2004

Self Designing Pattern Recognition System Employing Multistage Classification

Manal M. Abdelwahab
University of Central Florida

SELF DESIGNING PATTERN RECOGNITION SYSTEM EMPLOYING MULTISTAGE CLASSIFICATION

by

MANAL M. ABDELWAHAB
B. S. Alexandria University, 1996
M. S. Alexandria University, 1998

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2004

Major Advisor: Wasfy B. Mikhael

ABSTRACT

Recently, pattern recognition/classification has received a considerable attention in diverse engineering fields such as biomedical imaging, speaker identification, fingerprint recognition, etc. In most of these applications, it is desirable to maintain the classification accuracy in the presence of corrupted and/or incomplete data. The quality of a given classification technique is measured by the computational complexity, execution time of algorithms, and the number of patterns that can be classified correctly despite any distortion. Some classification techniques that are introduced in the literature are described in Chapter one.

In this dissertation, a pattern recognition approach that can be designed to have evolutionary learning by developing the features and selecting the criteria that are best suited for the recognition problem under consideration is proposed. Chapter two presents some of the features used in developing the set of criteria employed by the system to recognize different types of signals. It also presents some of the preprocessing techniques used by the system.

The system operates in two modes, namely, the learning (training) mode, and the running mode. In the learning mode, the original and preprocessed signals are projected into different transform domains. The technique automatically tests many criteria over the range of parameters for each criterion. A large number of criteria are developed from the features extracted from these domains. The optimum set of criteria, satisfying specific conditions, is selected. This set of criteria is employed by the system to recognize the original or noisy signals in the running mode.

The modes of operation and the classification structures employed by the system are described in details in Chapter three.

The proposed pattern recognition system is capable of recognizing an enormously large number of patterns by virtue of the fact that it analyzes the signal in different domains and explores the distinguishing characteristics in each of these domains. In other words, this approach uses available information and extracts more characteristics from the signals, for classification purposes, by projecting the signal in different domains. Some experimental results are given in Chapter four showing the effect of using mathematical transforms in conjunction with preprocessing techniques on the classification accuracy. A comparison between some of the classification approaches, in terms of classification rate in case of distortion, is also given.

A sample of experimental implementations is presented in chapter 5 and chapter 6 to illustrate the performance of the proposed pattern recognition system. Preliminary results given confirm the superior performance of the proposed technique relative to the single transform neural network and multi-input neural network approaches for image classification in the presence of additive noise.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Dr. Wasfy Mikhael for his continuous support and patience. His help and guidance throughout my studies at UCF are invaluable. I will always be indebted to him for he has taught me a lot. His experience was really of great benefit to me. His advice in academia and otherwise have undoubtedly left a mark on me.

I would also like to thank Dr. Berg for his valuable input and meticulous effort in reviewing much of this research. My deepest thanks also go to members of my committees: Dr. F. Moslehy, Dr. I. Batarseh, and Dr. M. Haralambous for their encouragement and patience during my studies.

Finally, I would like to thank my husband for his help and patience without which I would not have been able to pursue and accomplish this dream. The little patience of my little kids, Marwan and Ranna, is also appreciated.

Last but not least, I would like to thank my parents. They inspired me, and simply put: without them I would not be the person I am today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BCU: Binary Classification Unit

DCT: Discrete Cosine Transform

DFT: Discrete Fourier Transform

DT: Decision Tree

HMM: Hidden Markov Models

MCE: Minimum Classification Error

MCMTNN: Multi Criteria Multi Transform Neural Network

MSE: Minimum Square Error

N: number of signals

NN: Neural Network

STNN: Single Transform Neural Network

SVD: Singular Value Decomposition

VQ: Vector Quantization

WH: Walsh-Hadamard Transform

WT: Wavelet Ttransform

CHAPTER ONE: INTRODUCTION

1.1 Problem Addressed

Pattern Recognition has become a point of interest to many researches due to its applications in many fields. Their main interest is increasing the accuracy of the pattern recognition process in the presence of corrupted or incomplete data.

In designing any pattern recognition system, there are four important factors that must be taken into consideration: processing speed, recognition rate, size, and power consumption [30]. There are two ways to increase the accuracy of pattern recognition system. One is to improve the performance of a single classifier; another is to combine the results of multiple classifiers (decisions combination).

Ordinarily, a pattern recognition problem may involve a number of pattern classes with each class consisting of various features. It is difficult for a single classifier to achieve perfect solution. The multiple classifier method thereby becomes the best choice for solving pattern recognition problems. Generally speaking, multiple classifier method is superior to a single classifier method if the classifiers are selected carefully and the combination algorithm can take the advantages of each individual classifier and avoid its

weakness [49]. There are two most important issues in designing a multiple classifiers system:

1) Classifier selection.
2) Decision combination.

In this chapter, some of the pattern recognition applications are briefly described for the sake of completeness. A number of the classification techniques, presented in the literature, are given with their advantages and disadvantages. Finally, the difference between the single and multiple classifiers will be demonstrated.

1.2 Pattern recognition applications

Recently, pattern recognition has received great deal of attention in diverse engineering fields such as oil exploration, biomedical imaging, speaker identification, automated data entry, fingerprint recognition [51], evaluation of the fetal state as carried out by obstetricians [28], digital modulation type classification under various SNR and multi-path propagation channel conditions [45], etc.

Many valuable contributions have been reported [20, 65, 67] in the different fields of applications. In most of these applications, it is desirable to maintain the classification accuracy in the presence of corrupted and/or incomplete data.

1.2.1 Face recognition

Face recognition is one of the important research topics in the pattern recognition area that has been receiving the attention of many researchers due to its useful applications, such as system security, military, human-computer interface, etc [18, 60]. In order to design an artificial image classification or recognition scheme which should have robustness in classification approaching as close as possible to that of human biological recognition system, two factors must be taken into account [90]:

1) It must be able to automatically extract global properties of the images.
2) It must be able to filter out the variation such as scaling and rotation in the images.

Key issues in face recognition are how to model the "face space" which facial images occupy in the original image space and how to design the feature extraction procedure that is best suited for that design model [80]. Two main approaches for feature extraction have been presented in literature. The first approach is based on extracting the local structure of face images such as the shapes of the eyes, nose and mouth. The second one is statistical-based approaches that extract global features from the whole image [42].

Existing face detection methods include [93] template-based, neural network-based, model-based, color-based and motion-based approaches. Gabor-wavelet-labeled elastic graph matching and "eigenface" or "Fisherface" algorithms are two of the major approaches proposed in the literature for facial image processing [68]. The eigenface algorithm is based on statistical representation of face space. Any face could be reconstructed by the summation of weights for each face. Eigenface is a fast, simple and practical method. But, it does not provide invariance over changes in scale and lightning conditions [31].

1.2.2 Handwriting recognition

Handwritten numerals recognition is also one of the important pattern recognition applications. Many features and classification methods have been proposed in past several decades to recognize them [41, 96]. One approach includes two categories, statistical and syntactic methods. The first category includes techniques such as matching, moments, characteristic points and mathematical transforms, while the second one includes essential structural features such as skeletons and contours. The other approach includes the use of a single classifier and combined classifiers.

Pattichis *et al*. have used AM-FM models to describe fingerprints and they have obtained significant gains in classification performance [83]. Optical Character Recognizer (OCR) has also played a good role in handwritten recognition [96].

1.3 Classification

The strength of a given classification technique is measured by the number of different categories that can be distinguished [90]. It is also measured by the computational complexity, execution time of algorithms, and the number of patterns that can be classified correctly despite any distortion [66, 69].

Each technique has its strong and weak points which make it suitable for certain types of problems. Classifier generality refers to the ability to classify images correctly despite distortions. Distortions may include image noise or image content variations such as scale,

translation and rotation of objects, as well as plastic deformations of depicted objects as the difference between a smiling face and a frowning face.

Sandberg [89] shows that if we are given a finite number m of pairwise disjoint sets $C_1, C_2 \ldots Cm$ of signals, and we would like to synthesize a system that maps the elements of each C_j into a real number a_j so that the numbers $a_1, a_1 \ldots a_m$ are distinct, the structure in Figure 1.1 can perform this classification assuming only that the C_j are compact subsets of a real normed linear space X.

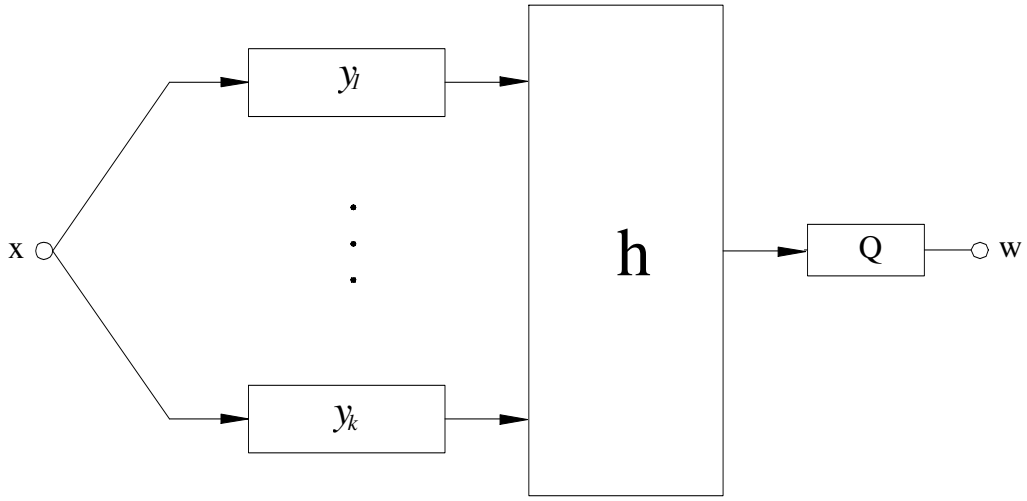


Figure 1.1: Classification network

In Figure 1.1, x is the signal to be classified, $y_1,\ldots, y_k$ are functions that can be taken to be linear, h denotes a continuos memoryless nonlinear mapping that, for example can be implemented by a neural network having one hidden nonlinear layer, the block labeled Q is a quantizer that for each j maps numbers in the interval $(a_j - 0.25p, a_j + 0.25p)$ into a_j, where $p = \min_{i \neq j} | a_j - a_i |$, and w is the output of the classifier.

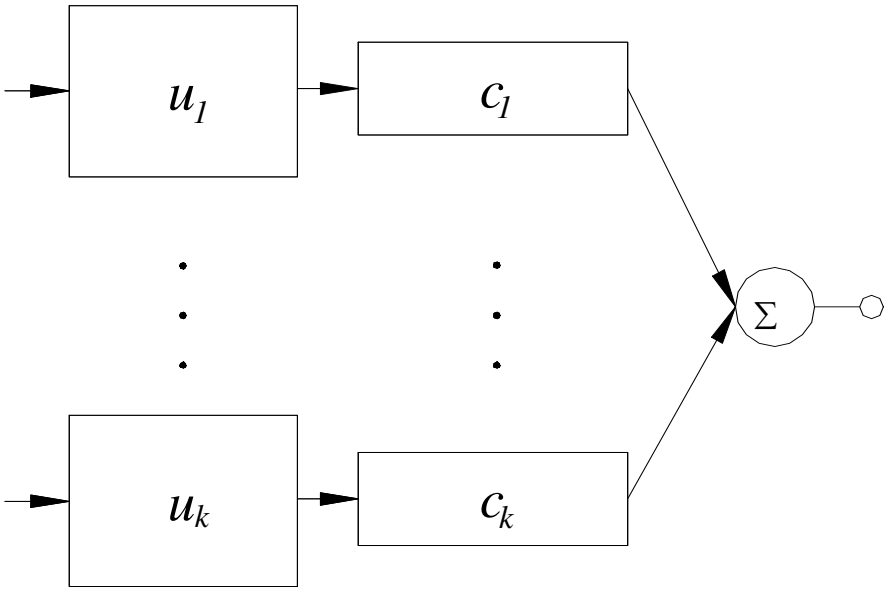


Figure 1.2: A structure for h

It can be shown in Figure 1.2 that h can be synthesized as a parallel combination of simple structures followed by a summing device. In the figure, each block labeled c_j denotes multiplication by a real number c_j and every u_j block denotes the taking of a real-valued function u_j defined on the reals.

1.4 Clustering

Clustering is a topic closely related to classification. In clustering, the existence of predefined pattern classes is not assumed, the actual number of classes is unknown, or the class memberships of the vectors are generally unknown. The task of clustering process is therefore to group the feature vectors to clusters in which the resemblance of the patterns is stronger than between the clusters [65]. It is a fundamental means for multivariate data analysis widely used in numerous applications, especially in pattern recognition [109]. Clustering techniques have been investigated extensively for decades. The existing approaches to data clustering include statistical approach (e.g., the k-means algorithm), optimization approach (e.g., branch and bound method), simulated annealing technique and neural network approach.

Zeng et al [109] proposed a method where clustering starts with an estimate of the local distribution. First, small clusters are constructed, called seed clusters, according to the local distribution. Then, those clusters whose distributions are consistent with each other are merged together. The clustering process moves from small, local data points towards bigger clusters with a specified probability density function. Their merging is performed in order to reduce the complexity of data representation as well as to provide statistically supported generalization ability for classification.

1.4.1 Clustering methods

According to Fukunage [29], the existing clustering methods can be divided into two approaches. One is the parametric approach, and the other one is the nonparametric one.

1.4.1.1. Parametric Approach

The widely used k-means method is one example of the parametric approach. In this method, a criterion is given and data is arranged into a pre-assigned number of groups to optimize the criterion. Another kind of parametric approach assumes some mathematical form to express the data distribution, such as summation of normal distributions. Both parametric methods are developed based on the view of the global data structure and are ready to be improved in combination with other optimization methods. However, the performance of these methods depends on the assumption of the cluster number, which is hard to estimate beforehand in real applications.

1.4.1.2. Nonprametric Approach

There are many classification tasks in which no assumptions can be made about the characterizing parameters. Non parameteric approaches are designed for those tasks.

In the nonparametric approach, data are grouped according to the valleys of the density function, such as in the valleyseeking method. This method does not require knowledge of the number of clusters beforehand. But since its performance is, in general, very sensitive to the control parameters and the data distribution, its application is limited.

1.5 Classification techniques

There are a large number of classifiers that have been presented in the literature such as: Nearest-neighbor, Bayesian, polynomial discriminant, neural networks, self-organizing map, decision tree, etc. In the following sections, some classification techniques introduced in the literature are presented.

1.5.1 Neural Network

Neural Network (NN) has been a powerful tool in engineering area after the development in computer technology [101]. They have shown very good performance in case of noisy signals. At the same time, they are easy to construct [86]. They have been employed and compared to conventional classifiers for a number of classification problems. The results have shown that the accuracy of the NN approach is equivalent to, or slightly better than, other methods. Also, due to the simplicity and generality of the NN, it leads to classifiers that are more efficient [46, 113].

1.5.1.1. Description

The NN consist of neurons connected by weighted links. The weights are adjusted during the training of the network to achieve human-like pattern recognition [17]. The choice of the learning algorithm, weight initialization, the input signal representation and the structure of the network is very important [36]. The number of hidden neurons and layers must be sufficient to provide the discriminating capability required for the given application. However, if there are too many neurons, the neural network will be incapable of generalizing between input patterns when

there are minor variations from the training data [26]. Furthermore, there will be a significant increase in cost and in the time required for training.

1.5.1.2. Characteristics

As reported in the literature, NN classifiers possess unique characteristics, some of which are:

1) NN classifiers are distribution free. NN's allow the target classes to be defined without consideration to their distribution in the corresponding domain of each data source [6]. In other words, using neural networks is a better choice when it is necessary to define heterogeneous classes that may cover extensive and irregularly formed areas in the spectral domain and may not be well described by statistical models.

2) NN classifiers are important-free. When neural networks are used, data sources with different characteristics can be incorporated into the process of classification without knowing or specifying the weights on each data source. Until now, the importance-free property of neural networks has mostly been demonstrated empirically [10]. Efforts have also been made to establish the relationship between the importance-free characteristic of neural networks and their internal structure, particularly their weights after training [113]. In addition, NN implementations lend themselves to reduced storage and computational requirements.

3) NN classifiers are capable of forming non-linear decision boundaries [46] and they do not require the decision functions to be given in advance [36]. This makes them flexible in modeling real world complex relationships [110] and they can approximate any function with arbitrary accuracy.

1.5.1.3. Training

The NN learning is generally classified as unsupervised or supervised. Unsupervised methods determine classes automatically, but generally show limited ability to accurately divide terrain into natural classes [44, 47].

Supervised methods have yielded higher accuracy than unsupervised ones, but suffer from the need for human interaction to determine classes and training regions. Backpropagation is one of the algorithms used in training the supervised neural network. It is based on linear model (steepest descent) and it is less computationally expensive [36]. It has been shown in the literature that the multilayer perceptron trained using backpropagation approximates the Bayes optimal discriminant functions for both two-class and multi-class recognition problems [88]. However, there are some drawbacks that are associated with backpropagation such as convergence to local minimum and the absence of specific methods for determining the network structure [17].

1.5.2 Bayesian Classifier

Bayesian method is one of the classification techniques that have been introduced in the literature. It provides the optimal performance from the standpoint of error probabilities in a statistical framework [36]. The success of the Bayesian methods depends on the assumptions used to obtain the probability model [19, 110]. This makes them unsuitable for some applications such as Synthetic Aperture Radar (SAR) image classification based on a feature space comprising texture measures [93]. However, they have been applied to Artificial Neural Networks inorder to regularize training to improve the performance of the classifier [63].

Grouping images into meaningful categories using low-level visual features is an important problem that has been addressed in the literature [107]. Vailaya [103] et al. have used binary Bayesian classifiers to capture high-level concepts from low-level image features for images belonging to the same class. They have also considered the hierarchical indexing scheme for classification of vacation images.

1.5.3 Classification Decision Tree

The decision trees are considered as one of the most popular classification approaches due to their accuracy and simplified computational properties [32, 95, 111]. Moreover, they are fast in training [14]. They are capable of performing non-linear classification [4] and they do not rely on statistical distribution. This yields to successful applications in many fields such as remote sensing data [93].

The tree is composed of a root node, intermediate nodes and terminal nodes. The data set is classified at each node according to the decision framework defined by the tree [48]. It starts with a coarse classification, and then followed by a fine classification where finally each group contains only one signal.

Classification decision trees have the advantages of employing more than one feature. Each feature provides partial information about the signal. The combination of such features can be used to obtain accurate recognition decision [91]. There are more than one decision tree that can be used for a given example. But the smaller the decision tree, the superior it becomes [100].

A large number of methods have been proposed in the literature for the design of the classification tree. Classification and Regression Trees (CART) is one of the approaches that have achieved high popularity [32]. It was developed during the years 1973 through 1984 [4]. It has the advantage of constructing classification regions with sharp corners. However, it is computationally expensive [32]. In this approach, splitting continues until terminal nodes are reached. Then, a pruning criterion is used to sequentially remove splits [4]. Pruning can be implemented by using different data than those used in training. The main advantages of pruning is reducing the size of the decision tree [93] and hence reducing the classification error [12] and avoiding both overfitting and underfitting.

Most of the pruning methods proposed in the literature are based on removing some of the nodes of the tree. Kijsirikul et al. [57] have introduced a pruning method which employs neural networks, trained by backpropagation algorithm, to give weights to nodes according to their significance instead of completely removing them.

1.5.4 Vector quantization

Vector quantization is a powerful technique for data compression. Recently, it has been used to simplify image processing tasks such as halftoning, edge detection [22], speech and image recognition [33] and enhancement classification.

Vector Quantization and Classification can be combined because both techniques can be designed and implemented using methods from statistical clustering and classification trees [22]. They can be implemented with a tree structure that greatly reduces the encoding complexity [40]. It has been shown that if an optimal vector quantizer is obtained, under certain design constraints

and for a given performance objective, no other coding system can achieve a better performance. This approach has several advantages in coding and in reducing the computation in speech recognition [38].

One of the most widely used algorithms is the Lloyd algorithm. It improves a codebook by alternately optimizing the encoder for the decoder and the decoder for the encoder [22].
Linear Vector Quantization (LVQ) has been used to classify the various kinds of signals. The reasons to use the LVQ are that it can process the unsupervised classification and treat many input data with small computational burden [61]. In other words, it can treat high dimensional input and has simple learning structure.

A LVQ is composed of two layers; a competitive layer that learns the feature space topology and the linear layer that transforms classes into target classes. It can be used as a method for training competitive layers of the unsupervised neural network model developed by Kohonen, called Self-Organizing Map (SOM), in a supervised manner. It also has the advantage of increasing the classification accuracy of the SOM network [36].

