017
241	c:\transients\ken3\25fwo204.raw	5	5	100.000	2.627e-006
242	c:\transients\ken3\25fwo223.raw	5	5	100.000	2.677e-002
243	c:\transients\ken3\25fwo241.raw	5	5	100.000	2.053e-004
244	c:\transients\ken3\25fwo282.raw	5	5	100.000	1.234e-003
245	c:\transients\ken3\25fwo314.raw	5	5	100.000	1.643e-003
246	c:\transients\ken3\25fwo335.raw	5	5	100.000	3.846e-009
247	c:\transients\ken3\25fwp063.raw	5	5	99.9999	2.750e-011
248	c:\transients\Yaesu1\25fwp323.raw	6	6	99.9995	8.397e-004
249	c:\transients\Yaesu1\25fwp325.raw	6	6	100.000	4.611e-007
250	c:\transients\Yaesu1\25fwp331.raw	6	6	100.000	5.822e-004
251	c:\transients\Yaesu1\26fwb212.raw	6	6	100.000	6.095e-002
252	c:\transients\Yaesu1\26fwb235.raw	6	6	100.000	2.815e-003
253	c:\transients\Yaesu1\26fwb240.raw	6	6	99.9999	6.437e-010
254	c:\transients\Yaesu1\26fwb242.raw	6	6	100.000	1.570e-002
255	c:\transients\Yaesu1\26fwb251.raw	6	7	87.7067	2.373e-015
256	c:\transients\Yaesu1\26fwb272.raw	6	6	100.000	1.549e-002
257	c:\transients\Yaesu1\26fwb274.raw	6	6	100.000	2.245e-004
258	c:\transients\Yaesu1\26fwb280.raw	6	6	100.000	1.361e-003
259	c:\transients\Yaesu1\26fwb281.raw	6	6	100.000	3.453e-003
260	c:\transients\Yaesu1\26fwb312.raw	6	6	100.000	1.647e-004
261	c:\transients\Yaesu1\26fwb314.raw	6	6	100.000	2.779e-003
262	c:\transients\Yaesu1\26fwb332.raw	6	6	100.000	1.856e-003
263	c:\transients\Yaesu1\26fwb334.raw	6	6	100.000	1.067e-004
264	c:\transients\Yaesu1\26fwb343.raw	6	6	100.000	1.640e-004
265	c:\transients\Yaesu1\26fwb345.raw	6	6	100.000	1.121e-003
266	c:\transients\Yaesu1\26fwb355.raw	6	6	100.000	1.472e-004