1.5.5 Hidden Markov Models

The Hidden Markov Model is a finite set of states, each of which is associated *w*ith a probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome o*r* observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are ``hidden" to the outside; hence the name Hidden Markov Model The Hidden Markov Models (HMM) are known to classify the data based on the statistical properties of the input. HMM are expected to do the job of recognition by extracting fuzzy features of the face in question and comparing it with the known (stored) one [66]. To use HMMs for classification of unknown input data, HMM are first trained to classify (recognize) known faces. HMMs are basically 1D models. Hence, to use them for face recognition, the face images that are 2D signals, must be represented in 1D format without loosing any vital information.

1.5.6 Classification to groups of constant statistical properties

The main idea of this algorithm is to group signals into a certain number of groups that have similar statistical properties within each group. The algorithm used is called *Equal Mean-Normalized Standard Deviation Algorithm* [54]. The method finds N_1, N_2… (number of elements in the resulting groups) such that the mean-normalized standard deviation of the gains in the resulting groups is equal.

In case of having two groups, the main purpose of this algorithm is seeking an integer N_1 (the number of signals belonging to group 1) out of the N input signals such that the mean normalized standard deviation remains almost constant in all groups.

Let the mean m_i and standard deviation σ_i be defined as follows

$$m_1 = \frac{1}{N_1}\sum_{n=1}^{N_1} g_n \qquad (1.1)$$

$$m_2 = \frac{1}{N - N_1}\sum_{n=N_1+1}^{N} g_n \qquad (1.2)$$

$$\sigma_1^2 = \frac{1}{N_1}\sum_{n=1}^{N_1} (g_n - m_1)^2 \qquad (1.3)$$

$$\sigma_2^2 = \frac{1}{N - N_1}\sum_{n=N_1+1}^{N} (g_n - m_2)^2 \qquad (1.4)$$

N_1 is chosen such that $q_1 = q_2$ (1.5)

Where the coefficient of variant $q_i = \frac{\sigma_i}{m_i}$ (1.6)

If there is no integer N_1 that solves (1.5), the following algorithm finds N_1 which minimizes $|q_1-q_2|$.

1.5.6.1.Algorithm

1) Choose an initial value for N_1 and set the iteration number i=0. Also choose i_{max} as an upper limit on the number of iterations.
2) Compute q_1 and q_2 using (1.6) and set i=i+1.
3) If ($|q_1-q_2|/q_1$) < δ or i>i_{max} , stop. Otherwise,

If $q_1<q_2$ set $N_1=N_1+\Delta N_1$ (1.7)

If $q_1>q_2$ set $N_1=N_1-\Delta N_1$ (1.8)

then go to step 2.

1.5.7 The K-MEANS clustering algorithm

The K-MEANS algorithm is one of the classification techniques that have been introduced in the literature. It is partially supervised because the number of clusters is predefined.

In order to clusters *M* feature vectors into *G* clusters, assume a data set of *M* vectors, v_i, *i=1, 2 …M* of dimensionality 1 X N.

1.5.7.1.Algorithm

1) Select *G* such that $G < M$ (1.9)

2) Define the clusters centers c_g , g = 1,2…G (1.10)

3) Associate each of vectors v_i to the closest center according to a distance measure. There are several distance measures defined in the literature. Euclidean distance is often used because of its simplicity.

4) The Euclidean distance D between two vectors $v_1 = \{v_{1,1}, v_{1,2} \ldots v_{1,N}\}$ and $v_2 = \{v_{2,1}, v_{2,2} \ldots v_{2,N}\}$ is defined as

$$D = \frac{1}{N} \sum_{i=1}^{N} \left(v_{1,i} - v_{2,i} \right)^2 \qquad (1.11)$$

5) The new cluster center c_g is the average of all vectors that belongs to this cluster.

$$\boldsymbol{c}_g^{new} = \frac{1}{N_g} \sum_{i=1}^{N_g} v_i \qquad (1.12)$$

where N_g is the numbers of vectors belonging to g^{th} cluster.

6) The algorithm is repeated until the change in centers is not significant.

1.6 Data Fusion

Data Fusion plays a good role in medical imaging, remote sensing, etc. Due to the availability of the large amount of data acquired by different types of sensors, it is mandatory to develop effective data fusion techniques capable of taking advantage of such multi-source and multi-temporal characteristics [13].

According to Kundar et al. in [62] multi-sensor data fusion refers to the acquisition, processing, and synergistic combination of information from various knowledge sources to provide a better understanding of the situation under consideration.

In remote sensing community, data fusion is defined as follows [82]: "data fusion is a formal frameworks in which are expressed means and tools for the alliance of data originating from different sources. It aims at obtaining information of greater quality; where the exact definition of "greater quality" will depend upon the application." The remote sensing community also classifies the data fusion techniques into three groups:

1) Data Level fusion: Combination of raw data from all sensors.
2) Feature level fusion: Extraction, combination and classification of feature vectors from all sensors.
3) Decision level fusion: Combination of outputs of the classifications achieved on each single source.

Several methods have been proposed in the literature for the classification of multi-sensor/multi-sources images. They are mainly based on statistical, symbolic and neural-network approaches.

"Stacked-vector" approach is one of the simplest statistical methods [13]. In this approach, a vector composed of features extracted from different sensors represents each pixel. The Dempster-Shafer evidence theory is used to classify multi-source data by taking into account the uncertainties related to the different data sources involved.

1.7 Ensemble of Classifiers

There are two ways to increase the accuracy of a pattern recognition system. One is to improve the performance of a single classifier; another is to combine the results of multiple classifiers by employing decision combination. It has been verified experimentally in the literature that multiple classifiers have better performance than single classifiers if they are selected carefully and the combining algorithm retains the advantages of each individual classifier and avoids its weakness [47, 48, 49].

This is due to two reasons [11]:

1) The risk of choosing the wrong class is lower.
2) Individual classifiers can be trained on different types of features of the same data, and the multiple classifiers can weight the classifiers based on the characteristics of the different features.

Bagging and Boosting are two popular methods that have shown great success when using ensemble of classifiers [14]. In bagging, all classifiers share with equal votes in the final decision. On the other hand, in boosting, each classifier shares in the final decision by different weight according to its accuracy in the problem under consideration.

The two most important issues in designing a multiple classifiers system are Classifier selection and decision combination.

1.7.1 Multiple Classifiers Structures

A number of image classification systems based on the combination of the outputs of a set of different classifiers have been proposed in the literature. Different structures for combining classifiers can be grouped as follows [49, 90]:

1) Parallel Structure
2) Pipeline structure
3) Hierarchical structure

For the parallel structure, the classifiers are used in parallel and their outputs are combined. In the pipeline structure, the system classifiers are connected in cascade [35, 56]. The hierarchical structure is a combination of the structures in i & ii above.

1.7.2 Decision Combination

The combination methods proposed in the literature are based on voting rules, statistical techniques, belief functions and other classifier fusion schemes such as [84,107]:

1) *Random decision*: the system selects randomly the final class among the classes selected by the individual classifiers.

2) *Majority decision*: each classifier contributes with equal weight to the final decision, and the final class is the class selected by the most of the individual classifiers.

3) *Hierarchical decision*: each classifier has a different weight in the final decision. This weight may be fixed or changeable.

Xu et al [107] proposed four approaches for solving pattern recognition problems: Average Bayes classifier, voting methods, Bayesian formalism and Dempster-Shafer formalism. Previous methods for classifier combination include intersection of decisions regions, voting methods, prediction by top choice combinations. In these methods, only the top choice from each classifier is used, which is usually sufficient for problems with a small number of classes. The examination of the strengths and weaknesses of each method leads to the problem of determining classifier correlation that is the central issue in deriving an effective combination method.

The decisions of the classifiers in Ho et al. [48, 49] are represented as the rankings of classes. The rankings contain more information than unique choices for a many-class problem. For a mixture of classifiers of various types, numerical scores such as distances to protoypes, values of arbitrary discriminate, estimates of posterior probabilities, and confidence measures are not directly usable because of the incomparability of their scales and, in some cases, inconsistency across different instances of a problem. Therefore, combination methods based on rankings are more general and applicable to a mixture of classifiers of arbitrary types [48, 49]. The rankings can be combined by the methods that either reduce or re-rank a given set of classes.

1) Class reduction

In class set reduction the objectives is to extract a subset from a given set of classes such that the subset is as small as possible yet still contains the true class.

2) Class reording

In class set reording, the objective is to derive a consensus ranking of the given classes, such that the true class is ranked as close to the top as possible. Three methods based on the highest rank, the Borda count and logistic regression were proposed for class set re-ranking.

Class reduction and class reording are equivalent under special conditions:

1) If it is required that the result set derived by a reduction method always contains only one class, then this is the same as requiring a reordering method to rank the true class always at the top.
2) If the rankings derived by reordering methods are so good that the true class is always ranked above a certain position, then it is always possible to include the true class in a neighborhood up to that position.

The two approaches can be applied to the same problem, so that the set of classes may first be reduced and then reranked, or first reranked and then reduced to a small neighborhood near the top of the ranking.

The advantage of the combination methods based on ranking is the classifier independent property, i.e these methods can weaken the inconsistency of the scales of different classifiers. On the contrary, the disadvantage of these methods is the tie problem will occur if more than two classes posses the same ranking [66].

Kittler et al [59] developed a common theoretical framework for combining classifiers. They derived the product rule, sum rule, max rule, min rule and median rule to take the product, sum, maximum, minimum and median values of the a posteriori probabilities p (w_k /x_i) (The probability that an input pattern with feature vector x_i is assigned to class w_k).

There are different classification techniques that have been proposed in literature based on the combination of classifiers [31] such as combination of an ensemble of neural networks and the k-nearest neighbor (K-NN) decision rule.

1.7.3 Ensemble of Neural Networks

In the field of pattern recognition, the combination of an ensemble of neural networks has been proposed to achieve image classification systems with higher performance in comparison with the best performance achievable employing a single neural network. This has been verified experimentally in the literature [34, 59]. Also, it has been shown that additional advantages are provided by a neural network ensemble in the context of image classification applications. For example, the combination of neural networks can be used as a "data fusion" mechanism where different NN's process data from different sources [67]. Ideally, the combination function should take advantage of the strengths of the individual classifiers, and avoid their weaknesses, to improve classification accuracy.

Ueda [102] has presented a method for linearly combining multiple neural network classifiers based on statistical pattern recognition theory. In his approach, several neural networks are first selected based on which works best for each class in terms of minimizing classification errors.

Then, they are linearly combined to form an ideal classifier that exploits the strengths of the individual classifiers. In this approach, the minimum classification error (MCE) criterion is utilized to estimate the optimal linear weights. In this method, the problem of estimating linear weights in combination is reformulated as a problem of designing linear discriminate functions using the minimum classification error (MCE) discriminate [55].

Let x be an observation vector when the task is to assign x to one of K classes. A decision rule in terms of discriminate functions is written as follows:

Decide $x \in w_k$ if

$$f^{(k)}(x) = \max_j f^{(j)}(x) \qquad (1.13)$$

Where $f^{(k)}(\,)$ is the discriminate function for class w_k

In the case of a neural network classifier, the k^{th} output unit corresponds to the discriminate function for w_k.

Let $f_m^k(x)$ denotes the output of k^{th} output unit of the m^{th} neural network for some input x after the m^{th} neural network has been trained.

Note that $0 < f_m^k(x) < 1 \qquad (1.14)$

Then we define the combined discriminate function for each class as linear combination of all M discriminate functions.

$$f_{com}^{(k)}(x\ ; a^{(k)}) = a^{(k)^T}\ f^{(k)}(x) \qquad (1.15)$$

$$\alpha^{(k)} = (\alpha_1^{(k)} ..... \alpha_M^{(k)})^T \in R^M \qquad (1.16)$$

$$f^{(k)}(x) = (f_1^{(k)}(x).... \quad f_M^{(k)}(x))^T \in R^M \qquad (1.17)$$

T denotes transpose, k=1, 2… K (1.18)

Decide $x \in w_k$ if

$$f_{com}^{(k)}(x) = \max_j f_{com}^{(j)}(x) \qquad (1.19)$$

Previous work showed that neural network ensembles are effective only if the neural networks forming them make different errors [64, 92]. As an example, Hansen and Salamon [44] showed that neural networks combined by the "majority" rule can provide increases in classification accuracy only if the nets make independent errors. Unfortunately, the reported experimental results pointed out that the creation of error- independent networks is not a trivial task, in the sense that, thought different in terms of their weights, architectures and other parameters, nets

can exhibit the same pattern of errors basically because of the so-called problem of network "symmetries".

In the neural network field, several methods for the creation of ensemble of neural networks making different errors have been investigated. Such methods basically lie on "varying" the parameters related to the design and training of neural networks. In particular, the main methods in the literature can be included in varying one of the following categories: varying the initial random weight, varying the network architecture, varying the network type, and varying the training data. The capabilities of the above methods were experimentally compared to create error-independent networks. It was concluded that varying the net type and the training data are the two best ways for creating ensembles of networks making different errors. However, it has been noted that neural network ensembles could be created using a combination of two or more of the above methods. Therefore, neural network researchers have stated to investigate the problem of the engineering design of neural network ensembles. The proposed approaches can be classified into main design strategies:

1) The "direct" strategy;
2) The "overproduce and choose" strategy.

The first design strategy is aimed to generate an ensemble of error-independent nets directly. Differently the "overproduce and choose" strategy is based on the creation of an initial large set of nets and the subsequent choice of the subset of the most error-independent nets.

CHAPTER TWO: FEATURE EXTRACTION

Feature extraction is extracting the information from the raw data that is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class pattern variability [44]. Thus, we obtain a feature space of reduced dimensions and complexity. This is necessary due to the technical limits in memory and computation time.

The features can be extracted from the signal or from system modeling based on this signal in the time, frequency, or joint time-frequency domain [36]. For high recognition accuracy, it is important to find a subspace in which a projection of the class means preserves the class distance such that the class separability is maintained as good as possible [24, 58, 80]. This could be achieved by mapping images to a set of coefficients (Transform Coding).

A wide verity of features have been used by researchers for classification of signals employing [105] correlation, mean, standard deviation, higher order moments, entropy, root mean square (rms), smoothness, shape, contrast, [53], etc. Relative or comparative measurements can also obtain useful features that help in classification of signals [85].

The pattern recognition system introduced in this contribution has the ability of testing many criteria over the range of parameters for each criterion. It selects the successful criteria that solve the problem under consideration.

Section 2.1 presents the important factors that must be taken into consideration when selecting the optimum set of features used in the classification process. In Section 2.2, the transforms used to develop the criteria employed by the proposed pattern recognition system are briefly given for the sake of completeness. Section 2.3 presents a brief description of the preprocessing techniques used in the implementation examples throughout this dissertation, as well as their advantages.

2.1. Factors affecting features selection

2.1.1. Cost

Cost is one of the important factors that must be taken into consideration when selecting the particular set of features employed by the pattern recognition system. The compuatation cost of features extraction differs from one type of feature to the other and from one technique to the other. The proposed pattern recognition system has the ability of comparing the different types of features and selecting the ones which are less computationally expensive.

2.1.2. Distortion

The statistical properties of the signals may be greatly affected when Signal to Noise Ratio (SNR) worsens. In other words, when the signals are corrupted by interference or the transmitting channel suffers from Doppler shifts, Rayleigh fading, or group delay. Therefore, the features should be tested over a wide range of SNR values to ensure that the statistical distributions will not be greatly affected in case of corruption.

2.1.3. Size of featrures

It is important to reduce the size of features for several reasons:

1) Storage.

2) Redundant features add nothing to the classification performance.

3) Weak features may reduce the separation between classes and hence degrade the classification performance.

2.2. Transform Coding

The goal of the transform is to decorrelate the original signal and this results in redistributing the signal energy among a small number of coefficient [101]. Discrete Fourier Transform, Discrete Cosine Transform, Walsh-Hadamard Transform, Singular Value Decomposition etc, are some of the transforms that have been introduced in the literature and will be described briefly in this Section.

2.2.1. Discrete Fourier Transform (DFT)

DFT has complex exponentials as its basis images and can compactly represent spectrally narrowband images [76]. It is a good method for analyzing stationary data. The high frequency features represent the details or noise in the signal, and the low frequency features represent the basic shapes.

Let $x(n)$ be a complex series with N samples,

The Fourier transform of the series can be expressed as in Equation (2.1)

$$X(K)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi kn/N} \qquad \text{k}=0,1,2,3\ldots\ldots\text{N-1} \qquad (2.1)$$

where *x* is a complex number.

The first sample *X(0)* of the transformed series is the DC component and it is known as the average of the input series.

Since its basis functions are infinite duration sine and cosine functions, the frequency information is global and this is not satisfactory when searching for localized features. This problem can be partially avoided by using the Short Time Fourier Transform STFT [39].

2.2.2. Discrete Cosine Transform (DCT)

DCT has the ability of information packing for a wide range of images. It provides good performance for images characterized by high correlation and small standard deviation. Moreover, it has high efficiency in coding edge areas and strong textures in the image [37]. It also has the advantage of efficient computation [23].

The 1 Dimensional (1-D) DCT transform is most easily expressed in matrix notation. For a one dimensional N points signal, let the elements of *G,* equal

$$g_{ij}=\begin{cases}\dfrac{1}{\sqrt{N}}, & \text{i}=0, \qquad 0\le \text{j}\le \text{N-1}\\ \sqrt{\dfrac{2}{\text{N}}}\cos\left[\dfrac{\pi(2j+1)i}{2N}\right], & 1\le \text{i}\le \text{N-1}, \quad 0\le \text{j}\le \text{N-1}\end{cases} \qquad (2.2)$$

For 2 dimensional signals (2-D):

$$Y = GXG^T \qquad (2.3)$$

Where [Y] is the (N x N) 2-D DCT transform of the input image [X] and [G] is the (N x N) 1-D DCT Transformation matrix.

BinDCT is a fast DCT implementation technique that has been presented in the literature [99]. It utilizes only shift and add operations and no multiplication is needed. This allows efficient implementations in terms of both chip area and power consumption.

2.2.3. Walsh Hadamard Transform (WH) and HaaR Transform

WH and Haar are more appropriate for spectrally wideband images. These transforms basis functions are square and rectangular waveforms defined over a fixed time interval. WH and Haar have appropriate spectral properties and can be developed using fast implementations [77] with the least computational complexity [50].

2.2.3.1. Walsh Transform

A commonly used notation for Walsh functions is wal(u,v), where *u* is the sequency of the functions and *v* is the sample index [25].

$$wal\,(u,v) = (-1)^{\sum_{i=0}^{n-1} u_i v_i} \qquad (2.4)$$

Where $n = \log_2 N$ and $u,v = 0,1,2\ldots..N-1$. The symbols u_i and v_i refer to the i^{th} bits in the binary representations of the integers u and v, respectively.

The 2-D Walsh transform of the input image can be written as

$$[B] = \frac{1}{N^2}[W][X][W]^T \qquad (2.5)$$

Where [B] is the (N x N) 2-D Walsh transform of the input image [X] and [W] is the (N x N) 1-D Walsh Transformation matrix.

2.2.3.2.Haar Transform

The 1-D Haar transform is most easily expressed in matrix notation. If we let the elements of G equal

$$g_{0j} = \frac{1}{N}, \qquad 0 \le j \le N-1 \qquad (2.6)$$

$$g_{ij} = \frac{1}{\sqrt{N}} \begin{cases} 2^{r/2}, & \frac{m-1}{2^r} \le \frac{j}{N} \le \frac{m-1/2}{2^r} \\ -2^{r/2}, & \frac{m-1/2}{2^r} \le \frac{j}{N} \le \frac{m}{2^r} \\ 0, & \text{otherwise} \end{cases}$$

Where $r = [\log_2 i]$ and $m = i - 2^r + 1, \quad 0 \le i \le N-1$

For 2-Dimensional signals :

$$Y = GXG^T \qquad (2.7)$$

Where [Y] is the (N x N) 2-D haar transform of the input image [X] and [G] is the (N x N) 1-D Haar Transformation matrix.

2.2.4. Singular Value Decomposition (SVD)

SVD has the property of packing energy in the least amount of coefficients for any image. It has a variety of applications in signal processing, automatic control as well as other areas but it has the disadvantage of high cost [23].

The Singular Value Decomposition (SVD) of a rectangular matrix X is a decomposition of the form

$$X = U\,S\,V^{T} \qquad (2.8)$$

where U and V are orthogonal matrices, and S is a diagonal matrix. The columns u_i and v_i of U and V are called the left and right singular vectors, respectively, and the diagonal elements s_i of S are called the singular values. The singular vectors form orthonormal bases and lead to the following relationship.

$$X\,v_i = s_i\,u_i \qquad (2.9)$$

2.2.5. Wavelet Transform (WT)

The Wavelet Transform is a powerful technique for representing data at different scales and frequencies. Daubechies presented the wavelet transform as " a tool that cuts up data or functions

or operators into different frequency components, and then studies each component with a resolution matched to its scale"[16]. There are several families of wavelets, such as the Haar, Biorthogonal, Coifflets, Daubechies, etc. The Daubechies family has been used a lot because its wavelet coefficients capture the maximum amount of the signal energy [39].