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
267	c:\transients\Yaesu1\26fwk361.raw	6	6	100.000	1.734e-004
268	c:\transients\Yaesu1\26fwk370.raw	6	6	100.000	1.926e-003
269	c:\transients\Yaesu1\26fwk372.raw	6	6	100.000	6.797e-003
270	c:\transients\Yaesu1\26fwk435.raw	6	6	100.000	1.843e-006
271	c:\transients\Yaesu1\26fwk450.raw	6	6	100.000	2.480e-008
272	c:\transients\Yaesu1\26fwk455.raw	6	6	100.000	7.291e-005
273	c:\transients\Yaesu1\26fwk483.raw	6	6	100.000	8.990e-004
274	c:\transients\Yaesu1\26fwk502.raw	6	6	100.000	5.113e-003
275	c:\transients\Yaesu1\26fwk505.raw	6	6	100.000	2.704e-004
276	c:\transients\Yaesu1\26fwk524.raw	6	6	100.000	2.773e-004
277	c:\transients\Yaesu1\26fwk532.raw	6	6	100.000	5.899e-005
278	c:\transients\Yaesu1\26fwk541.raw	6	6	100.000	2.852e-004
279	c:\transients\Yaesu1\26fwk544.raw	6	6	100.000	3.999e-003
280	c:\transients\Yaesu1\26fwk564.raw	6	6	100.000	4.680e-004
281	c:\transients\Yaesu1\26fwk570.raw	6	6	100.000	7.218e-005
282	c:\transients\Yaesu1\26fwk571.raw	6	6	100.000	3.993e-002
283	c:\transients\Yaesu1\26fwk573.raw	6	6	100.000	2.228e-004
284	c:\transients\Yaesu1\26fwk575.raw	6	6	100.000	2.042e-004
285	c:\transients\Yaesu1\26fwk585.raw	6	6	100.000	7.498e-004
286	c:\transients\Yaesu1\26fwl002.raw	6	6	100.000	3.892e-003
287	c:\transients\Yaesu1\26fwl003.raw	6	6	100.000	3.176e-004
288	c:\transients\Yaesu1\26fwk233.raw	6	6	100.000	3.645e-005
289	c:\transients\Yaesu1\26fwk264.raw	6	6	100.000	1.995e-003
290	c:\transients\Yaesu1\26fwk285.raw	6	6	100.000	2.386e-005
291	c:\transients\Yaesu1\26fwk341.raw	6	6	100.000	3.214e-004
292	c:\transients\Yaesu1\26fwk365.raw	6	6	100.000	4.082e-005
293	c:\transients\Yaesu1\26fwk453.raw	6	6	100.000	2.992e-004
294	c:\transients\Yaesu1\26fwk522.raw	6	6	100.000	6.251e-004
295	c:\transients\Yaesu1\26fwk551.raw	6	6	100.000	8.045e-004
296	c:\transients\Yaesu2\26fwl125.raw	7	7	100.000	3.665e-008
297	c:\transients\Yaesu2\26fwl132.raw	7	7	100.000	5.922e-005
298	c:\transients\Yaesu2\26fwl133.raw	7	7	100.000	4.710e-003
299	c:\transients\Yaesu2\26fwl140.raw	7	7	100.000	6.252e-014
300	c:\transients\Yaesu2\26fwl173.raw	7	7	99.9998	1.555e-007
301	c:\transients\Yaesu2\26fwl180.raw	7	7	100.000	4.691e-010
302	c:\transients\Yaesu2\26fwl181.raw	7	7	99.9180	3.642e-005
303	c:\transients\Yaesu2\26fwl190.raw	7	7	99.9994	4.992e-008
304	c:\transients\Yaesu2\26fwl200.raw	7	7	99.9996	3.441e-003
305	c:\transients\Yaesu2\26fwl202.raw	7	7	99.9487	1.025e-005
306	c:\transients\Yaesu2\26fwl204.raw	7	7	99.9994	1.699e-005
307	c:\transients\Yaesu2\26fwl212.raw	7	7	100.000	1.427e-004
308	c:\transients\Yaesu2\26fwl224.raw	7	7	100.000	2.004e-005
309	c:\transients\Yaesu2\26fwl225.raw	7	7	99.9838	6.051e-015
310	c:\transients\Yaesu2\26fwl231.raw	7	7	100.000	1.250e-002
311	c:\transients\Yaesu2\26fwl233.raw	7	7	93.1440	9.664e-010
312	c:\transients\Yaesu2\26fwl243.raw	7	7	100.000	7.943e-004
313	c:\transients\Yaesu2\26fwl245.raw	7	7	100.000	9.344e-004
314	c:\transients\Yaesu2\26fwl251.raw	7	7	100.000	3.280e-005
315	c:\transients\Yaesu2\26fwl254.raw	7	7	99.9950	3.691e-005
316	c:\transients\Yaesu2\26fwl264.raw	7	7	100.000	2.334e-006
317	c:\transients\Yaesu2\26fwl270.raw	7	7	100.000	2.267e-005
318	c:\transients\Yaesu2\26fwl272.raw	7	7	100.000	1.319e-003
319	c:\transients\Yaesu2\26fwl274.raw	7	7	100.000	6.216e-005
320	c:\transients\Yaesu2\26fwl283.raw	7	7	100.000	1.477e-005

Appendix E: Experimental Classification Results

<u>Transient #</u>	<u>FileName and Path</u>	<u>Class</u>	<u>Predicted</u>	<u>Conf(%)</u>	<u>Activation</u>
321	c:\transients\Yaesu2\26fw1285.raw	7	7	100.000	3.616e-003
322	c:\transients\Yaesu2\26fw1293.raw	7	7	100.000	8.032e-003
323	c:\transients\Yaesu2\26fw1300.raw	7	7	100.000	2.014e-007
324	c:\transients\Yaesu2\26fw1312.raw	7	7	100.000	6.208e-006
325	c:\transients\Yaesu2\26fw1314.raw	7	7	100.000	1.170e-002
326	c:\transients\Yaesu2\26fw1320.raw	7	7	99.9978	2.223e-006
327	c:\transients\Yaesu2\26fw1322.raw	7	7	100.000	1.611e-006
328	c:\transients\Yaesu2\26fw1153.raw	7	7	100.000	1.143e-003
329	c:\transients\Yaesu2\26fw1194.raw	7	7	100.000	2.253e-003
330	c:\transients\Yaesu2\26fw1220.raw	7	7	100.000	2.229e-004
331	c:\transients\Yaesu2\26fw1241.raw	7	7	100.000	1.334e-004
332	c:\transients\Yaesu2\26fw1262.raw	7	7	100.000	3.268e-006
333	c:\transients\Yaesu2\26fw1281.raw	7	7	100.000	2.751e-008
334	c:\transients\Yaesu2\26fw1310.raw	7	7	99.6388	5.069e-007
335	c:\transients\Yaesu2\26fw1332.raw	7	7	100.000	1.181e-003

Number of Correct Classifications: 311/335.

VALIDATION SET #3 WITH MULTIMODAL SEGMENTATION

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
1	d:\Tran\force1\26fwo325.raw	0	0	99.8926	5.463e-003
2	d:\Tran\force1\26fwo341.raw	0	0	99.9984	1.043e-004
3	d:\Tran\force1\26fwo362.raw	0	0	99.9979	2.104e-002
4	d:\Tran\force1\26fwo380.raw	0	0	99.9989	1.378e-002
5	d:\Tran\force1\26fwo390.raw	0	0	100.000	2.124e-003
6	d:\Tran\force1\26fwo395.raw	0	0	100.000	5.131e-010
7	d:\Tran\force1\26fwo413.raw	0	0	100.000	9.578e-004
8	d:\Tran\force1\26fwo440.raw	0	0	99.9922	1.209e-001
9	d:\Tran\force1\26fwo453.raw	0	0	99.9286	3.794e-003
10	d:\Tran\force1\26fwo461.raw	0	0	99.9226	2.248e-002
11	d:\Tran\force1\26fwo465.raw	0	0	100.000	7.170e-004
12	d:\Tran\force1\26fwo481.raw	0	0	99.9812	4.327e-003
13	d:\Tran\force1\26fwo492.raw	0	0	99.9978	6.492e-004
14	d:\Tran\force1\26fwo500.raw	0	0	99.9998	1.808e-007
15	d:\Tran\force1\26fwo504.raw	0	0	99.6512	4.591e-003
16	d:\Tran\force1\26fwo515.raw	0	6	99.9876	4.538e-018
17	d:\Tran\force1\26fwo524.raw	0	0	99.9921	1.161e-001
18	d:\Tran\force1\26fwo532.raw	0	0	99.9132	9.382e-003
19	d:\Tran\force1\26fwo545.raw	0	0	99.9924	3.996e-013
20	d:\Tran\force1\26fwo554.raw	0	0	99.9952	1.166e-001
21	d:\Tran\force2\26fwp571.raw	1	1	98.6079	6.603e-004
22	d:\Tran\force2\26fwp592.raw	1	1	99.9953	1.866e-003
23	d:\Tran\force2\26fwq042.raw	1	1	99.5591	1.811e-003
24	d:\Tran\force2\26fwq061.raw	1	1	99.9973	1.735e-003
25	d:\Tran\force2\26fwq071.raw	1	1	99.9944	4.544e-003
26	d:\Tran\force2\26fwq084.raw	1	1	99.9971	8.951e-003
27	d:\Tran\force2\26fwq094.raw	1	1	93.5082	6.595e-003
28	d:\Tran\force2\26fwq125.raw	1	1	98.6123	2.460e-003
29	d:\Tran\force2\26fwq134.raw	1	1	99.9221	5.584e-003
30	d:\Tran\force2\26fwq144.raw	1	1	99.9958	1.787e-002
31	d:\Tran\force2\26fwq152.raw	1	1	99.9693	2.620e-002
32	d:\Tran\force2\26fwq160.raw	1	1	71.6363	2.144e-004
33	d:\Tran\force2\26fwq184.raw	1	1	99.4733	1.559e-002
34	d:\Tran\force2\26fwq202.raw	1	1	99.9227	6.945e-003
35	d:\Tran\force2\26fwq235.raw	1	1	99.9116	1.418e-003
36	d:\Tran\force2\26fwq253.raw	1	1	98.6045	6.049e-004
37	d:\Tran\force2\26fwq262.raw	1	1	99.4930	2.828e-003
38	d:\Tran\force2\26fwq265.raw	1	1	98.1628	9.004e-004
39	d:\Tran\force2\26fwq265.raw	1	1	98.1628	9.004e-004
40	d:\Tran\force2\26fwq273.raw	1	1	99.0124	2.360e-003
41	d:\Tran\force2\26fwq290.raw	1	1	99.9512	6.777e-003
42	d:\Tran\force2\26fwq330.raw	1	2	97.8754	1.450e-003
43	d:\Tran\force3\27fwi153.raw	2	2	99.9535	2.249e-002
44	d:\Tran\force3\27fwi190.raw	2	2	99.9989	6.353e-004
45	d:\Tran\force3\27fwi194.raw	2	2	99.5988	9.376e-003
46	d:\Tran\force3\27fwi210.raw	2	2	99.9578	4.982e-003
47	d:\Tran\force3\27fwi214.raw	2	2	80.2974	3.152e-004
48	d:\Tran\force3\27fwi250.raw	2	0	66.9519	3.435e-006
49	d:\Tran\force3\27fwi253.raw	2	2	99.9972	2.663e-002
50	d:\Tran\force3\27fwi264.raw	2	2	99.9998	1.882e-004