WT decomposes the signal into shifted and scaled versions of the mother wavelet ψ(t) [15, 39]. The wavelet transform of signal *s(t)* as given by [5] is shown in Equation (2.1)

$$W_s(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} s(t) \psi^* \left(\frac{t-b}{a} \right) dt \qquad (2.10)$$

The variables a and b control the the scale and the position of the wavelet respectively.

WT can be realized by finite impulse response (FIR) filters and this makes them suitable for real time applications [108]. It can also provides good resolution in the frequency domain at low frequencies and good resolution in the time domain at high frequencies [3]. This transform is capable of detecting hidden details of waveforms that might be unnoticed [39]. It has applications in many fields such as applied mathematics, filtering, getting rid of noise [9], detection of microcalcifications in digital mammograms [105], sound synthesis, computer vision, graphics [27], image enhancement, target detection, etc [16].

WT is also good for normalization because they permit different adjustments on different image scales. Normalization helps the analyst viewing the output of processed images [87].

Wavelet Transforms can be applied in different ways such as:

1) The traditional pyramid-type wavelet transform decomposes sub-signals in the low frequency channels.

2) The tree-structured wavelet transform in which the decomposition can be applied to the output of any filter h_{LL}, h_{LH}, h_{HL}, h_{HH}. This is useful for the applications where the low frequency region may not contain significant information [15].

3) The Adaptive wavelet neural network has also been introduced in the literature. It has the capability of feature extraction and target classification at the same time [112].

Wavelets can provide flexibility in the shape and form of the analyzer that studies the signal of interest. But with this flexibility comes the hard task of choosing and designing the appropriate wavelets for a given application. Especially, 2-D WT which is computationally intensive and operates on large data sets. Therefore, some researchers have proposed parallel solutions of DWT [81].

2.3.Mixed Transform

In previous sections, some single transforms are presented. It is noted that the signal may be represented efficiently by a particular transform only if the basis functions of the selected transform are similar in structure to the signal. Since signals such as speech and images consist of regions with various combinations of narrow and broadband components, it is hard to find an optimal transform that can represent these signals. However, mixed transform techniques can yield to more efficient signal representations than one transform [7] because they employ subsets of non-orthogonal basis functions chosen from two or more transform domains.

There are several mixed transform methods that have been introduced in literature and have shown promising results such as the method of Mikhael and Spanias [79], the method of Beex and DeBrunner [98], the method of Mikhael and Ramaswamy [78] and the method of Berg and Mikhael [8].

2.4.Preprocessing techniques

Preprocessing techniques play a great role in many image processing applications such as scene analysis and object recognition [94]. Different preprocessing techniques have been introduced in the literature. Edge detection is one of these techniques that has shown great efficiency in many applications.

Linear Prediction can also be considered as a preprocessing technique that can improve the classification accuracy.

2.4.1. Edge Detection

Edge detection techniques can play a good role in removing the effect of noise. Therefore, it is an important step in segmentation of noisy images, which is a difficult problem in pattern recognition [104].

The quality of any edge detection method is measured by the amount of information it carries to the following stages [70]. There are several approaches that have been introduced in literature for edge detection. Neural networks can be one of these useful tools. The thresholding nonlinearity

introduced by the bias weight and sigmoid function at the output of a node makes it suitable for edge detection. Backpropagation is one of the algorithms that can be used in training the neural network to recognize edges [97].

Vector Quantization can also be used for edge detection. It has the advantage of having lower computational complexity than the conventional facet edge detector [52].

There are some techniques that are presented in the literature that combine edge detection and encoding, together with adapted wavelet approximation on each side of the edges [21].

Some of the criteria tested by the proposed pattern recognition system are computed from the edges of the input images. The edge detection methods used in the implementation examples presented in this dissertation are Canny, Prewitt, Zerocrossing and Roberts methods. The Canny method can detect weak and strong edges. It is also less likely than other methods to be effected by noise [87,104].

2.4.2. Linear Prediction

Linear prediction is a mathematical operation where future values of a digital signals is esimaed as a linear function of previous samples. Linear Prediction is one of the popular tools used in data compression. It can also be used as a preprocessing step in pattern recognition. The images are divided into sub-images. Then each sub-image is represented by the filter coefficients $\mathbf{a_{ij}}$**.** These coefficients are projected into different transform domain. Several criteria can be developed from the features extracted from these domains. Based on the selected criteria and

classification technique, images are clustered into a particular number of groups according to the problem under consideration.

Figure 2.1 shows a simple demonstration of a linear predictor.

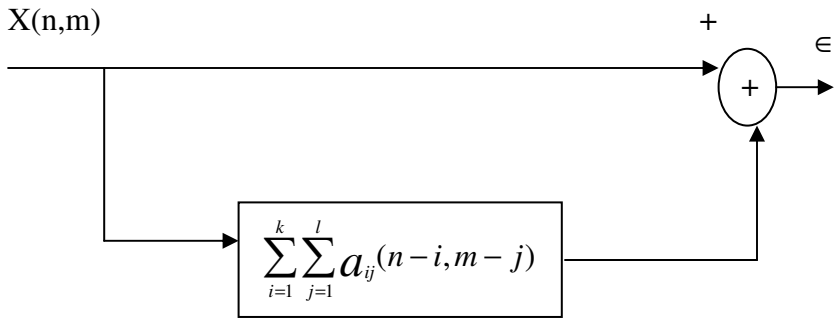


Figure 2.1 Linear Predictor

CHAPTER THREE: PROPOSED PATTERN RECOGNITION SYSTEM

3.1. Introduction

We propose a pattern recognition technique that can be designed to have evolutionary learning by developing the features and selecting the criteria that are best suited for the recognition problem under consideration. It is conjectured that, ultimately, it will be capable of recognizing an enormously large number of patterns by virtue of the fact that it analyzes the signals in different domains and explores the distinguishing characteristics in each of these domains. In other words, this approach uses available information and extracts more characteristics of the signals to be recognized by projecting the signal in different domains. Preprocessing techniques such as edge detection techniques can play a good role in enhancing the pattern recognition process. Many criteria are developed from the features extracted from the projection of the original and preprocessed signals in different domains. Based on the selected set of criteria and according to the classification technique used, the signals are grouped into a particular number of groups. Grouping of signals can be performed in parallel or in cascade. Finally, each signal will be identified by a composite index according to the group numbers throughout the classification process.

We can summarize the recognition process in 2 steps:

1) Feature extraction, which is described in detail in chapter 2, followed by criteria development and selection which is presented in this chapter

2) Classification of signals based on the developed criteria and according to the selected classification technique.

3.2. Criteria Selection

Feature extraction and criteria selection is the first step in the pattern recognition process. The signal projection, in each appropriately selected transform domain, reveals unique signal characteristics. The criteria in the different domains are properly formulated and their parameters adapted to obtain classification with desirable implementation properties such as speed and accuracy in case of noisy or corrupted data. Enormous number of criteria can be developed from the features extracted from each domain. For example: the ratio between the maximum and minimum values, the average value, the energy, the momentum, the standard deviation, the summation of a certain number of features, and so on.

The proposed pattern recognition technique automatically tests many criteria over the range of parameters for each criterion. It selects the successful criteria that solve the problem under consideration. More than one criterion can have the ability of grouping the signals to the same groups. Therefore, a voting scheme is employed such that the criteria of similar performance contribute to the final decision by a specific weight according to its accuracy. Moreover, the large number of available criteria enables the system to recognize different types of signals.

Many classification criteria can be developed to greatly enhance the performance of the classifier by exploiting:

1) Structure type.
2) Criteria selection and formulation from the information in the different domains.

3.3. Classification

The recognition process can be considered a special case of classification, at the classifier output, when each classified group contains only one signal. Signals can be recognized by either using a single stage or multi-stage system. In the proposed pattern recognition system, the N input signals are recognized using multi-stage system. Two cases are examined throughout this dissertation:

1) The parallel structure where the classifiers are connected in parallel.
2) The cascaded structure where the classifiers are connected in cascade.

In both parallel and cascaded structures, a potentially successful criterion i, with its selected values of the parameters, in a particular domain, clusters the input signals to each classifier in each stage into a number of distinct non-overlapping clusters. The cluster index, according to the i^{th} criterion, is denoted $c_{i,}$ where c_i = 1, 2, 3… $g_{i,}$, and g_i is the number of groups using the i^{th} criterion. Corresponding to a number n of selected criteria, i takes the values 1, 2… or n.

Finally, each signal will be identified by a composite index representing the cluster index it gains from passing through the different stages.

By studying different examples and using different number of groups, it was noticed that as the number of groups at each stage decreases, the immunity to noise increases. Because when the number of groups decreases, the Euclidian distance between the centers of any two groups increases.

Different classification techniques are examined to cluster signals at each node, at each stage, of the system such as: Vector Quantization, Neural networks, clustering signals such that the statistical properties are constant in all groups, clustering signals to groups such that the gap between groups are maximum, etc.

3.4. The Proposed Pattern Recognition System Structures

Figure 3.1 presents a general structure of the proposed pattern recognition system.
In this Section, two possible structures, classifiers connected in parallel and classifiers connected in cascaded are described in details. Some special cases and their advantages are also given.

In some cases, the pattern recognition system can be implemented as a combination of both parallel and cascaded classifiers according to the problem under consideration.

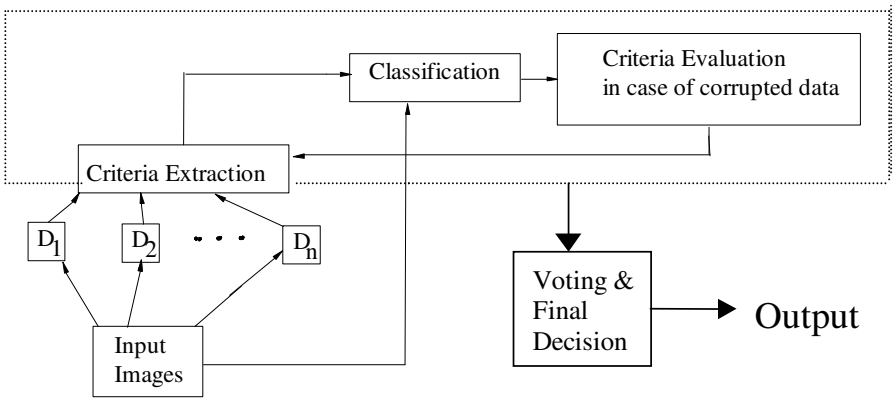


Figure 3.1: The proposed Pattern Recognition System

3.4.1. Parallel structure

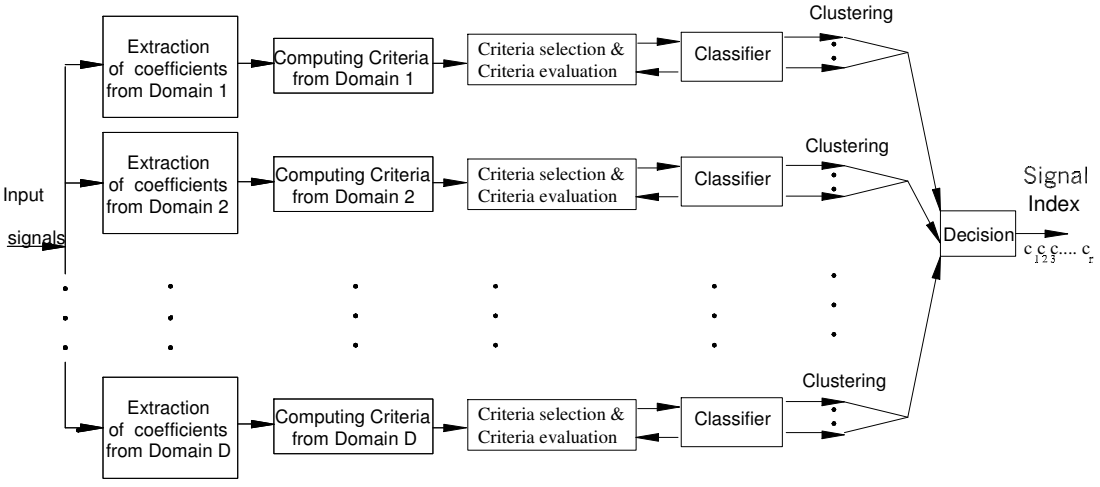


Figure 3.2: Parallel implementation of the proposed pattern recognition system

Figure 3.2 shows a parallel structure of the proposed pattern recognition system.

The original and preprocessed signals are projected into different transform domains in the first stage.

In the second stage, a large number of criteria are computed from the features extracted from the different domains.

In the following stage, each set of criteria is evaluated, in case of noisy signals, until the optimum set of criteria is selected.

In each branch, the signals are grouped into a certain number of groups based on the selected set of criteria and the employed classification technique.

In case of having N input signals and n criteria

$$N \leq \prod_{i=1}^{n} g_i \qquad (3.1)$$

g_i is the number of groups according to the i^{th} criteria

Finally each signal is recognized by a unique composite index according to the group number in each branch.

3.4.2. Cascaded structure

Figure 3.3 presents the general cascaded structure, decision tree, of the classifier employed by the proposed pattern recognition system.

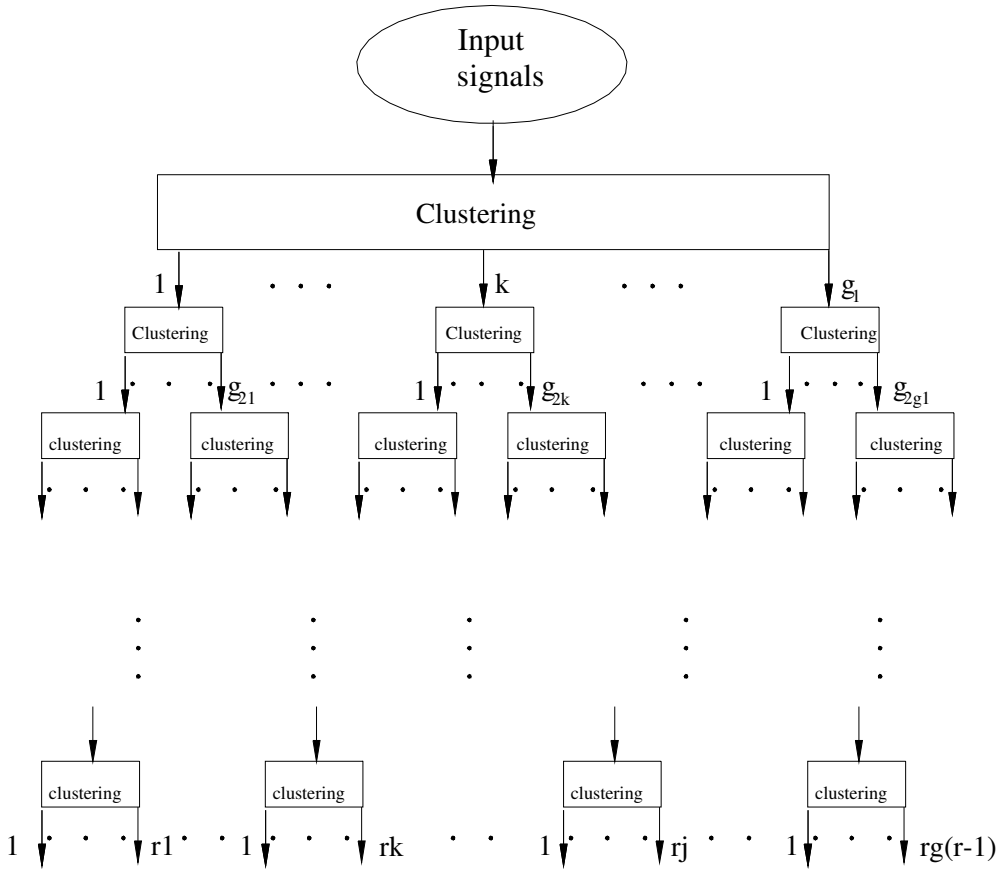


Figure 3.3: The cascaded structure of the classifier employed by the system

The proposed pattern recognition process, in case of using a cascaded structure for the classifier, can be described as follows:

In the first stage, the original and preprocessed signals are projected into a number of transform domains for extracting different types of features representing the signals in these domains. A large number of criteria are developed from the features extracted from these domains.

According to the selected set of criteria satisfying certain conditions such as simplicity, high accuracy, speed, etc, the signals at each stage are divided into a certain number of groups. There is no fixed rule to obtain the number of groups resulting from clustering the input signals, at each node, at each stage of the proposed classification structure. The number of groups will differ according to the problem under consideration.

The total number of criteria employed by the proposed system to recognize N signals in a binary cascaded structure is equal to N-1.

After selecting the optimum set of criteria that can be employed by the system to cluster the signals at each stage of the classification structure, the system keeps searching all available criteria to select more sets of criteria that give the same output at each node, at each stage.

A voting scheme is used such that each criterion will contribute to the final decision by a certain weight based on its accuracy.

When the classification process is completed, each signal is represented by a unique composite index, corresponding to the signal path through the decision tree, from the input to one of the terminal nodes of the tree.

3.4.3. Binary Structure

This is a special case of the cascaded classification structure. It is called a binary decision tree. The signals at each node, in each stage, are binary clustered. The main advantage of binary clustering is the high immunity to noise. However, the number of stages required to recognize any input signal is increased.

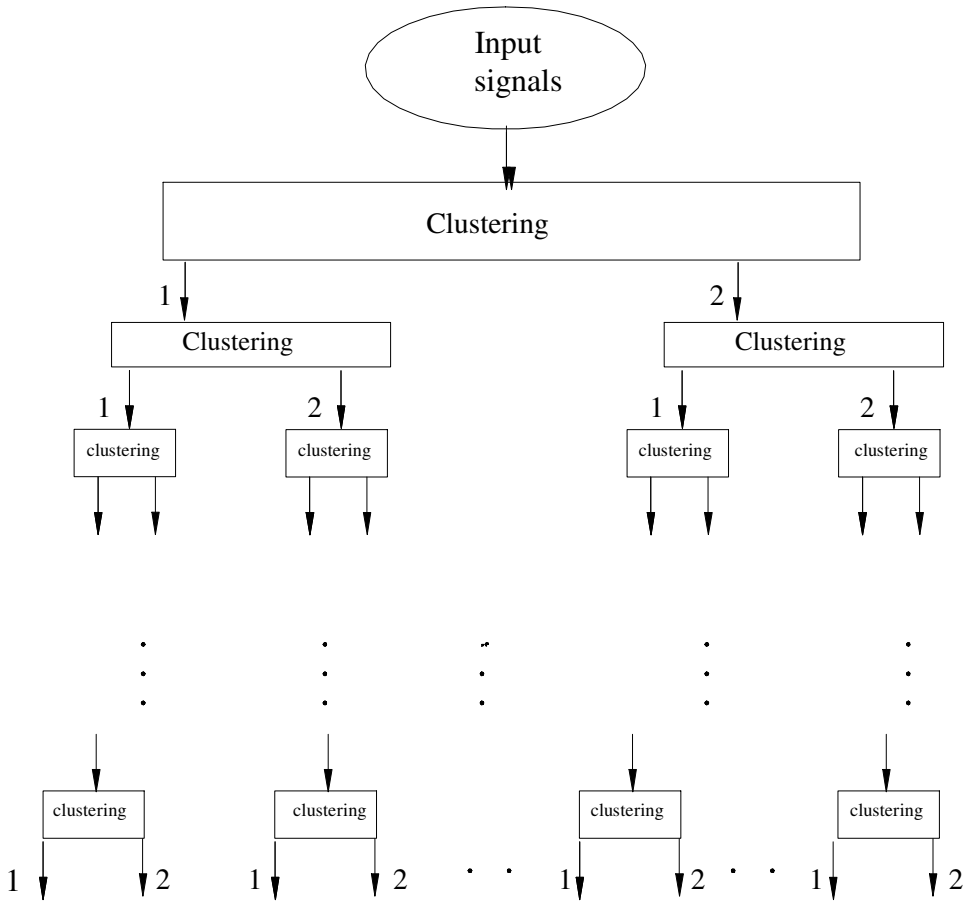


Figure 3.4: Binary Classification Structure

The number of stages needed to recognize any input signal depends on the nature of the input data used in training. This yields to two cases:

3.4.3.1.Signals of different probability of occurrence

The input signals are of different probabilities of occurrence. In other words, some of the signals have a high probability of occurrence. Therefore, it is more convenient to use the minimum possible number of criteria that can distinguish them from the other signals. On the contrary, some signals have a low probability of occurrence. So, more criteria are employed in the recognition of these signals.