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
51	d:\Tran\force3\27fwi311.raw	2	2	99.9990	1.737e-004
52	d:\Tran\force3\27fwi315.raw	2	2	99.9717	2.796e-003
53	d:\Tran\force3\27fwi324.raw	2	2	98.0694	8.816e-003
54	d:\Tran\force3\27fwi332.raw	2	2	99.3552	6.830e-003
55	d:\Tran\force3\27fwi344.raw	2	2	97.6002	8.355e-004
56	d:\Tran\force3\27fwi355.raw	2	2	70.6164	3.199e-004
57	d:\Tran\force3\27fwi381.raw	2	2	99.9826	2.472e-002
58	d:\Tran\force3\27fwi394.raw	2	2	99.9475	1.142e-004
59	d:\Tran\force3\27fwi425.raw	2	2	99.9891	1.230e-002
60	d:\Tran\force3\27fwi433.raw	2	2	99.9995	1.506e-004
61	d:\Tran\force3\27fwi454.raw	2	2	99.9879	1.616e-002
62	d:\Tran\force3\27fwi461.raw	2	2	99.9972	8.852e-003
63	d:\Tran\force3\27fwi465.raw	2	2	99.9950	6.967e-002
64	d:\Tran\force3\27fwi473.raw	2	2	99.9850	5.993e-003
65	d:\Tran\ken1\25fwn245.raw	3	5	94.4883	1.353e-011
66	d:\Tran\ken1\25fwn255.raw	3	3	100.000	1.323e-011
67	d:\Tran\ken1\25fwn263.raw	3	3	100.000	2.738e-008
68	d:\Tran\ken1\25fwn273.raw	3	5	100.000	1.073e-019
69	d:\Tran\ken1\25fwn332.raw	3	3	100.000	2.294e-004
70	d:\Tran\ken1\25fwn371.raw	3	3	100.000	3.917e-007
71	d:\Tran\ken1\25fwn375.raw	3	3	100.000	1.048e-006
72	d:\Tran\ken1\25fwn382.raw	3	3	100.000	3.343e-005
73	d:\Tran\ken1\25fwn392.raw	3	3	100.000	4.877e-007
74	d:\Tran\ken1\25fwn395.raw	3	3	100.000	1.811e-003
75	d:\Tran\ken1\25fwn405.raw	3	3	91.5468	5.758e-010
76	d:\Tran\ken1\25fwn413.raw	3	3	100.000	1.716e-004
77	d:\Tran\ken1\25fwn415.raw	3	3	100.000	1.208e-003
78	d:\Tran\ken1\25fwn423.raw	3	3	100.000	3.420e-004
79	d:\Tran\ken1\25fwn430.raw	3	3	100.000	6.638e-005
80	d:\Tran\ken1\25fwn433.raw	3	3	100.000	4.882e-017
81	d:\Tran\ken1\25fwn445.raw	3	3	100.000	4.985e-004
82	d:\Tran\ken1\25fwn452.raw	3	3	100.000	5.336e-004
83	d:\Tran\ken1\25fwn455.raw	3	5	100.000	6.166e-010
84	d:\Tran\ken1\25fwn462.raw	3	4	99.9996	2.025e-010
85	d:\Tran\ken2\25fwn524.raw	4	4	100.000	1.620e-002
86	d:\Tran\ken2\25fwn531.raw	4	4	100.000	2.917e-005
87	d:\Tran\ken2\25fwn534.raw	4	4	100.000	4.342e-023
88	d:\Tran\ken2\25fwn545.raw	4	4	100.000	2.095e-028
89	d:\Tran\ken2\25fwn552.raw	4	4	100.000	1.066e-002
90	d:\Tran\ken2\25fwn562.raw	4	4	100.000	5.459e-003
91	d:\Tran\ken2\25fwn580.raw	4	4	100.000	2.039e-003
92	d:\Tran\ken2\25fwn584.raw	4	4	100.000	1.105e-002
93	d:\Tran\ken2\25fwn594.raw	4	4	100.000	1.768e-002
94	d:\Tran\ken2\25fwo002.raw	4	4	100.000	1.592e-002
95	d:\Tran\ken2\25fwo005.raw	4	4	100.000	1.453e-003
96	d:\Tran\ken2\25fwo015.raw	4	4	100.000	5.525e-004
97	d:\Tran\ken2\25fwo022.raw	4	4	100.000	1.739e-004
98	d:\Tran\ken2\25fwo030.raw	4	4	100.000	1.034e-002
99	d:\Tran\ken2\25fwo033.raw	4	4	100.000	1.890e-002
100	d:\Tran\ken2\25fwo045.raw	4	4	100.000	1.549e-002
101	d:\Tran\ken2\25fwo062.raw	4	4	100.000	7.223e-003
102	d:\Tran\ken2\25fwo065.raw	4	4	100.000	5.648e-004
103	d:\Tran\ken2\25fwo075.raw	4	4	100.000	4.986e-017
104	d:\Tran\ken2\25fwo082.raw	4	4	100.000	7.541e-003