Example

This simple example, 15 facial images (Olivetti Research Laboratory ORL), illustrates the first case where the probability of occurrence differs from one image to the other.

The input images to each node of the classification tree are clustered into two groups such that one image is distinguished from the other input images to this node.

The total number of criteria used to recognize all 15 images equal 14 criteria.

The number of stages needed to recognize any image depends on the probability of occurrence of this image. This can be shown in the tree structure in Figure 3.5. In this example, two mathematical transforms have been used; Discrete Cosine Transform and Singular Value Decomposition. Many criteria have developed from the features extracted from these domains.

For example:

1) (A: DCT 333 lowest band) which means 3 criteria are computed: the summation of 3 x 3 low frequency components, the summation of the following 3 x 3 mid frequency components

and the summation of the following 3 x 3 high frequency components. The criteria selected by classification unit A is the summation of the 3 x 3 low frequency components.

2) (F: SVD101010 mid band) which means 3 criteria are developed: the summation of the 10 largest singular values, the summation of the second 10 largest values and the summation of the third 10 largest values. The criteria selected by classification unit F is the summation of the second 10 largest singular values.

Table 3.1 shows that the number of stages required to recognize any image differs according to its probability of occurrence.

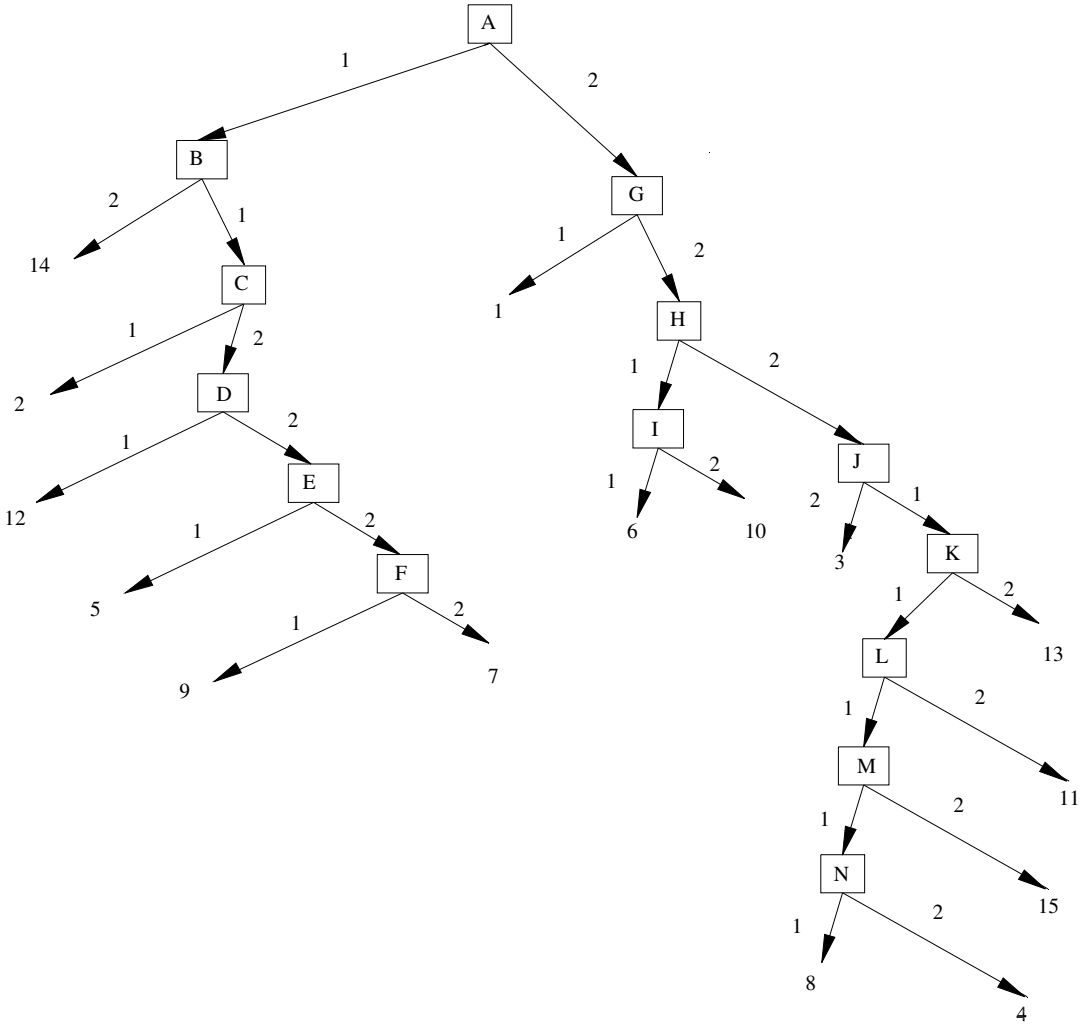


Figure 3.5: Recognition of 15 facial images using 14 criteria

A: DCT 333 lowest band, B: DCT 433 lowest band, C: DCT 333 mid band,

D: DCT 343 mid band, E: DCT 353 mid band, F: SVD 101010 mid band,

G: DCT 433 lowest band, H: DCT 533 lowest band, I: SVD 101010 highest band. J: DCT

553 mid band, K: DCT 533 mid band, L: DCT 333 mid band,

M: DCT 353 mid band, N: DCT 455 mid band.

Table 3.1

The number of stages required to recognize 16 facial images

Image number	**Number of stages**
1,14 (highest probability of occurrence)	2
2	3
3,6,10,12	4
5,13	5
7,9,11	6
15	7
4,8 (lowest probability of occurrence)	8

3.4.3.2.Signals of equal probability of occurrence

The probability of occurrence of the input signals is unknown. Consequently, no particular ordering in grouping the signals is beneficial for computational reduction. Therefore, the selected technique employs the minimum possible number of criteria to recognize any of the input signals. The set of criteria used, in the recognition process, differs from one signal to the other. However, the number of criteria used is almost the same for all signals.

Let the maximum number of stages (criteria) used to recognize any signal of the **N** input signals be **S.**

S is the approximation of $\frac{\log(N)}{\log(2)}$ to the nearest highest integer. (3.2)

Example 1

In this example, the proposed pattern recognition system is trained to recognize synthetic aperture radar (SAR) image chips and the associated JPEG files of the 2S1, BRDM-2, BTR-60, D7, T62, ZIL-131, ZSU-23/4, and SLICY, Figure 3.6. The imagery in this example was collected as part of the MSTAR Data Collection #1, Scene 1 and as part of the MSTAR Data Collection #2, Scenes 1, 2, and 3. The data was collected by Sandia National Laboratory (SNL) using the STARLOS sensor.

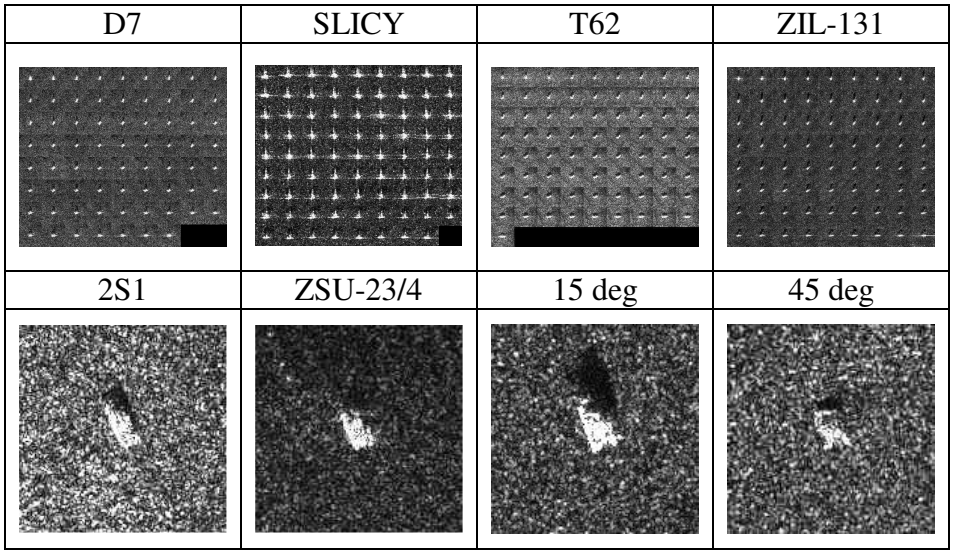


Figure 3.6: Eight radar images

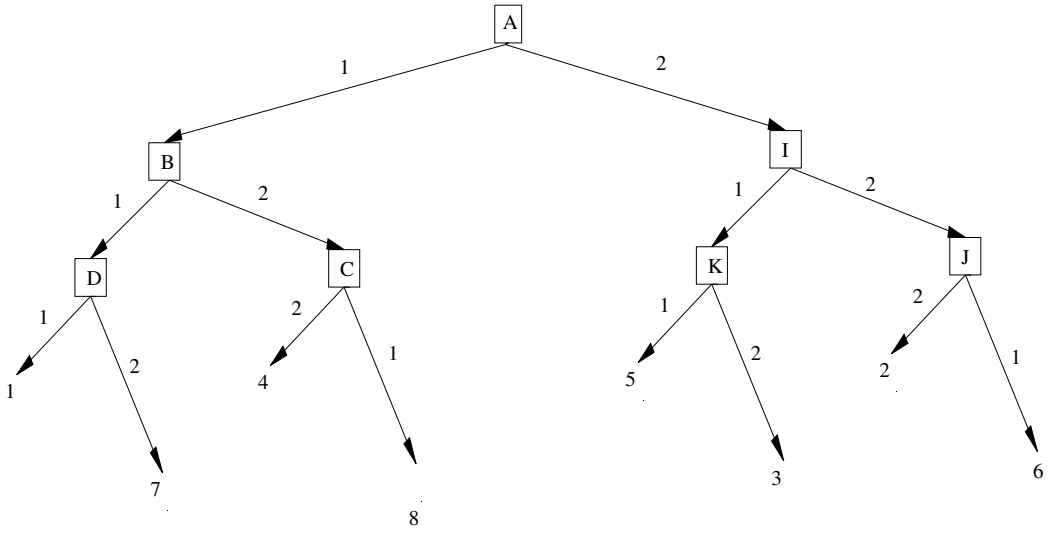


Figure 3.7: Recognition of 8 images using 7 criteria

- The 8 images are recognized successfully using the tree structure in Figure 3.7.
- Any image of the 8 images can be recognized using 3 criteria.
- When the system was tested by noisy images, up to 60 Gray level white Gaussian noise were added to the original images, all images were recognized correctly.

Example 2

15 facial images (Olivetti Research Laboratory ORL) are of unknown probability of occurrence .and it is required to recognize any of these images with the minimum possible number of stages.

The maximum number of stages required to recognize any of the 15 images is approximation of

$$\frac{\log(15)}{\log(2)}$$

Image 13 can be recognized when using 3 criteria. Any other image of the remaining *7* images can be recognized by using 4 criteria.

The tree structure used is demonstrated in Figure 3.8.

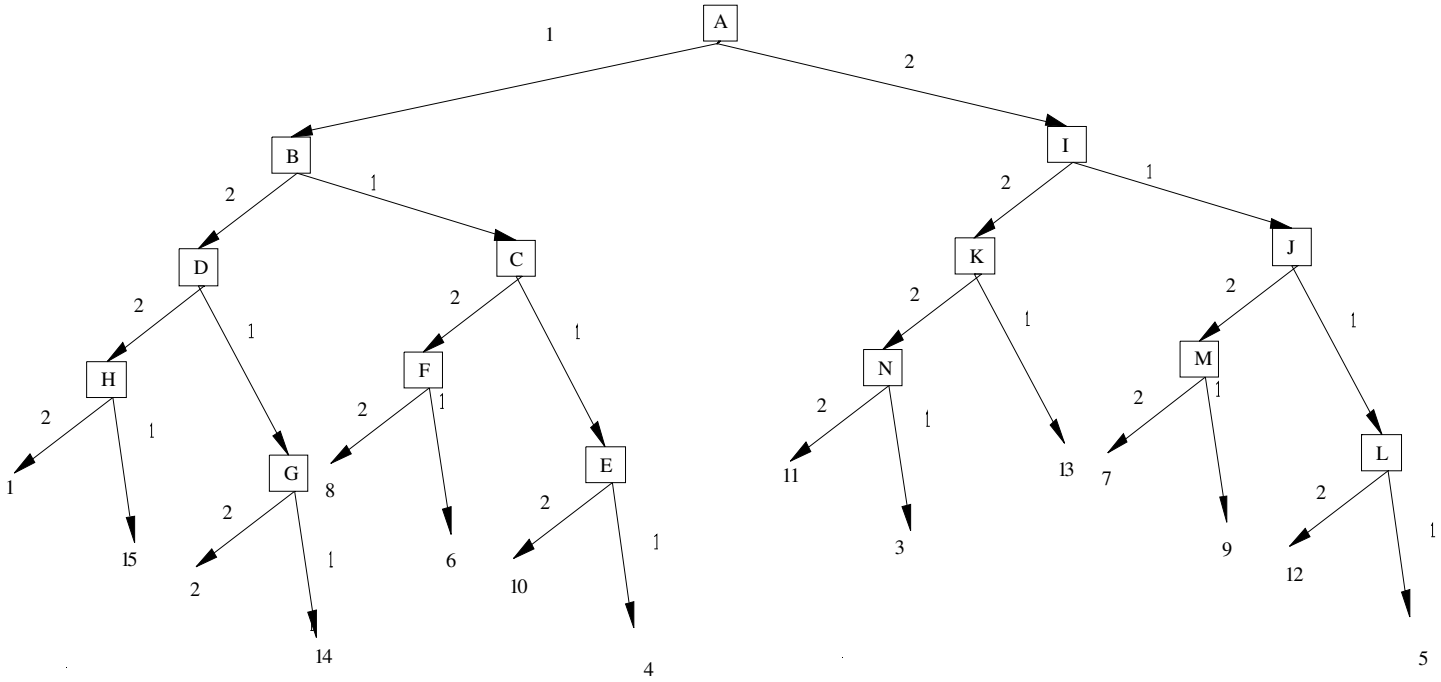


Figure 3.8**:** Recognition of 15 images using 14 criteria and minimum number of stages

A: DCT 332 mid band, B: DCT 333 lowest band, C: DCT 333 lowest band, D: DCT 333 lowest band, E: SVD 101010 highest band, F: SVD 101010 lowest band, G: DCT 333 lowest band, H: DCT 555 mid band, I: DCT 333 lowest band. J: DCT 333 highest band, K: DCT 333 highest band, L: DCT 433 lowest band, M: SVD 101010 lowest band, N: SVD 101010 lowest band.

More than one set of criteria can lead to the same tree structure. Some sets of criteria give the same composite indices to the signals. Others can give different indices. The important thing is that at the final stage each signal will have a unique composite index that distinguishes it from all other signals.

In case of having *N* input signals , $N = 2^n$ **,** where *n* is an integer, let *NC* be the possible number of sets of criteria, for the same tree structure, that yields a unique composite index ($c_1c_2c_{3...}\ c_n$), with minimum number of digits, for each of the N input signals.

$$NC = \prod_{i=0}^{k} \left({}_{w}C_{z} \right)^{2^i} \tag{3.3}$$

$$k = \frac{\log(N)}{\log(2)} - 1 \tag{3.4}$$

$$w = \frac{N}{2^i} \tag{3.5}$$

$$z = \frac{N}{2^{i+1}} \tag{3.6}$$

Where ***C*** is Combination

$${}^{n}p_{c} = \frac{n!}{c!(n-c)!} \tag{3.7}$$

(

Example 3

In this example, 3 possible sets of criteria , Figure 3.9 - 3.11, are given to show that there are more than one set of criteria that can be used for the same tree structure,. The main goal is recognizing images regardless of the set of criteria used and the composite index that identifies each of the unknown images. The important thing is that each image has a unique index which is different from all other images.

First set of criteria

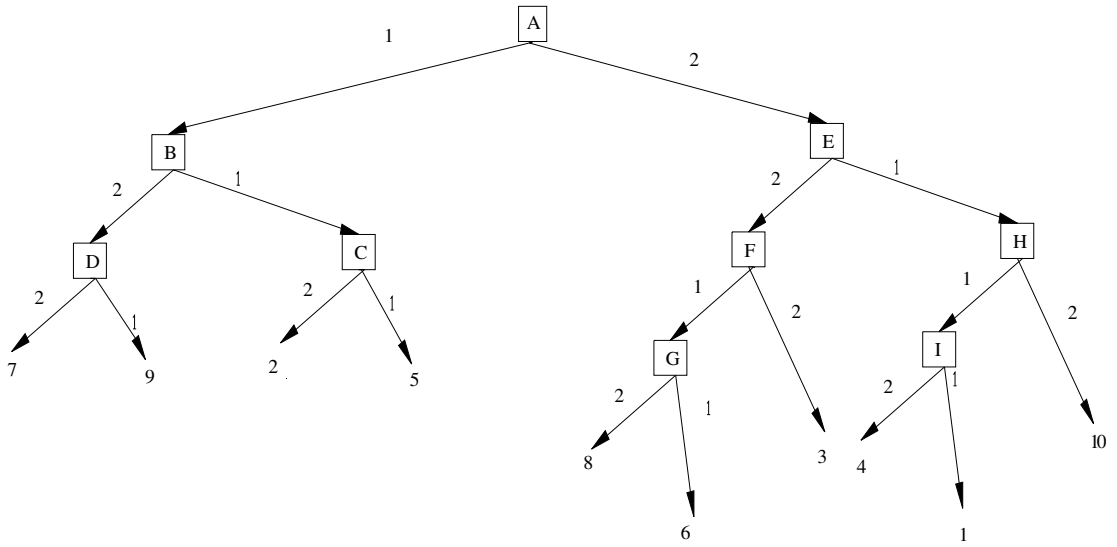


Figure 3.9: First set of criteria that can be used to recognize the 10 images

A: DCT (333, 1), B: DCT (343, 3), C: HAAR (333, 1), D: SVD (1010101, 3), E: DCT (333, 3), F: DCT (333, 2), G: SVD (101010, 3), H: DCT (334, 3), I: SVD (101010, 3).

- Image 7 is identified by *122*
- Image 6 is identified by *2211*

Second set of criteria

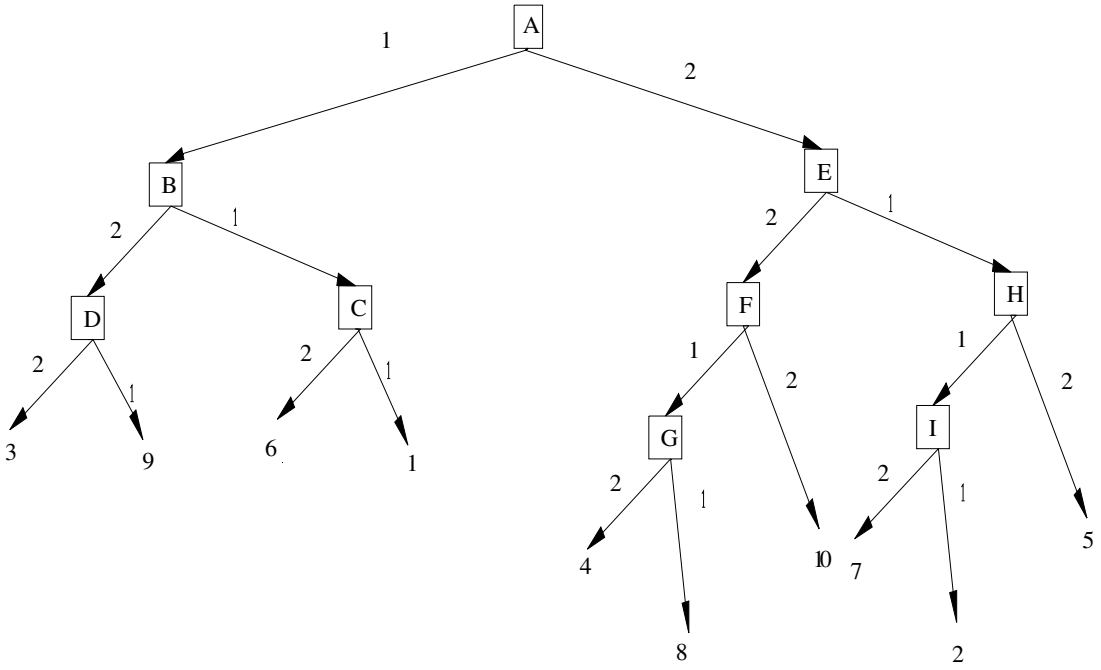


Figure 3.10: Second set of criteria that can be used to recognize the 10 images

A: SVD (555, 2), B: DCT (333, 1), C: DCT (333, 3), D: DCT (333, 1), E: DCT (333, 1), F: HAAR (333, 1), G: HAAR (554, 3), H: DCT (433, 1), I: DCT (333, 2).