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
105	d:\Tran\ken2\25fwo085.raw	4	4	100.000	3.215e-004
106	d:\Tran\ken3\25fwo160.raw	5	5	100.000	6.026e-015
107	d:\Tran\ken3\25fwo182.raw	5	5	100.000	6.257e-003
108	d:\Tran\ken3\25fwo190.raw	5	5	100.000	2.828e-016
109	d:\Tran\ken3\25fwo192.raw	5	5	100.000	1.226e-008
110	d:\Tran\ken3\25fwo195.raw	5	5	100.000	2.008e-003
111	d:\Tran\ken3\25fwo213.raw	5	5	100.000	8.576e-009
112	d:\Tran\ken3\25fwo220.raw	5	5	100.000	4.598e-009
113	d:\Tran\ken3\25fwo223.raw	5	5	100.000	1.904e-007
114	d:\Tran\ken3\25fwo231.raw	5	5	100.000	1.405e-015
115	d:\Tran\ken3\25fwo241.raw	5	5	100.000	1.519e-019
116	d:\Tran\ken3\25fwo253.raw	5	5	100.000	2.868e-004
117	d:\Tran\ken3\25fwo265.raw	5	5	100.000	2.371e-005
118	d:\Tran\ken3\25fwo282.raw	5	5	100.000	4.801e-013
119	d:\Tran\ken3\25fwo301.raw	5	5	100.000	3.555e-006
120	d:\Tran\ken3\25fwo305.raw	5	5	100.000	5.441e-007
121	d:\Tran\ken3\25fwo314.raw	5	5	100.000	1.308e-006
122	d:\Tran\ken3\25fwo321.raw	5	5	100.000	8.118e-004
123	d:\Tran\ken3\25fwo332.raw	5	5	100.000	6.424e-005
124	d:\Tran\ken3\25fwo342.raw	5	5	100.000	1.182e-004
125	d:\Tran\ken3\25fwo352.raw	5	5	100.000	2.553e-005
126	d:\Tran\ken3\25fwp063.raw	5	5	100.000	2.134e-018
127	d:\Tran\ken3\25fwp075.raw	5	5	100.000	1.269e-013
128	d:\Tran\Yaesu1\25fwp321.raw	6	6	100.000	4.638e-008
129	d:\Tran\Yaesu1\25fwp325.raw	6	6	100.000	3.662e-011
130	d:\Tran\Yaesu1\26fwk212.raw	6	6	100.000	1.089e-002
131	d:\Tran\Yaesu1\26fwk233.raw	6	6	100.000	6.059e-010
132	d:\Tran\Yaesu1\26fwk240.raw	6	6	100.000	3.081e-003
133	d:\Tran\Yaesu1\26fwk264.raw	6	6	100.000	1.404e-004
134	d:\Tran\Yaesu1\26fwk274.raw	6	6	100.000	1.420e-004
135	d:\Tran\Yaesu1\26fwk281.raw	6	6	100.000	9.814e-008
136	d:\Tran\Yaesu1\26fwk285.raw	6	6	100.000	3.375e-008
137	d:\Tran\Yaesu1\26fwk314.raw	6	6	100.000	1.482e-006
138	d:\Tran\Yaesu1\26fwk334.raw	6	6	100.000	1.449e-004
139	d:\Tran\Yaesu1\26fwk341.raw	6	6	100.000	1.542e-003
140	d:\Tran\Yaesu1\26fwk345.raw	6	6	100.000	2.657e-003
141	d:\Tran\Yaesu1\26fwk361.raw	6	6	100.000	1.060e-003
142	d:\Tran\Yaesu1\26fwk365.raw	6	6	100.000	9.570e-009
143	d:\Tran\Yaesu1\26fwk372.raw	6	6	100.000	2.675e-005
144	d:\Tran\Yaesu1\26fwk450.raw	6	7	100.000	1.230e-019
145	d:\Tran\Yaesu1\26fwk453.raw	6	6	100.000	5.858e-004
146	d:\Tran\Yaesu1\26fwk483.raw	6	6	100.000	1.900e-002
147	d:\Tran\Yaesu1\26fwk505.raw	6	6	100.000	5.597e-005
148	d:\Tran\Yaesu1\26fwk522.raw	6	6	100.000	4.260e-005
149	d:\Tran\Yaesu1\26fwk532.raw	6	6	100.000	2.532e-006
150	d:\Tran\Yaesu1\26fwk544.raw	6	6	100.000	2.774e-005
151	d:\Tran\Yaesu1\26fwk551.raw	6	6	100.000	6.505e-004
152	d:\Tran\Yaesu1\26fwk570.raw	6	6	100.000	2.163e-007
153	d:\Tran\Yaesu1\26fwk573.raw	6	6	100.000	1.593e-004
154	d:\Tran\Yaesu1\26fwk585.raw	6	6	100.000	7.506e-005
155	d:\Tran\Yaesu1\26fwl003.raw	6	6	100.000	8.506e-010
156	d:\Tran\Yaesu2\26fwl132.raw	7	7	100.000	2.059e-008
157	d:\Tran\Yaesu2\26fwl140.raw	7	7	100.000	6.045e-018
158	d:\Tran\Yaesu2\26fwl153.raw	7	7	100.000	4.353e-005