- Image *7* is identified by *2112*
- Image *6* is identified by *112*

Third set of criteria

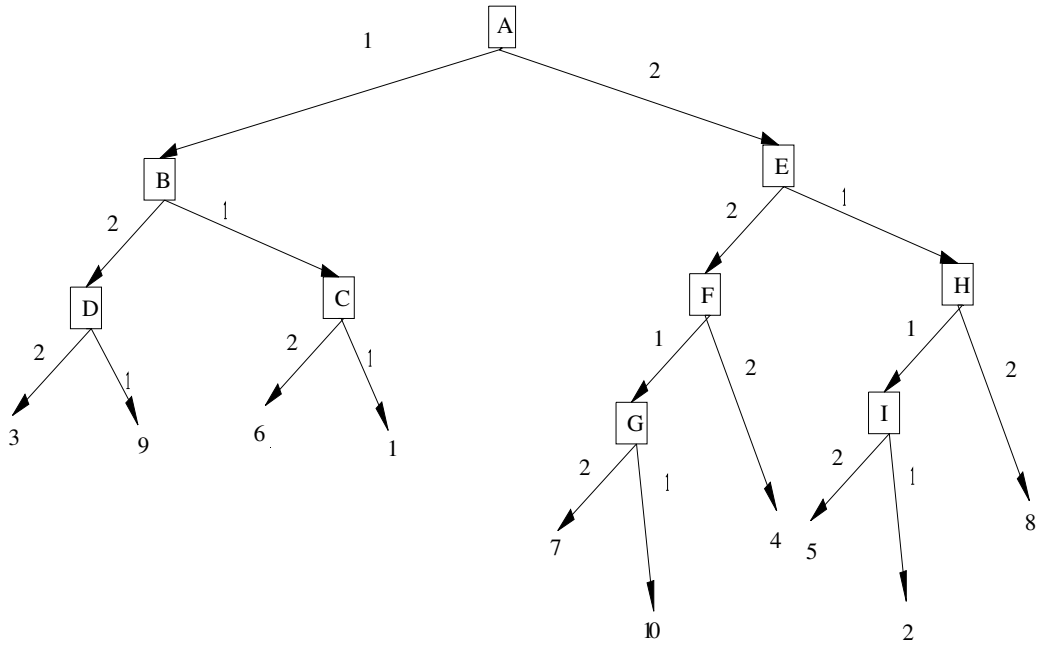


Figure 3.11: Third set of criteria that can be used to recognize the 10 images

A: SVD (555, 2), B: DCT (333, 1), C: DCT (333, 3), D: DCT (333, 1), E: DCT (433, 2), F: HAAR (333, 1), G: HAAR (554, 3), H: HAAR (433, 2), I: DCT (333, 3).

- Image *7* is identified by *2212*
- Image *6* is identified by *112*

It is worthwhile to note that the above two cases (signals of different probability of occurrence and signals of the same probability of occurrence) can be combined together. This means that we could proceed with the second case, grouping signals into 2 groups of almost the same number of signals. Then, at a certain stage, for a specific group, a criterion is employed to distinguish one signal from the other signals belonging to the same group.

3.5. Pattern recognition algorithm

In this section, the algorithm employed by the proposed pattern recognition system, using Matlab software, is summarized into 2 stages: The learning mode and the running mode.

3.5.1. Learning Mode

1) Some input information required by the system is:
   1.1. The number of signals.
   1.2. The size of signals.
   1.3. The number of groups at each stage of classification tree.
   1.4. The classification techniques used at each stage of classification tree.
   1.5. The amount of noise added to the original signal.
   1.6. The number of votes that contribute to the final decision.

2) Projection of signals into different transform domains.

3) Computation of criteria.

4) Grouping signals into the required number of groups based on the developed criteria and according to the selected classification technique.

5) Computation of classification accuracy in case of noisy images.

6) Selecting the optimum set of criteria for a specific tree structure.

7) Searching the available criteria for more criteria giving the same output.

8) Voting such that each criterion contributes to the final decision by a certain weight.

3.5.2. Running Mode

1) When the unknown signal is introduced to the system, it takes the path, through the decision tree from the input node to the terminal node, determined in the training mode the gives the highest classification accuracy.

2) At the terminal node, the unknown signal is identified by a composite index according to the group numbers assigned to it throughout its path in the tree structure.

3) The unknown signal is then recognized by comparing the resulting composite index to the available database from the training mode.

CHAPTER FOUR: EXPERIMENTAL RESULTS

4.1. Introduction

As mentioned in previous chapters, enormous sets of criteria can be developed from the features representing the signals in different domains. The proposed pattern recognition system is capable of testing all available criteria and selecting the optimum set of criteria according to the problem under consideration. It is also capable of employing the classification technique that gives high accuracy in case of noisy signals according to the problem under consideration.

In this chapter, some of these criteria are studied in detail. The effect of using preprocessing techniques on the accuracy of the classification process is also shown. Some experimental results illustrating a comparison between the performance of two classification techniques; Vector Quantization and Neural Networks in the presence of noisy or corrupted data are also given.

4.2. Vector Quantization versus Neural Networks

Both Neural Networks (NN) and Vector Quantization (VQ) are great classification techniques that have shown promising results in the literature.

In this section, the classification results obtained when images are binary clustered using NN are presented. The classification results when using VQ design techniques are also shown. Then, a comparison between the performances of these two classification techniques, in case of noisy or corrupted data, is given.

In the following examples, 32 facial images are downloaded from the Internet.
The 32 facial images are projected into 3 transform domains; HAAR, SVD and DCT. Based on the criteria developed from the transform domains, a NN is trained to group the images into 2 groups of the same statistical properties using backpropagation algorithm. Then, this NN is tested with noisy images (up to 80 Gray level white Gaussian noise added to the original images).

The facial images are also binary clustered using a VQ codebook. This codebook is then tested with the noisy images.

Tables 4.1 - 4.3 show a comparison between the performance of NN and VQ in binary clustering noisy images when using different transform methods.

4.2.1. Result obtained when using Haar Transform

- Column 1 represents the size of blocks (low sequency, mid sequency and high sequency) resulting from projecting the signal into the HAAR domain.
- Column 2 represents which bands (1= low sequency, 2=mid sequency and 3= high sequency) are used to develop the criteria.
- Column 3 represents the number of images binary clustered correctly when using NN.
- Column 4 represents the number of images binary clustered correctly when using VQ codebook.

Table 4.1

Comparison between the performance of NN and VQ when noisy images are binary classified based on the features extracted from HAAR Transform

Block size	**Band**	**No. of correct images using NN**	**No. of correct images using VQ**
334	1	27	29
334	2	29	29
334	3	26	29
335	3	30	31
344	2	28	29
344	3	30	30
345	3	29	28
355	2	19	31
355	3	28	29
434	1	30	29
434	2	28	30
434	3	29	28
435	3	17	29
444	2	30	31
445	3	30	29
454	2	29	26
454	3	30	29
455	3	30	30
534	1	30	29

4.2.2. Results obtained when using the features extracted from Singular Value Decomposition (SVD)

- Column 1 represents the criteria developed from the singular values extracted from the input images.
- Column 2 represents the number of images binary clustered correctly when using NN.
- Column 3 represents the number of images binary clustered correctly when using VQ.

Table 4.2

Comparison between the performance of NN and VQ when noisy images are binary clustered based on the features extracted from SVD.

Criterion used	**No. of correct images from NN**	**No. of correct images from VQ**
Summation of the largest 5 singular values	26	27
S Summation of 6^{th} to 10^{th} singular values	29	31
Summation of 11^{th} to 15^{th} singular values	30	30
Summation of 11^{th} to 20^{th} singular values	31	28
Summation of 6^{th} to 15^{th} singular values	30	31
Summation of 16^{th} to 20^{th} singular values	30	19
Summation of 16^{th} to 25^{th} singular values	27	17
Summation of the largest 10 singular values	26	29
Summation of 11^{th} to 16^{th} singular values	30	30
Summation of 21^{3t} to 25^{th} singular values	17	14
Summation of 21^{st} to 30^{th} singular values	14	12

4.2.3. Results obtained when using the features extracted from Discrete Cosine Transform (DCT)

- Column 1 represents the size of blocks (low frequency, mid frequency and high frequency) as a result of projecting the signal into the DCT domain.
- Column 2 represents which bands (1= low frequency, 2=mid frequency and 3= high frequency) used to develop the criteria.
- Column 3 represents the number of images binary clustered correctly when using NN.
- Column 4 represents the number of images binary clustered correctly when using VQ.

Table 4.3

Comparison between the performance of NN and VQ when noisy images are binary clustered based on the features extracted from DCT

Block size	**Band**	**No. of correct images from NN**	**No. of correct images from VQ**
333	1	31	28
333	2	30	29
333	3	30	29
334	3	29	27
335	3	29	26
343	2	31	26
343	3	30	26
344	3	31	29
345	3	31	31
353	2	31	31
353	3	29	30
354	3	28	28
355	3	31	31
433	1	31	29
433	2	31	30
433	3	29	29
434	3	31	31
435	3	31	31
443	2	30	26
443	3	29	28
444	3	29	29
445	3	30	30
453	2	31	31
453	3	30	28

Block size	Band	No. of correct images from NN	No. of correct images from VQ
454	3	30	30
455	3	30	30
533	1	31	25
533	2	30	30
533	3	29	29
534	3	27	29
535	3	31	31
543	2	29	29
543	3	31	29
544	3	31	30
545	1	31	25
553	2	30	29
553	3	31	31
554	3	30	30
555	3	31	29

Tables 4.1 - 4.3 show a small sample of the classification results obtained when testing the trained NN or the designed VQ codebook with noisy signals. For certain types of criteria, VQ codebook gives better results than NN. For some other criteria, NN gives better results than Vector Quantization. In some other cases, both techniques have the same accuracy.

4.3. Edge Detection

Edge detection is one of the preprocessing techniques that has been introduced in the literature and has shown a great efficiency in classification and pattern recognition. Some of the criteria tested by the proposed pattern recognition system are computed from the edges of the input images.

This section presents the classification accuracy when using edge detection as a preprocessing technique.

4.3.1. Edge Detection and standard deviation

Standard Deviation is one of the criteria employed by the proposed system. This criterion is computed as follows: Edges of the input images are found by means of Canny, Prewitt, Zerocrossing or Roberts edge detection techniques. The images are divided into blocks of different sizes and the mean of each block is computed. The standard deviation of each image is computed and based on this standard deviation; a vector quantization codebook is designed. In this codebook the images are grouped into 2 groups. The following examples illustrate the results obtained when this codebook is tested with noisy images.

4.3.1.1.Experimental results

In this experiment, 32 facial images are clustered into 2 or 3 groups by employing standard deviation and vector quantization.

Table 4.4 presents the classification results when grouping noisy images (up to 80 gray level white Gaussian noise added to the original images) into 2 groups based on the criteria developed from standard deviation and Canny algorithm, as an edge detection technique, for sub-images of different sizes.

Table 4.4

Classification results using Canny algorithm and standard deviation

Sub-image size	2 x 2	4 x 4	8 x 8	16 x16
No. of images clustered correctly	19	23	22	23

Table 4.5 presents the classification results when grouping noisy images (up to 80 gray level white Gaussian noise added to the original images) into 2 groups based on the criteria developed from standard deviation and Prewitt algorithm, as an edge detection technique, for sub-images of different sizes.

Table 4.5

Classification results using Prewitt algorithm and standard deviation

Sub-image size	2 x 2	4 x 4	8 x 8	16 x16
No. of images clustered correctly	26	21	26	23

Table 4.6 shows a comparison between the classification accuracy when grouping noisy images into 2 groups and when grouping noisy images into 3 groups in case of using Prewitt edge detection technique

Table 4.6

Comparison between the results obtained when grouping images into 2 or 3 groups based on a criteria developed from standard deviation and Prewitt algorithm

Block size	**No. of groups**	**No. of images classified correctly**
2 x 2	2	26
2 x 2	3	23
4 x 4	2	21
4 x 4	3	15
8 x 8	2	26
8 x 8	3	24
16 x 16	2	23
16 x 16	3	14

4.3.2. The effect of using edge detection techniques in conjunction with mathematical transforms on the classification accuracy

In this section, simple examples are presented to show the effect of using edge detection techniques on the classification accuracy. The images are projected into different transform domains. Then, the images are grouped into 2 groups based on the features extracted from these domains.

4.3.2.1. Edge detection in conjunction with SVD

The main goal of this experiment is to test the classification accuracy in case of having noisy images when employing vector quantization codebook and edge detection techniques in conjunction with singular value decomposition.

Thirty- two facial images are clustered into 2 groups using vector quantization codebook, based on the criteria developed from the singular values extracted from the edges of images. Then noisy images are then generated from the original images by adding up to 80 Gray level white Gaussian noise.

Table 4.7 presents the classification accuracy of a small sample of criteria used to binary cluster the noisy images, when using different edge detection techniques such as: Canny, Prewitt, Zerocrossing and Robert.

Table 4.7

Classification results using SVD and different edge detection techniques

Preprocessing Technique		Canny	Prewitt	Zero crossing	Roberts	none
SVD	Largest singular value	22	24	27	27	31
	Summation of largest 4 singular values	17	24	27	21	24
	Summation of 7 largest singular values	19	25	26	24	25
	Summation of 10 largest singular values	21	24	26	19	26
	Summation of 11 largest singular values	19	27	28	21	28
	Summation of second 5 largest singular values	20	26	27	21	27
	Summation of second 10 largest singular values	20	25	26	20	19
	Summation of second to the 6^{th} largest singular values	24	23	22	21	20

Form Table 4.7, it is clear that the classification accuracy depends on the criterion used as well as on the preprocessing technique

4.3.2.2.Edge detection in conjunction with DCT

In this example, a vector quantization code book is designed to cluster 32 facial images into 2 groups. The grouping of images is based the criteria developed from the edges extracted from the images after being projected into discrete cosine transform domain.

The vector quantization codebook is then tested with noisy images. Noisy images are generated from the original images by adding up to 80 Gray level white Gaussian noise.

Table 4.8 shows the resulting classification accuracy for a small sample of criteria used to binary cluster the noisy images when different edge detection techniques such as: Canny, Prewitt, Zerocrossing and Robert are used.

Table 4.8

Classification results using DCT and different edge detection techniques

Block size	Band	Canny	Prewitt	Zero crossing	Roberts	None
333	1	30	28	31	29	28
333	2	26	24	30	23	28
333	3	21	24	23	25	29
433	1	26	26	25	25	29
433	2	22	25	23	20	30
433	3	22	22	21	22	29
443	2	18	26	23	18	27
443	3	18	27	20	24	29
454	2	20	26	23	19	28
454	3	26	20	16	19	26
533	1	24	25	25	21	32
533	2	20	26	23	23	28
533	3	18	27	20	24	30
544	2	20	26	19	25	26
544	3	26	20	15	19	22
553	2	22	25	22	21	26
553	3	24	25	22	25	22

Table 4.8 shows that the classification accuracy depends on the criterion used as well as on the preprocessing technique.

4.3.2.2. Mixed transforms: Wavelet and DCT

In this example, 32 facial images are clustered into 2 clusters using mixed transforms. First, the discrete wavelet transform is employed to split the images into sub-bands (level 2). Then, the resulting sub-bands are projected into DCT domain.

A vector quantization code book is designed to cluster 32 facial images into 2 groups based on the criteria developed from the features resulting from the previous step.

Noisy images are generated from the original images by adding up to 80 Gray level white Gaussian noise. Table 4.9 reports the number of the noisy images binary clustered correctly when DCT is applied to the sub-bands resulting from the wavelet transform.

Table 4.9

Classification results using mixed transforms (Wavelet and DCT)

Block size	**Band**	**Approximation**	**Vertical**	**Horizontal**	**Diagonal**	**No wavelet**
333	1	23	25	19	13	28
333	2	25	21	21	16	28
333	3	22	22	24	20	29
433	1	22	25	20	16	29
433	2	22	22	19	23	30
433	3	19	23	23	16	29
443	2	22	21	19	21	27
443	3	15	15	13	22	29
454	2	22	21	16	22	28
454	3	21	25	18	20	26
533	1	23	23	23	16	32
533	2	18	16	25	21	28
533	3	15	20	21	13	30
544	2	20	17	18	17	26
544	3	21	25	22	15	22

4.3.2.4. Edge detection, wavelet and DCT

In this example, 32 facial images are binary clustered using mixed transform and edge detection techniques

First, the discrete wavelet transform is employed to split the images into subbands (level 2). Then, 4 edge detection methods: Canny, Prewitt, zerocrossing and Robert methods are used to extract the edges of the subbands. Finally, the extracted edges are projected into DCT domain.

A vector quantization code book is designed to cluster 32 facial images into 2 groups based on the criteria developed from the features resulting from the previous step.

Noisy images are generated from the original images by adding up to 80 Gray level white Gaussian noise. Table 4.10 reports the number of the noisy images binary clustered correctly when different edge detection methods are applied to the images resulting from the four wavelet filters.

Table 4.10

Classification results using WT, DCT and different edge detection methods

Block size	Band	Wavelet filter	Canny	Prewitt	Zero crossing	Robert	No edges No wavelet
333	1	approximation	25	29	27	20	28
333	1	Vertical	19	26	21	23	28
333	1	Horizontal	20	26	25	19	28
333	1	diagonal	20	22	20	29	28
333	2	approximation	24	20	25	17	28
333	2	Vertical	19	21	21	22	28
333	2	Horizontal	19	19	22	20	28
333	2	diagonal	17	21	23	25	28
333	3	approximation	25	20	25	17	29
333	3	Vertical	21	20	23	23	29
333	3	Horizontal	22	16	21	20	29
333	3	diagonal	19	28	22	27	29
334	3	approximation	22	21	26	18	27
334	3	Vertical	21	22	24	20	27
334	3	Horizontal	19	16	25	16	27
334	3	diagonal	17	24	23	28	27
433	1	approximation	24	24	26	16	29
433	1	Vertical	19	14	25	20	29
433	1	Horizontal	17	23	23	15	29
433	1	diagonal	20	26	22	24	29
433	2	approximation	17	19	19	17	30
433	2	Vertical	18	23	21	21	30
433	2	Horizontal	22	20	22	19	30

Block size	Band	Wavelet filter	Canny	Prewitt	Zero crossing	Robert	No edges No wavelet
433	2	diagonal	17	24	16	24	30
433	3	approximation	20	21	23	20	29
433	3	Vertical	19	22	24	27	29
433	3	Horizontal	17	17	22	14	29
433	3	diagonal	18	26	24	27	29
445	2	approximation	16	18	18	21	27
445	2	Vertical	20	23	24	18	27
445	2	Horizontal	18	21	27	20	27
445	2	diagonal	13	27	20	26	27
445	3	approximation	19	22	22	21	26
445	3	Vertical	17	25	24	21	26
445	3	Horizontal	18	17	24	17	26
445	3	diagonal	20	28	24	28	26
533	1	approximation	21	23	26	22	31
533	1	Vertical	17	23	20	23	31
533	1	Horizontal	20	19	23	18	31
533	1	diagonal	18	29	21	29	31
533	2	approximation	21	18	23	14	26
533	2	Vertical	14	16	24	20	26
533	2	Horizontal	14	19	21	20	26
533	2	diagonal	24	22	24	26	26
533	3	approximation	21	20	20	18	30
533	3	Vertical	15	19	21	21	30
533	3	Horizontal	17	18	23	21	30
533	3	diagonal	18	23	21	27	30

From the above classification results, the following conclusions can be drawn:

1) The execution time in case of dividing images into blocks and computing the standard deviation increases as the size of block increases.

2) In case of noisy images, grouping signals into 2 groups gives better classification accuracy than grouping the signals into 3 groups.

3) The given results show that sometimes the use of preprocessing techniques such as edge detection techniques improves the classifier's performance.

4) It is also worthwhile to note that the classification accuracy depends on the detection technique, as well as the criterion employed in classification. Some images are classified correctly when using a particular criterion and a specific edge detection technique. On the contrary, some other images are not classified correctly when using the same criterion but classified correctly when using other criteria.

5) The proposed system has the advantage of selecting the optimum criterion, edge detection method and classification technique according to the problem under consideration.

4.4. Correlation

In this section, we briefly describe a simple method that can be used for evaluating the accuracy of the classifier output. In this approach, the correlation coefficients between the input noisy images and all original images in the database are computed.

The unknown image is identified when the correlation coefficient between the unknown image and a particular image is higher than the correlation coefficient between the unknown image and all other images.

The main advantage of this technique is the high classification accuracy. However, it has the disadvantage of the full search for the input image among all available data.

4.4.1. Example 1

In this example, the noisy images are generated by adding up to 50 gray level white Gaussian noise to the 16 original facial images.

The correlation coefficients between the unknown image and all other images are computed. Then, the unknown image is identified based on the value of these coefficients.

Table 4.11

The correlation coefficients between noisy images and original images

		Original Image															
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Noisy Image	**1**	***0.88***	0.30	0.49	0.25	0.07	0.08	0.17	0.07	0.10	0.17	0.08	0.26	0.14	0.02	0.34	0.55
	2	0.28	***0.95***	0.18	0.27	0.29	0.31	0.37	0.16	0	0.41	0.17	0.22	0.05	0.30	0.25	0.27
	3	0.47	0.20	***0.92***	0.47	0.01	0.04	0.10	0.35	0.21	0.18	0.07	0.24	0.40	0.08	0.58	0.32
	4	0.24	0.26	0.47	***0.92***	0.09	0.05	0.05	0.25	0.11	0.15	0.16	0.09	0.30	0.08	0.55	0.06
	5	0.06	0.29	0	0.09	***0.95***	0.21	0.17	0.01	0.08	0.12	0.05	0.15	0.05	0.18	0.17	0.09
	6	0.08	0.34	0.04	0.06	0.22	***0.91***	0.36	0.03	0.009	0.16	0.07	0.14	0.01	0.08	0.04	0.17
	7	0.15	0.37	0.10	0.05	0.17	0.35	***0.94***	0.17	0.03	0.32	0.05	0.02	0.06	0.19	0.01	0.36
	8	0.07	0.17	0.35	0.24	-0.02	0.05	0.17	***0.93***	0.29	0.16	0.14	0.14	0.33	0.05	0.18	0.11
	9	0.09	-0.01	0.20	0.09	0.08	-0.01	-0.04	0.28	***0.93***	-0.06	0.29	0.24	0.20	0.09	0.08	0.10
	10	0.14	0.44	0.17	0.15	0.13	0.16	0.32	0.17	-0.05	***0.92***	-0.02	0.08	0.01	0.23	0.08	0.17
	11	0.089	0.18	0.06	0.16	-0.05	0.06	0.03	0.13	0.30	0.01	***0.93***	0.11	0.06	0.16	0.05	0.08
	12	0.26	0.24	0.24	0.09	0.16	0.14	-0.03	0.15	0.23	0.08	0.11	***0.92***	0.21	0.12	0.16	0.16
	13	.139	0.07	0.42	0.33	0.06	0.00	-0.07	0.35	0.20	-0.01	0.06	0.21	***0.89***	-0.05	0.45	0.03
	14	0.02	0.33	0.08	0.09	0.19	0.10	0.21	0.06	0.10	0.23	0.17	0.13	0.03	***0.99***	0.13	0.12
	15	0.34	0.26	0.61	0.57	-0.17	0.04	0.02	0.19	0.08	0.08	0.07	0.17	0.44	0.12	***0.91***	0.10
	16	0.54	0.28	0.34	0.07	-0.18	0.18	0.37	0.12	0.09	0.17	0.08	0.16	0.03	0.13	0.11	***0.90***

Table 4.11 shows that for any image of the 16 images in the given example, the highest correlation coefficient is the one between the noisy image and its corresponding original image.

4.4.2. Example 2

In this example, up to ***60*** gray level white Gaussian noise are added to each of 16 original facial images.

The correlation coefficient is computed between each image and all other images.

For all images, the highest correlation coefficient is the correlation coefficient between the noisy image and the corresponding original image.

4.4.3. Example 3

In this example, up to***70*** gray level white Gaussian noise are added to each of 16 original facial images.

The correlation coefficient is computed between each image and all other images.

For all images, the correlation coefficient between the noisy image and its corresponding original image is the highest correlation coefficient.

4.4.4. Conclusions

1) The execution time of this algorithm is very small.

2) For each image, the correlation coefficient between the noisy image and its original image is greater than the correlation coefficient between the noisy image and any other image in the database.

3) The correlation coefficient between the noisy image and the corresponding original image decreases as the noise increases.

CHAPTER FIVE: IMPLEMENTATION EXAMPLES: THE PROPOSED PATTERN RECOGNITION SYSTEM IN CONJUNCTION WITH NEURAL NETWORKS

5.1. Introduction

As mentioned in Chapter 3, there are 3 classification structures that can be employed by the proposed pattern recognition approach; parallel, cascaded and a combination of both parallel and cascaded structures.

In this chapter, the parallel structure in conjunction with neural Network has been employed by the proposed pattern recognition system to recognize images, Multi-Criteria Multi-Transform Neural Network classifier MCMTNN classifier.

5.2. Proposed Pattern Recognition system: A Parallel Implementation

In this implementation example, Multi-Criteria Multi-Transform Neural Network Classifier MCMTNN [1, 71, 75], the pattern recognizer shown in Figure 5.1, extracts the features in parallel, from more than one transform domain. These features are obtained from the transform coefficients representing the input signals in the different domains. Different classification

criteria in each domain can be developed using the coefficients in that particular domain such as the spectral characteristics, the energy distribution in the different transform domain regions, etc.

A potentially successful criterion *i*, with its selected values of the parameters, in a particular domain, clusters the *N* input signals into a number of distinct non-overlapping clusters. The cluster index, according to the i^{th} criterion, is denoted $c_{i,}$ where c_i = 1, 2, 3… $g_{i,}$, and g_i is the number of groups using the i^{th} criterion. Corresponding to a number *n* of selected criteria, *i* takes the values 1, 2… or n. It is worthwhile to note that more than one criterion can be derived from a given domain. Also, g_i is in general, different for the different *i's.*

The NN Classifier learning continues, by testing all the criteria presented over the parameters range for each criterion, until a successful set of criteria is obtained. A successful classifier using *n* criteria should yield a unique composite index ($c_1c_2c_{3...}$ c_n) corresponding to each of the N input signals.

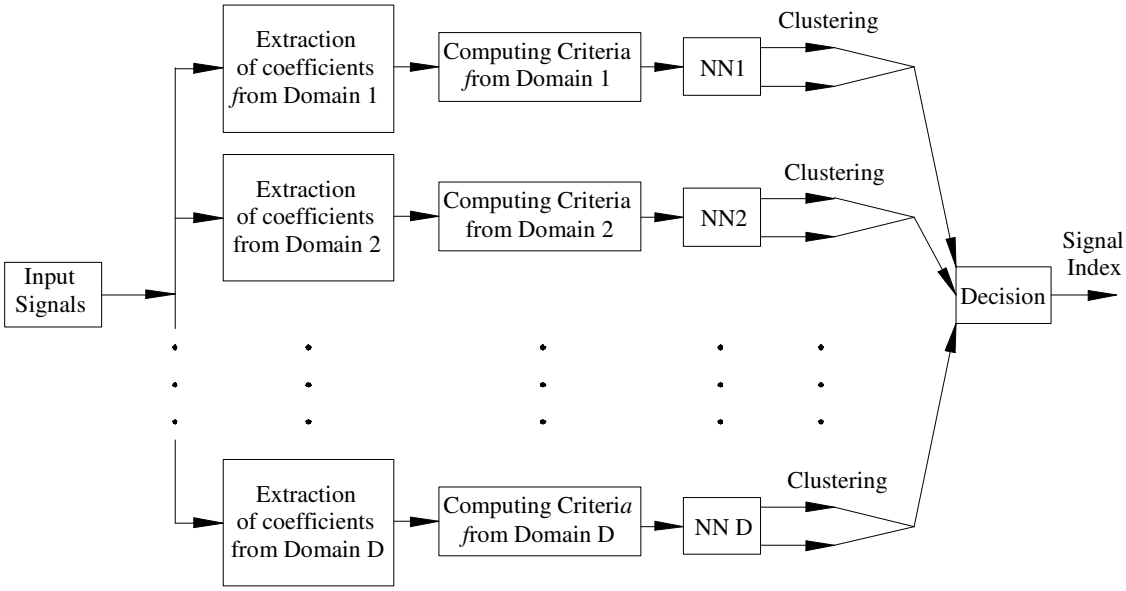


Figure 5.1: A parallel implementation of the proposed MCMTNN classification technique.

It is easy to show that

$$N \leq \prod_{i=1}^{n} g_i \qquad (5.1)$$

$$c_1 = 1, 2\ldots g_1, \quad c_2 = 1, 2\ldots g_2, \ldots \quad c_n = 1, 2\ldots g_n \qquad (5.2)$$

$$n = n_1 + n_2 + n_3 + \ldots + n_D \qquad (5.3)$$

where D is the number of transform domains, and n_k is the number of criteria in the k^{th} domain, (k=1, 2….D).

S_{c_i} = subset of N signals in cluster of index c_i, i =1,2,…n

A Venn diagram is given in Figure 5.2 for clarification for the case $n = 3$.

The hatched area in Figure 5.2 is either empty or contains one particular signal S_p (p=1, 2…or N). S_p occurs only once for all combinations of possible values 1, 2 …g_1, 1, 2 … g_2, and 1, 2 …g_3 of c_1, c_2 and c_3, respectively.

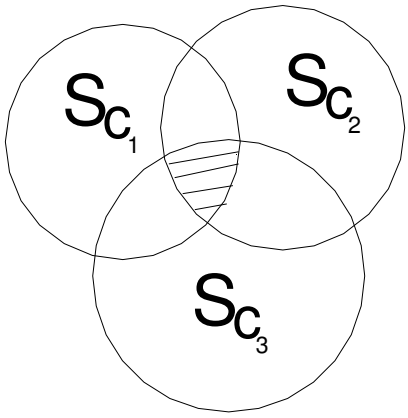


Figure 5.2: Venn Diagram of clusters obtained using different criteria, for a three criteria case. S_{c_i} is the set of signals in cluster of index c_i using the i^{th} criterion.

5.3. Implementation Example

Sample experimental results are given to illustrate the performance of the proposed technique. In this example, thirty-two, 8 bit gray level, facial images are downloaded from the Internet (Olevitti Research Laboratory ORL), Figure 5.3, are introduced to the classifier. The results obtained using the proposed MCMTNN technique, in case of noisy images, are presented and compared to those obtained from a NN employing Single Transform, STNN. It is also compared to the results obtained from Multi-Input Neural Networks.

Figure 5.3: Thirty-two facial images downloaded from the Internet.

5.3.1. MCMTNN Classifier

Here, DCT, HAAR and SVD have been used to demonstrate the technique. The selection of the transforms is not unique and is, intuitively, problem dependent. In addition to computational complexity considerations, such as the existence of a fast computation algorithm for a given transform, it is expected that a transform with a higher energy compaction properties, and consequently feature reduction, is a good candidate to produce successful results.

A resulting successful structure was obtained. It uses three NN's in parallel and projects the images in three domains, namely, DCT, HAAR, and Singular Values, Figure 5.4. Each NN has one neuron in the input layer, 10 neurons in the first hidden layer, 15 neurons in the second hidden layer, and one neuron in the output layer.

Many criteria have been evaluated in case of noisy data. The following criteria are the ones which leads to the best classification accuracy in the given example.

- The first criterion, selected by the first NN, is the sum of the 4 x 4 spatial low frequency components in the DCT domain for each image.
- The sum of the 4 x 4 low sequency HAAR coefficients for each image was chosen by the second NN.
- Criterion 3, from the singular value decomposition (SVD), resulting from the third NN learning, is the sum of the ten largest singular values for each image.

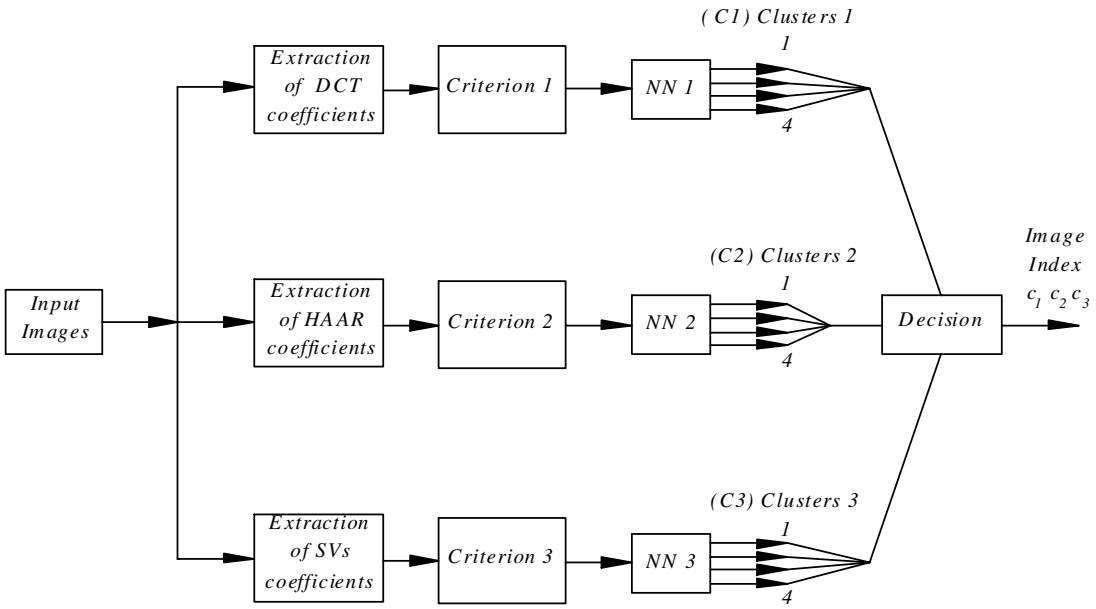
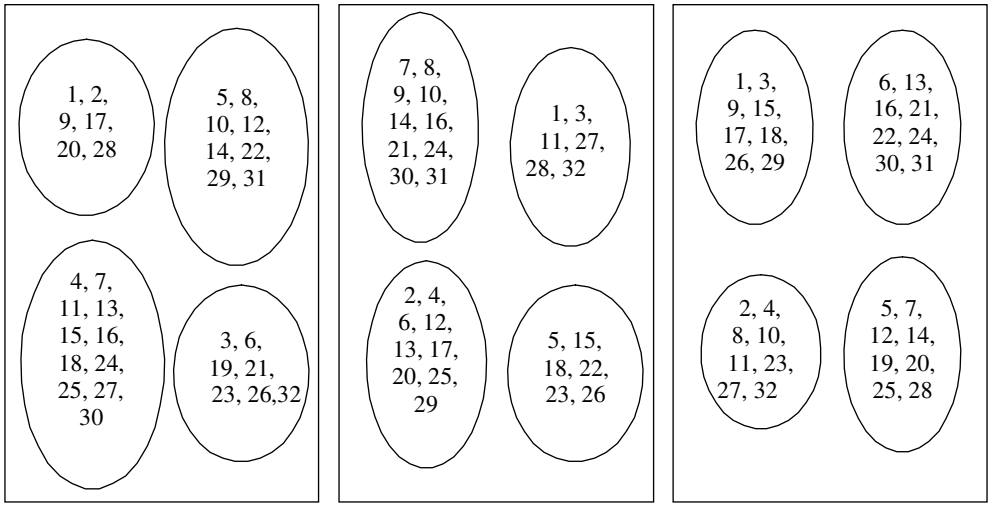


Figure 5.4: An implementation example of the proposed MCMTNN classifier

Each NN clusters the images in four groups, Figure 5.5, i.e, $g_i = 4$ for each NN, and $i = 1$, 2 and 3. The backpropagation algorithm was used for training, yielding a mean square error (MSE) of 10^{-5}.



(a) Four clusters of images 1 to 32 according to criterion 1 in the DCT domain, from NN1.

(b) Four clusters of images 1 to 32 according to criterion 2 in HAAR domain, from NN2.

(c) Four clusters of images 1 to 32 according to criterion 3 using SVD, from NN3.

Figure 5.5: Clusters C1, C2 and C3 of images 1 to 32 obtained from each NN

The MCMTNN classifier, denoted Classifier 1, successfully yields a unique composite index, $c_1c_2c_3$, for each image, Figure 5.5, i.e, the classification accuracy is100%. It is easy to obtain the image number by a simple AND operation of c_1, c_2, and c_3.

In this work, a set of criteria that classified the images successfully was presented. This set is not unique. There is more than one set of criteria that can be used to produce the same composite index for each image. The particular set selected by the system among the successful ones satisfies additional constraints such as computational complexity, noise immunity, etc. In other words, the proposed technique, when presented with a given classification task, yields more than one classifier. In this experiment another classifier, denoted Classifier 2, is obtained which yields 100% classification accuracy. It uses the same structure as classifier 1 except the third NN, where the learning results in a criterion that employs the sum of the second ten largest singular values instead of the first ten largest singular values.

The performance of different classifiers can be evaluated and/or the redundancy can be used to devise a voting scheme to enhance the accuracy of classification of incomplete or corrupted data. Alternatively, a design criterion can be incorporated in the design of the classifiers such that the fused data from the different classifiers yield acceptable performance under nonideal conditions. This is presently pursued. An example is given in the following section.

5.3.2. MCMTNN Classifier for corrupted images

Noisy images are generated from the original images by adding up to 80 gray level white Gaussian noise. The noisy images are presented to Classifier 1 and Classifier 2 described in Sec 5.3.1 and the following results are obtained:

5.3.2.1. Classification of noisy images by Classifier 1

From the thirty two images in Figure 5.3, thirty images are clustered correctly when using the first criterion (DCT), all images are clustered correctly when using the second criterion (HAAR) and 26 images are clustered correctly when using the third criterion (SVD). By simple AND operation of the outputs of the 3 NNs, 26 images out of the thirty two are recognized successfully when using this classifier.

5.3.2.2. Classification of noisy images by Classifier 2

From the input thirty two images in Figure 5.3, thirty images are clustered correctly when using the first criterion (DCT), all images are clustered correctly when using the second criterion (HAAR) and 24 images are clustered correctly when using the third criterion (SVD). By combining the outputs of the 3 NNs, 24 images out of the 32 are recognized successfully when classifier 2 is used.

It is worthwhile to note that classifier 2 is designed such that the images that Classifier 1 fails to recognize are recognized successfully by Classifier 2, and vice versa, when an additional deciding appropriate criterion is introduced. This is illustrated in the following Section.

5.3.2.3. Classification enhancement by combining Classifier 1 and Classifier 2

By using a simple detector and another feature of the image, energy in this example, we determine which of the two results, from classifiers 1 and 2, when they disagree, is correct. This additional voting step resulted in recognizing all of the thirty two images successfully.

5.3.3. STNN Classifier

This structure uses one NN and one transform, Figure 5.6.

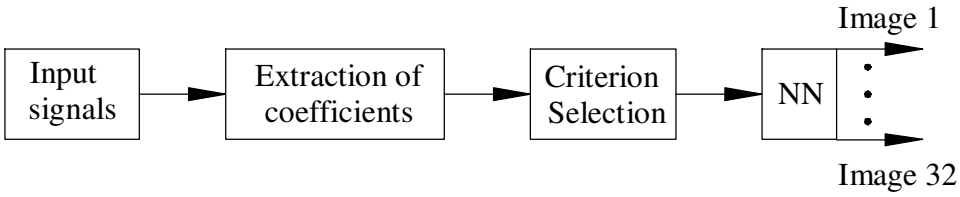


Figure 5.6: Recognition of images using the STNN classifier

The STNN classifier is trained with different criteria in different domains.

The neural network that successfully classifies the images in Figure 5.3 has one neuron in the input layer, 20 neurons in the first hidden layer, 30 neurons in the second hidden layer, 40 neurons in the third hidden layer, 50 neurons in the fourth hidden layer and one neuron in the output layer. A Backpropagation algorithm is used for training with MSE of 10^{-5}.

It is worthwhile to note that the number of outputs of the STNN classifier is much more than that of the proposed approach. The time taken in training the STNN classifier is expected, and confirmed experimentally, to be more than that of the time taken to train the MCMTNN. It is approximately five times in this example, since the NN used for the STNN classifier contains 140 neurons in the hidden layers, while the NNs used for the MCMTNN only contain 25. Moreover, grouping images into 4 groups would obviously take less time for training than the time taken to group images into 32 groups.

The classifiers are tested with images corrupted with noise. The criterion in a particular domain that gives the best results is retained. After many trials, the best results obtained using one of the 3 transforms, DCT, HAAR and SVD, are as follows:

1) In the DCT domain, six images out of the thirty-two are recognized successfully and the criterion selected by the STNN is the sum of the 4 x 4 low frequency components.

2) In the HAAR domain, eight images out of the thirty-two are recognized successfully and the criterion selected by the STNN is the sum of the 4 x 4 spatial low sequency components, for each image.

3) Using SVD, eight images out of the thirty-two are recognized successfully when using the sum of the 4 largest Singular values for each image.