Appendix E: Experimental Classification Results

Transient #	FileName and Path	Class	Predicted	Conf(%)	Activation
159	d:\Tran\Yaesu2\26fw1190.raw	7	7	100.000	3.778e-013
160	d:\Tran\Yaesu2\26fw1194.raw	7	7	100.000	7.177e-008
161	d:\Tran\Yaesu2\26fw1202.raw	7	7	100.000	1.009e-003
162	d:\Tran\Yaesu2\26fw1220.raw	7	7	100.000	6.136e-008
163	d:\Tran\Yaesu2\26fw1233.raw	7	7	100.000	7.740e-017
164	d:\Tran\Yaesu2\26fw1241.raw	7	7	100.000	3.959e-007
165	d:\Tran\Yaesu2\26fw1245.raw	7	7	100.000	4.627e-006
166	d:\Tran\Yaesu2\26fw1254.raw	7	7	100.000	1.431e-005
167	d:\Tran\Yaesu2\26fw1262.raw	7	7	100.000	2.183e-007
168	d:\Tran\Yaesu2\26fw1270.raw	7	7	100.000	1.812e-008
169	d:\Tran\Yaesu2\26fw1274.raw	7	7	100.000	7.197e-008
170	d:\Tran\Yaesu2\26fw1281.raw	7	7	100.000	9.704e-012
171	d:\Tran\Yaesu2\26fw1285.raw	7	7	100.000	2.628e-005
172	d:\Tran\Yaesu2\26fw1300.raw	7	7	100.000	5.300e-015
173	d:\Tran\Yaesu2\26fw1310.raw	7	7	100.000	4.765e-005
174	d:\Tran\Yaesu2\26fw1314.raw	7	7	100.000	1.974e-004
175	d:\Tran\Yaesu2\26fw1332.raw	7	7	100.000	1.289e-006

Number of Correct Classifications: 167/175.