5.3.4. Multi-Input Neural Network

In this Section, a multi-input neural network has been trained to recognize thirty two facial images in Figure 5.3. The structure used has the following specifications:
3 neurons in the input layer, 7 neurons in the first hidden layer, 15 neurons in the second hidden layer, 20 neurons in the third hidden layer, and one neuron in the output layer. Backporpagation algorithm has been used in training.

Different combinations of criteria have been introduced to the multi-input neural network. The combination that gives the best results is: The sum of the 4 x 4 spatial low frequency components in the DCT domain for each image, the sum of the 4 x 4 low sequency HAAR coefficients for each image and the sum of the ten largest singular values for each image.

Different mean square errors have been tried.

The following results are obtained:

1) 10^{-1} MSE, 8 images out of 32 images are recognized correctly.

2) 10^{-3} MSE, 8 images out of 32 images are recognized correctly.

3) 10^{-5} MSE, only 4 images out of 32 are recognized correctly.

CHAPTER SIX: PROPOSED PATTERN RECOGNITION SYSTEM USING DECISION TREE

6.1. Introduction

Classification decision tree algorithms have been used recently in pattern recognition problems. In this chapter, we are proposing a self-designing system, employing the classification tree algorithms [74], capable of recognizing a large number of signals. The classifiers are connected in cascaded. A large number of classification techniques can be used to binary classify signals presented at the nodes of the classification decision tree. Neural Networks and Vector Qunatization are two classification techniques, used in the presented implementation examples, that have shown high classification accuracy in the literature. At the terminal nodes, each signal is identified by a composite index according to its path through the decision tree. The results obtained show the feasibility of the proposed system in case of noisy images.

6.2. Proposed Pattern Recognition system: A Cascaded Implementation

A combination of the original as well as the preprocessed signals is projected into different transform domains. Enormous sets of criteria, characterizing the signals, can be developed from the signal representations in these domains. The availability of a large number of criteria enables the system to recognize different types of signals.

At each node of the classification tree, an appropriately selected criterion, of less complexity and more immunity against noise, is optimized and employed by the optimum classification technique to divide the signals, presented at that node in that stage, into two approximately equal groups. At the terminal node of the tree, each signal is represented by a unique composite, binary word index, corresponding to the signal path through the tree.

Two extreme situations are given in the following sections. In Figure 6.1 the pattern with the highest probability of occurrence is identified in one stage, while the pattern with the lowest probability is identified after N-1 stages. For the structure in Figure 6.2, each pattern is identified after r stages, where $N = 2^r$.

The optimum classification structure is selected according to the nature of the problem under consideration.

6.3. Classification Structures

6.3.1. Recognition of one Signal at Each Stage

The first design of the binary classification unit (BCU-1) [2,73], is used when apriori information regarding the probability of occurrence of each signal, with one of them being dominant, in the group of signals presented to the BCU, is available.

In this structure, there is only one node and one BCU at each stage, Figure 6.1.

One of the signals, s_{ij}, in the group of signals presented to BCU-1 at the j^{th} stage, say $(BCU\text{-}1)_j$, is dominant, i.e, has the highest probability in this group. $(BCU\text{-}1)_j$ employs criterion C_j to group the input set of signals, S_j, to $(BCU\text{-}1)_j$ in two groups. One group g_{wj} contains the dominant signal only, s_{ij}, where s_{ij} is the signal uniquely identified by $(BCU\text{-}1)_j$ at the j^{th} stage while the second group, $g_{w(j+1)}$, contains the remaining signals in the input group S_j. thus

$$g_{wj} = s_{ij} \qquad (6.1)$$

$$g_{w(j+1)} = S_j - s_{ij} = S_{j+1} \qquad (6.2)$$

where the subscript w_j and w_{j+1} are binary words equal to j and j+1, respectively.

Referring to the multistage structure in Figure 6.1, the set S of N signals, $s_1, s_2, s_3 \ldots s_N$, is to be classified where $S = (s_1, s_2, s_3 \ldots s_N)$.

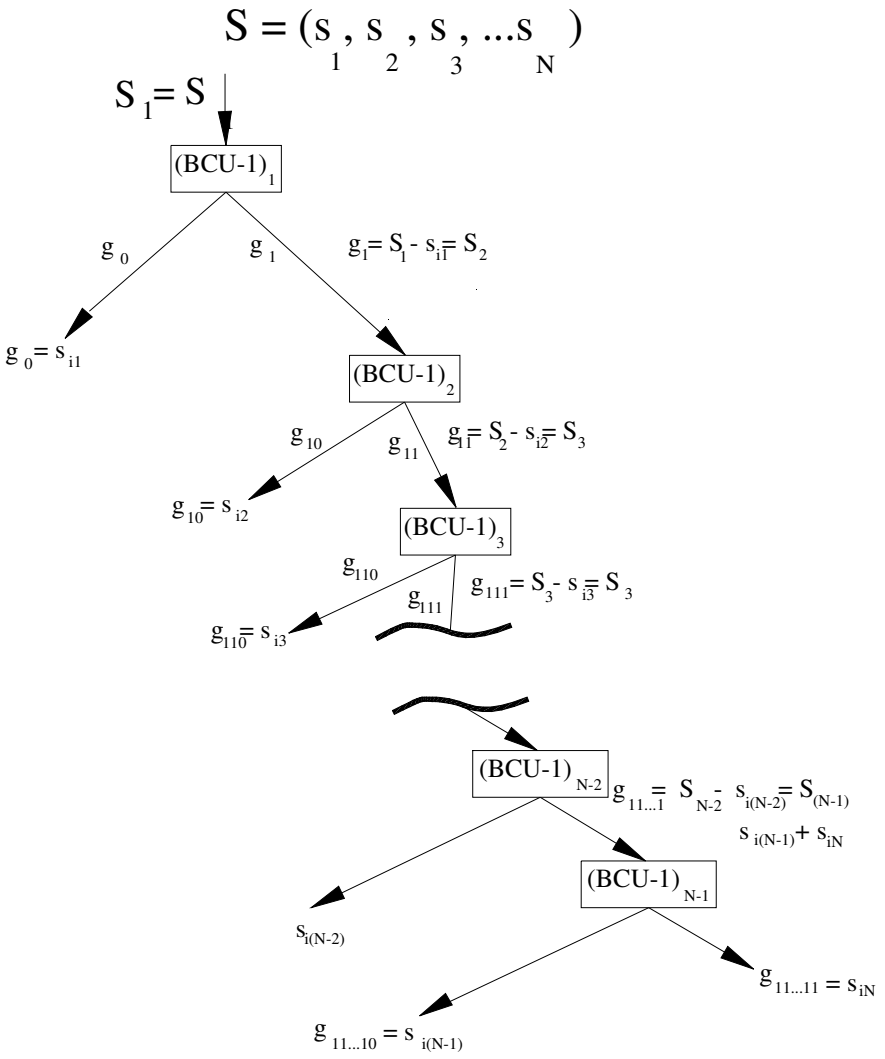


Figure 6.1: A Tree Structure for Recognition of N Signals, one signal identified at each stage

The probability of occurrence associated with the patterns are

$$P_{s_{i1}}, P_{s_{i2}} \ldots P_{s_{iN}} \quad \text{such that} \quad P_{s_{i1}} > P_{s_{i2}} > \ldots > P_{s_{iN}} \tag{6.3}$$

The N signals are extracted in a descending order of their probability. Thus, a signal with higher probability is classified with fewer classification stages compared with a signal with lower probability. This, in general, is expected to lead to an overall reduction in the average time and amount of computations required for recognition.

6.3.2. Classification of Signals of Equal Probability of Occurrence

The second design, BCU-2, is used when either no probability information is available or all the signals in the group presented to the BCU are almost equally probable. The BCU used at each node, BCU-2, groups the input patterns in two groups, each containing the same number of patterns. The corresponding structure is shown in Figure 6.2.

- At the j^{th} stage, there are q nodes

$$q = 2^{j-1} \tag{6.4}$$

- The number of stages required to uniquely identify each of the $N = 2^r$ input patterns is N_s.

$$N_s = r \tag{6.5}$$

- The total number of (BCU-2)'s equals is N_{bcu}

$$N_{bcu} = N-1 \qquad (6.6)$$

- At the k^{th} node, (k=1, 2 …q), in the j^{th} stage, $(BCU\text{-}2)_{jk}$ employs criterion C_{jk} to group its input set of patterns, S_{jk}, into two groups, g_{wo}, and g_{w1}, where w_o and w_1 are j bit binary words equal to 2k-2 and 2k-1, respectively.

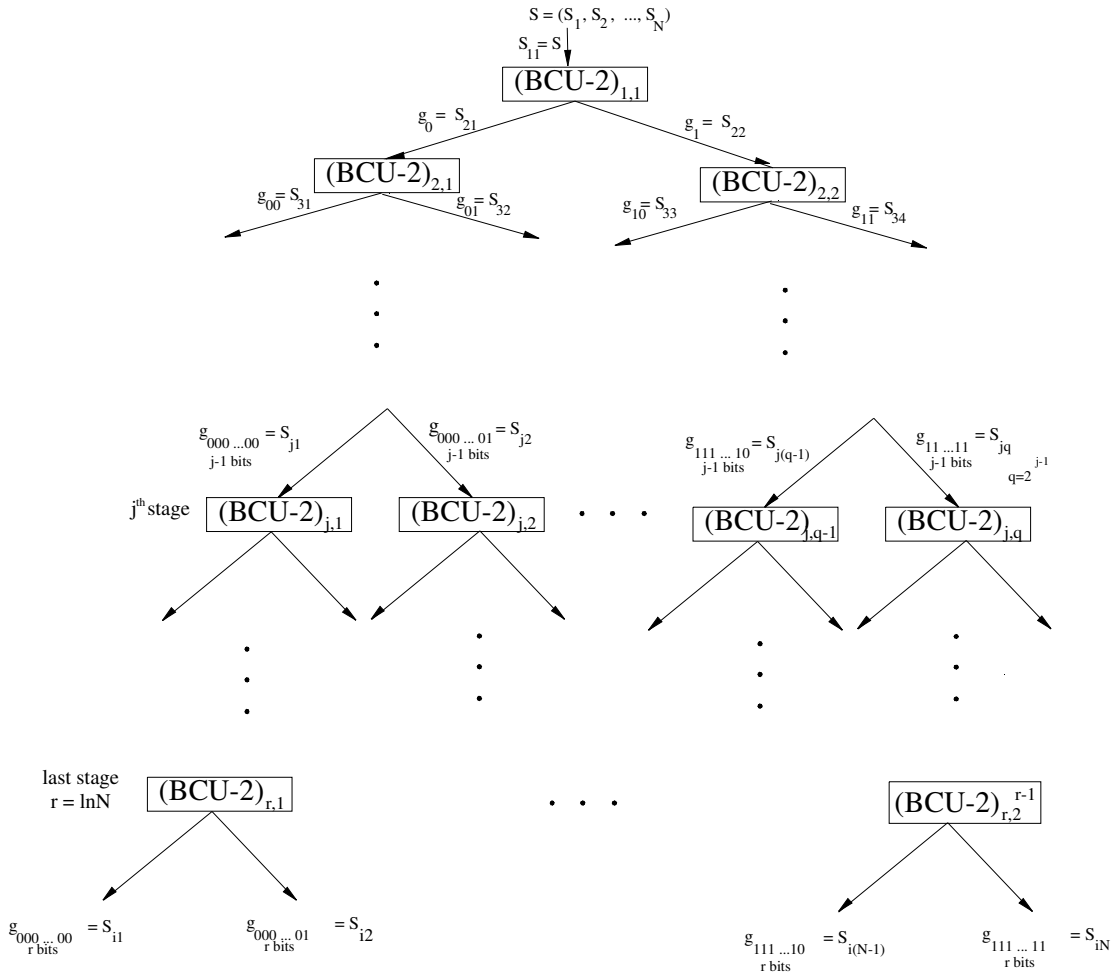


Figure 6.2: A tree structure for recognition of N signals of unknown probability of occurrence

The system operates in two modes, namely, the learning (training) mode, and the running mode.

6.4. Modes of Operation

6.4.1. Learning Mode

The algorithm, in the learning mode, Figure 6.3, is summarized as follows:

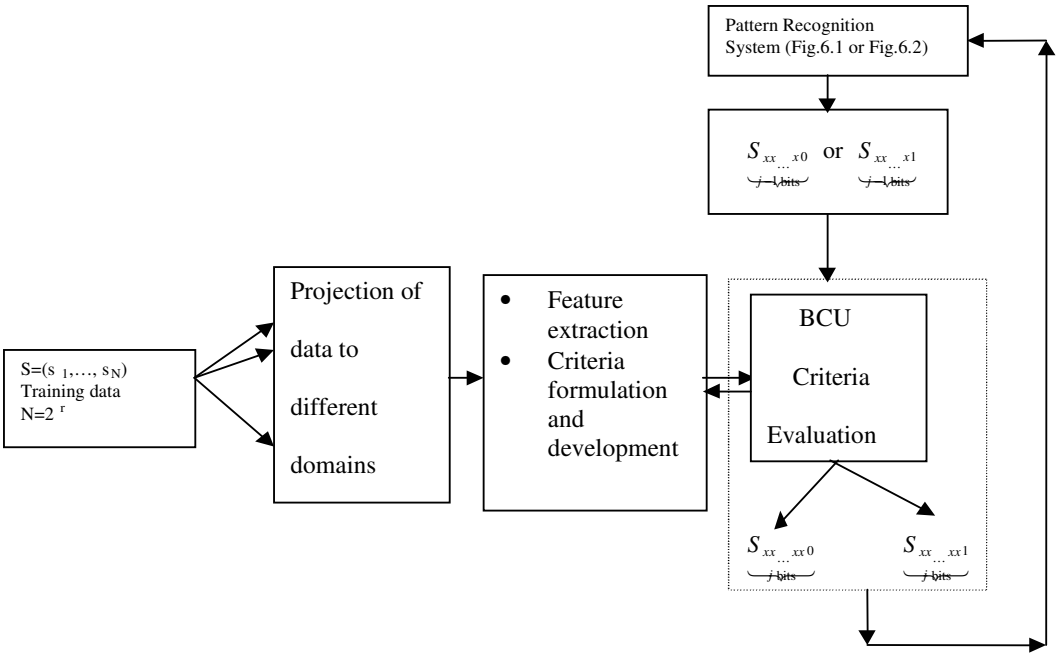


Figure 6.3 The Proposed Pattern Recognition System in the Learning Mode

The original and preprocessed signals are projected and analyzed in different transform domains. Distinguishing characteristics are extracted in each of these domains. Corresponding to each feature, a criterion is developed that computes a quantitative measure of the feature parameters.

The system tests all available criteria. Starting from j=1, Figure 6.1or Figure 6.2, and progressing towards the output, different criteria are evaluated.

The quantitative measure using C_{jk} divides the input signals set S_{jk} to Binary classification unit $(BCU)_{jk}$ into two subgroups, $g_{xxx...x0}$ and $g_{xxx...x1}$. A signal in the input group S_{jk} belongs to subgroup $g_{xxx...x0}$ if the signal falls in a certain range, say r_o, and belongs to subgroup $g_{xxx...x1}$ if it is in the range r_1. The distance from r_o to r_1 is a guard range corresponding to unsuccessful classification.

It is worthwhile to note that to enhance the system's reliability particularly under nonideal conditions such as noisy or corrupt data, several (BCU)'s each operating with its appropriately selected criterion, are used in a voting scheme to replace a critical $(BCU)_{jk}$. The prespecified optimality constraints for adapting/ selecting a criterion C_{jk} include computational complexity, distance between the subgroups (guard range), classification accuracy with noisy / distorted data, etc.

The steps above are followed until each member of the *N* input patterns in case of recognition (*N* district groups in case of classification) is uniquely classified, by the *k* bits binary word subscript of $g_{xxx...x}$ at the output stage depending on the number of stages used.

For unique classification of *N* patterns, the number of Binary Classification Units (BCU's) needed equals *N-1*. It is assumed that no apriori information is available regarding the distribution signals to be classified. As a result, no particular order in the grouping or extraction of a certain member is emphasized.

In the training mode, in the given examples, different classification methods are attempted to binary cluster the signals at each node of the classification decision tree. The results obtained show that vector quantization techniques give higher classification accuracy, in the presence of

white Gaussian additive noise, than the other classification methods presented in literature. Vector quantization has been gainfully exploited in this system so that the full search of the whole codebook is avoided.

6.4.2. Running Mode

The steps in the running mode, Figure 6.4, can be described as follows:

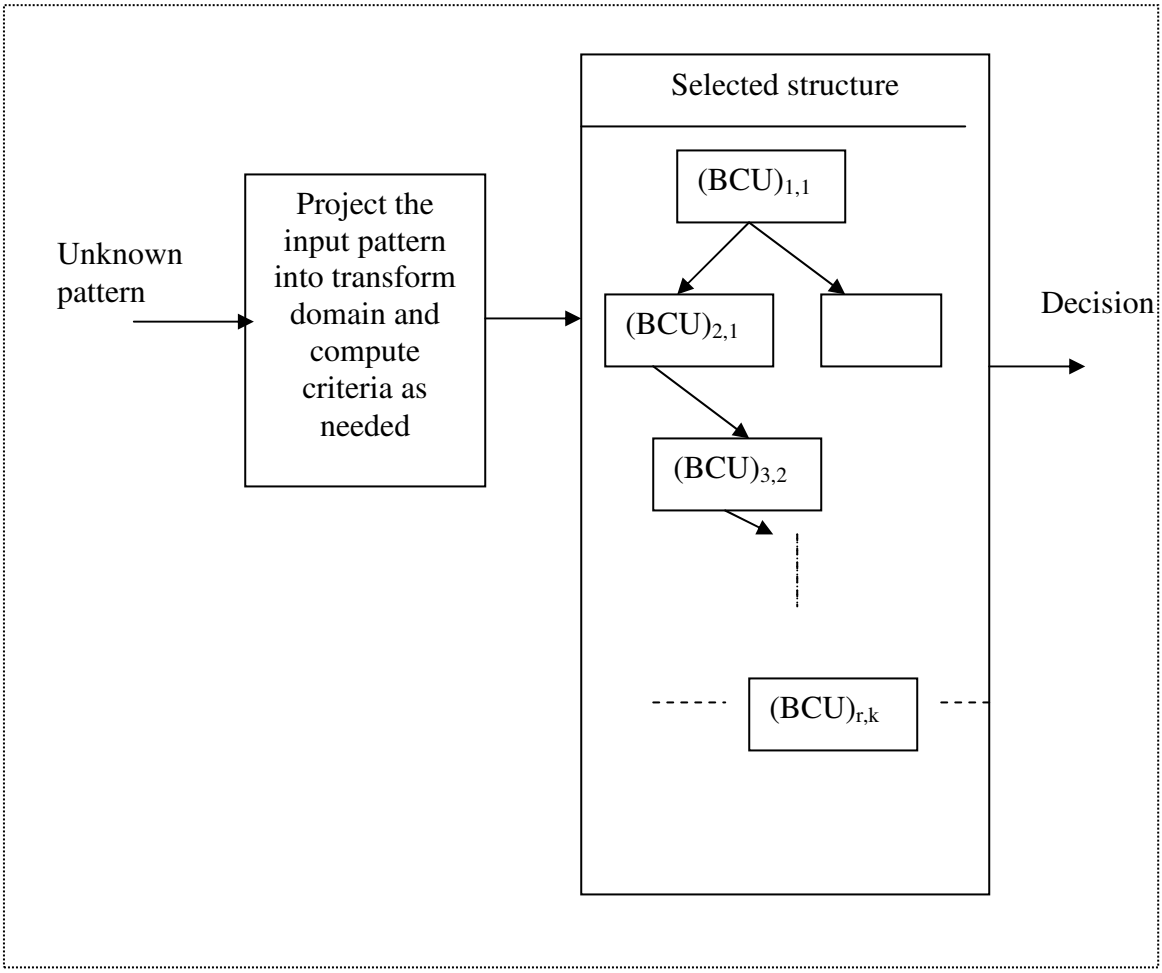


Figure 6.4: The Proposed Pattern Recognition System in the Running Mode

The unknown pattern P_u is projected into the appropriately selected domains in the training mode.

According to the path P_u takes in Figure 6.1 or Figure 6.2, say it reaches the input of $(BCU)_{jk}$, the appropriate criterion C_{jk} is computed, and the correct group(path) $g_{xxx\ldots xo}$ or $g_{xxx\ldots x1}$ is identified.

The process is repeated for j=1, 2…k where the unknown pattern is recognized by the *k* bits binary word subscript of $g_{xx\ldots x}$.

6.5. Implementation Examples

6.5.1. Eight Biomedical images Example

In this example, eight, 8 bit gray level, biomedical images are downloaded from the Internet (Dept. of Dermatology-University of Iowa, college of medicine), Figure 6.5 Noisy images were generated by adding up to 80 gray level white Gaussian additive noise to the 8 original images, Figure 6.6. The system was trained to recognize the noise free images. Then, the system was redesigned to recognize the corrupted images too.

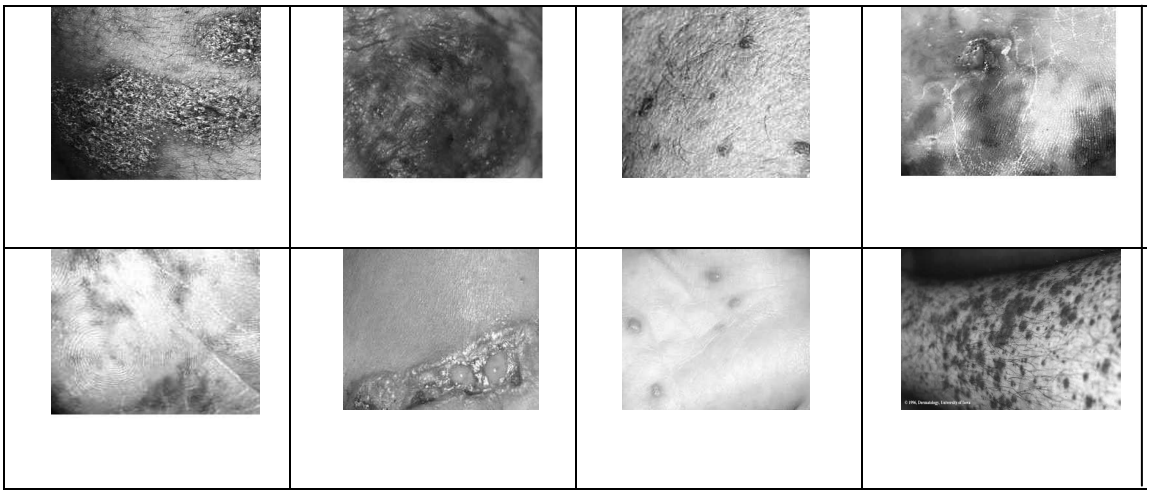


Figure 6.5: Eight biomedical images downloaded from the Internet

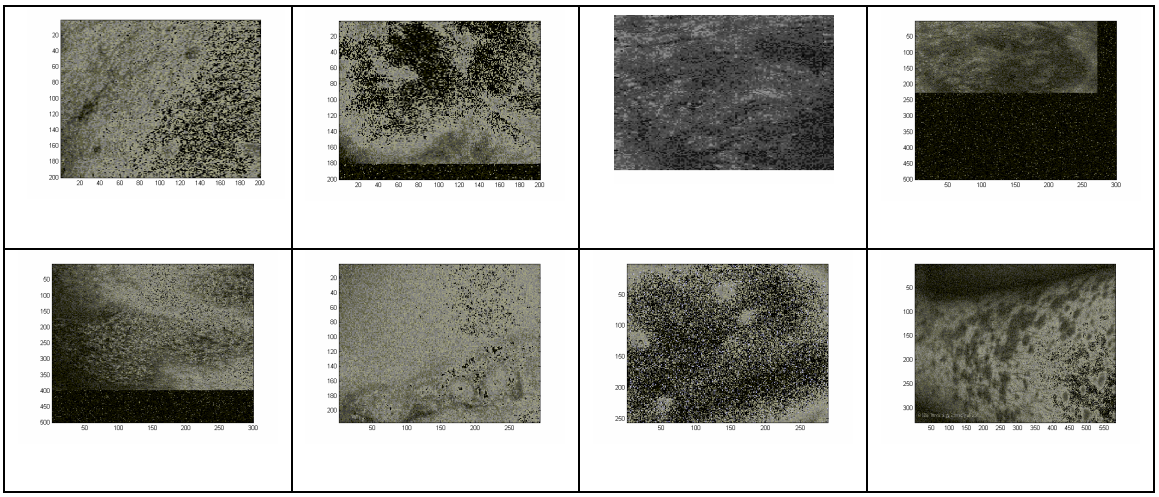


Figure 6.6: Eight noisy biomedical images

6.5.1.1.Results using VQ in conjunction with DT

The proposed system employed the classification structure shown in, Figure 6.7.

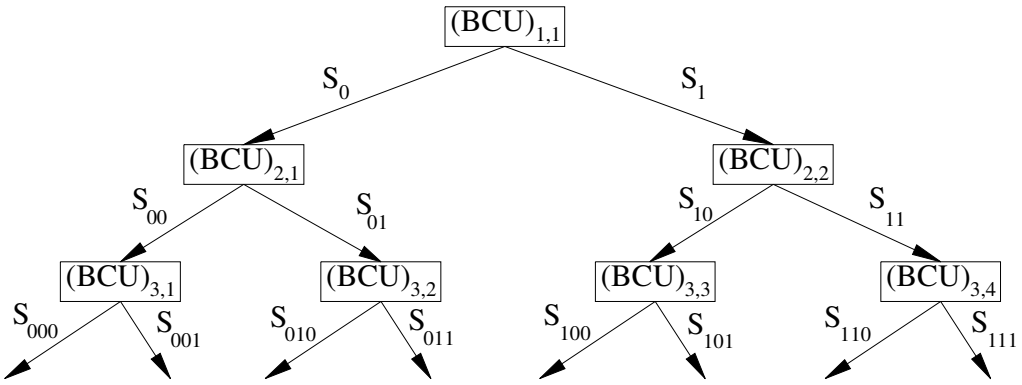


Figure 6.7: Recognition of 8 images using the proposed system

The performance of the proposed pattern recognition system in this example can be described as follows:

At the first stage, the input images are grouped into 2 groups; S_0 and S_1. Then, in the next stage, each group is divided into 2 more groups. The process continues until finally each of the input images is uniquely identified using 3 criteria.

There is a unique one to one correspondence between the identified patterns S_{00} to S_{111} and the input patterns s_1 to s_8 in the input set S.

The following criteria have been employed by the BCU's used in this example:

1) Summation of the largest 10 singular values representing each input image.
2) Summation of the 3 x 3 spatial low frequency components in the DCT domain for each image.

3) Summation of the 3 x 3 low sequency HAAR coefficients.

4) The ratio between the maximum gray level and the mean value of each image.

The system was able to recognize all 8 noisy images correctly using 7 criteria.

6.5.1.2.Results using Single Transform Neural Network Classifier

This structure uses one NN and one transform, Figure 6.8. The STNN classifier is trained with different criteria developed from different transform domains.

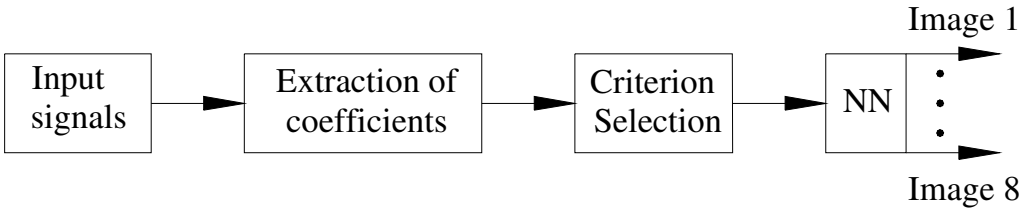


Figure 6.8: Recognition of 8 images using STNN Classifier

The neural network, which successfully recognizes the images, has one neuron in the input layer, 10 neurons in the first hidden layer, 15 neurons in the second hidden layer and one neuron in the output layer.

A Backpropagation algorithm with different MSE (10^{-1}, 10^{-3}, 10^{-5}) is used in training the neural network.

The classifier is tested with images corrupted with up to 80 gray level additive noise. The best results are obtained when the sum of the 3 x 3 spatial low frequency components in the DCT

domain for each image is introduced to the input of the neural network. Two images out of 8 images are recognized correctly.

6.5.1.3. Results using Multi-Input Neural Network classifier

A Multi-Input Neural Network has been trained to recognize the 8 biomedical images. Different combinations of 3 criteria have been used in training the Multi-Input Neural Network with different mean square error (MSE).

When testing the classifier with noisy images, 3 images out of 8 images are recognized correctly.

6.5.2. Eight Facial images Example

In this example, eight, 8 bit gray level, facial images, Figure 6.9, are downloaded from the Internet (Olevitti Research Laboratory ORL). Noisy images were generated by adding up to 80 gray level white Gaussian additive noise to the 8 images.

Figure 6.9: Eight facial images downloaded from the Internet

6.5.2.1.Results using VQ in conjunction with DT

The proposed technique is employed resulting in a structure similar to the one in the previous example, Figure 6.7. The system employs the optimum set of criteria that are capable of grouping the noisy images at each stage of the pattern recognition process with the highest accuracy.

All 8 images are recognized successfully.

6.5.2.2.Results Using Single Transform Neural Network Classifier

The best obtainable results yield three images out of the eight images that are recognized correctly.

6.5.2.3. Results using Multi-input Neural Network Classifier

The best obtainable results yield four images out of the eight images to be recognized correctly.

By comparing the results, including these two sample examples, when the proposed approach, the STNN, and the Multi-Input Neural Network are used, the proposed approach gives superior results in terms of classification accuracy and computational complexity. This can be shown in Table 6.1.

Table 6.1

Comparison between the classification accuracy of the Proposed Technique, STNN and the Multi-Input Neural Network for the recognition of the signals in the previous two examples.

Approach	**Proposed Classifier**	**STNN**	**Multi-Input Neural Network**
Classification Accuracy of Example 1	100%	25%	37.5%
Classification Accuracy of Example 2	100%	37.5%	50%

6.5.3. Thirty-two Facial images Example

In this example, 32 eight bit gray level, facial images are downloaded from the Internet (Olevitti Research Laboratory ORL), Figure 6.10.

Up to 80 gray level white Gaussian additive noise was introduced to the 32 original images, Figure 6.11.

It is to be noted that the classifier in this example was originally trained with noise free images. When the images were corrupted with noise, the performance was degraded. The classifier performance was optimized by the proper selection of criteria and structure to restore the high accuracy.

In this section, the results obtained using the proposed technique are given and compared to the results obtained from using single stage neural network. It is also compared to the results obtained when using multi-input neural network.

Figure 6.10: Thirty-two facial images downloaded from the Internet

6.5.3.1.Results obtained using the proposed pattern recognition technique in conjunction with VQ

.After testing all available criteria, the optimum set of criteria that is less susceptible to noise in this example is as follows:

1) DCT transform: sum of the 3 x 3 low frequency components, sum of the 4 x 4 low frequency components, sum of the 3 x 3 high frequency components, the maximum DCT coefficient, the ratio between the maximum and minimum coefficients, etc.

2) Haar transform: summation of 5x5 low sequency HAAR coefficients, summation of 3x3 mid sequency HAAR coefficients, summation of 4x4 low sequency HAAR coefficients, etc.

3) Singular value decomposition: summation of largest 10 values, summation of the largest 5 singular values, the largest singular value extracted from the binary image representing the edges of the image to be recognized, etc.

4) Energy of signal.

The signals presented at each node, in each stage, of the classification decision tree were binary clustered using vector quantization techniques. When the system was tested with noisy images, all images were recognized correctly.

The classifier design in this example was performed for a particular noise and signal degradation range. When the input images in the running mode have lower Signal to Noise ratio, in comparison with those used in the training mode, the 100% accuracy will not be maintained, and the classifier has to be redesigned. It is expected as the Signal to Noise ratio drops below some value, the complexity of the classifier will increase and the classification accuracy will decrease.

6.5.3.2. Results obtained using the proposed pattern recognition technique in conjunction with NN

After testing all available criteria, the optimum set of criteria that is less susceptible to noise in this example is as follows:

1) DCT transform: sum of the 3 x 3 low frequency components, sum of the 4 x 4 low frequency components, sum of the 3 x 3 high frequency components, the maximum DCT coefficient, the ratio between the maximum and minimum coefficients, etc.

2) Haar transform: summation of 5x5 low sequency HAAR coefficients, summation of 3x3 mid sequency HAAR coefficients, summation of 4x4 low sequency HAAR coefficients, etc.

3) Singular value decomposition: summation of largest 10 values, summation of the largest 5 singular values, the largest singular value extracted from the binary image representing the edges of the image to be recognized, etc.

The signals at each node, in each stage, were split into 2 groups such that the statistical properties are the same in the two groups [54]. Then, neural network was trained to split images, at each node of the decision tree, into 2 groups of almost the same number of images. The structure of the neural network used in this example has the following specifications: 7 neurons in the first hidden layer, 10 neurons in the second hidden layer and one neuron in the output layer. Back propagation algorithm, with 10^{-5} MSE, was used in training the neural network.

At the running mode, when the noisy images were introduced to the neural network, the recognition accuracy was 96.8%.

6.5.3.3.Results using Single Stage Neural Network

The neural network was trained to recognize any of the 32 images using only one stage. The single stage neural network, STNN, employed different criteria developed from different domains.

The neural network that was successfully trained to recognize images in this example has one neuron in the input layer, 20 neurons in the first hidden layer, 30 neurons in the second hidden layer, 40 neurons in the third hidden layer, 50 neurons in the fourth hidden layer and one neuron in the output layer. A Backpropagation algorithm was used for training with MSE of 10^{-5}.

The neural network was tested with images corrupted with noise. The criterion in a particular domain that gives the best results was retained. After many trials, the best results obtained were as follows:

1) In the DCT domain, six images out of the thirty-two were recognized successfully and the criterion selected by the STNN is the sum of the 4 x 4 spatial low frequency components for each image.
2) In the HAAR domain, eight images out of the thirty-two are recognized successfully and the criterion selected by the STNN is the sum of the 4 x 4 low sequency HAAR coefficients for each image.

3) Using SVD, eight images out of the thirty-two are recognized successfully when using the sum of the 4 largest Singular values for each image.

6.5.3.4.Results using Multi-Input Neural Network

A multi-input neural network has been trained to recognize 32 facial images in Figure 6.10.

The structure used has the following specifications: 3 neurons in the input layer, 7 neurons in the first hidden layer, 15 neurons in the second hidden layer, 20 neurons in the third hidden layer, and one neuron in the output layer. Backporpagation algorithm has been used in training.

Three criteria have been introduced to the multi-input neural network:

1) The sum of the 4 x 4 spatial low frequency components in the DCT domain for each image.

2) The sum of the 4 x 4 low sequency HAAR coefficients for each image.

3) The sum of the ten largest singular values for each image.

Different mean square errors have been tried.

The following results are obtained:

1) 10^{-1} MSE, 8 images out of 32 images are recognized correctly.

2) 10^{-3} MSE, 8 images out of 32 images are recognized correctly.

3) 10^{-5} MSE, only 4 images out of 32 are recognized correctly.

Figure 6.11: Thirty-two noisy facial images

CHAPTER SEVEN: CONCLUSIONS AND FUTURE WORK

7.1. Conclusions

In this dissertation, a novel one and multidimensional signal recognition technique using multicriteria processing and data fusion has been proposed.

The signal recognition process is performed in 2 stages, feature extraction followed by classification. Some criteria are computed from the features extracted from the signals when projected into different transform domains. Other criteria are developed from the signals after preprocessing. Canny, Prewitt, Zerocrossing and Roberts Edge Detection techniques have been employed to extract some useful features of the input signals. Different classification techniques have been attempted such as the classification decision tree, neural networks, vector quantization, classification such that the distance between the groups are maximum, clustering the signals to groups of the same statistical properties, etc.

The proposed algorithm for the pattern recognition system is described in detail in Chapter 3. The system operates in two modes: the learning mode and the running mode.

1) In the learning mode, the optimum set of criteria satisfying certain conditions such as simplicity, high immunity against noise… etc is selected by the system to classify the signals at each stage of the multi-stage classifier.

2) In the running mode, the signals to be recognized, are presented at each node, in each stage and assigned an index based on the selected criteria, in the training mode, and according to the classification technique used. Finally, at the terminal nodes, each signal is identified by a unique composite index.

Chapter 4 presents some of the experimental results obtained when classifying images using certain types of classification techniques, based on different types of criteria. It also presents the effect of using preprocessing techniques on the classification accuracy. The results obtained show that the optimum set of criteria and most suitable classification approach for recognizing any signal is problem dependent. In other words, the classification accuracy resulting from using a particular criterion and a specific classification method differs from one problem to the other. Therefore, the ability of the proposed pattern recognition system to select the optimum criteria and the most suitable classification technique that suits the problem under consideration leads to superior classification accuracy relative to some existing techniques.

Chapter five presents some implementation examples illustrating the performance of the proposed technique and its high accuracy in image recognition when using the parallel structure. The system in these examples employs NN's in parallel and three classification criteria obtained from the image projections in three transform domains. The obtained results are compared to a traditional, Single Transform Neural Network, implementation. It is also compared to Multi-Input Neural Networks. The comparison proves the feasibility of the proposed technique and its improved performance in terms of an appreciable reduction in the overall computational complexity, increased speed and high accuracy.

Some experimental results are also presented in Chapter 6 demonstrating the performance of the proposed pattern recognition system using the cascaded structure in the form of the classification decision tree. The proposed approach has been applied to facial images, as well as biomedical images. The results obtained verify the high efficiency, the reduced computational complexity, and superior classification accuracy of the proposed system, relative to Single Input Neural Network and Multi-Input Neural Network.

The large number of available criteria made the proposed pattern recognition system capable of recognizing different types of signals. It has the property of evolutionary learning by selecting the criteria as well as optimizing each criterion for best overall performance in terms of speed, accuracy, and complexity. The proposed system has also shown a great efficiency for different classification environments such as noisy and incomplete data.

7.2. Future Work

Although different types of criteria and classification methods have been examined, throughout this dissertation, by the proposed pattern recognition system, there is still no general approach available that can be applied for all cases. Therefore, more work must be done to examine more criteria and more classification techniques to enhance the performance of the suggested pattern recognition system for all types of problems.

APPENDIX

SOURCE CODE OF DEVELOPED ALGORITHMS

%The main goal of this algothm is to identify any input signal with a composit index based on the groups numbers. Many criteria are tested for grouping the signals . The selected set of criteria gives the best classification accuracy in case of corrupted data. Also a voting scheme is employed to enhance the performace of the system.

```
clc
clear all
ni=input('enter the number of images  ');
g=input('the number of groups   ');
gs=input('enter which set of images as noisy images    ');
k=1;
maxz=input('enter the maximum number of times we do regrouping  ');
nc=4;
nt=1;
nmc=input('enter the number of criteria doing the same function     ');
ns=input('enter the maximum number of gray levels added for noise   ');
nb=input('enter the size of image  ');
fi=input('enter the first image we start with    ');
fil=input('enter ced for edge and c for orignal  ');
system2(nb,ni,g,k,maxz,ns,gs,fi,fil);
regroupcritmg(ni,nt,g);
[ss]=similarcrit(ni,nmc);
votepart(ni);              % This part is for votes of partial accuracy
votemg(ni,ss,g);
out=zeros(ni,3);
for i=1:ni
  [wr,vtn]=votesgim(ni,g,i);
  out(i,1)=i;
  out(i,2)=vtn;
  out(i,3)=wr;
end
save out.dat out -ascii –double


function []=regroupcritmg(ni,nt,ng)

% This function is used to regroup images  into ng groups
% number of stages used =n
% nt=number of trials
n=log(ni)/log(2);

% imagind is a matrix of all images and their indices produced from
% the different criteria and this criteria,
for v=1:nt
    fd=['s' num2str(v) '.dat' ];
```

```
    fp=fopen(fd,'w');
    imagind=zeros(ni+1,ni+2);
    for i=1:ni
       imagind(1,i+2)=i;
end
    fprintf(fp,'%d\n',imagind(1,:));
    % lets choose the first criterion to be dct.
    load indexdct2.dat
    load indexdct.dat
    load indexhaar.dat
    load indexsvd.dat
    load indexedsvd.dat
    load indexenav.dat
    load indexmxmndct.dat
    load indexmxmnhaar.dat
    load edgfor.dat
    load indexwav.dat
    load indexedwav.dat


    [rd,cd]=size(indexdct);
    [rh,ch]=size(indexhaar);
    [rs,cs]=size(indexsvd);
    [res,ces]=size(indexedsvd);
    [re,ce]=size(indexenav);
    [rdm,cdm]=size(indexmxmndct);
    [rhm,chm]=size(indexmxmnhaar);
    [ref,cef]=size(edgfor);
    [rw,cw]=size(indexwav);
    [rwe,cwe]=size(indexedwav);


    y=zeros(2,(ni+2));
    y(1,:)=indexdct(1,(ni+2));
    y(1,1)=0;
    y(1,2)=0;
    %for u=1:(ni+2)
    % imagind(2,u)=1;   % this step because we don't know which criteria
    %end


    % It is better to start with crtieria that gives zero errors then see
    % the groups are almost the same.
    st=zeros(1,(ni+4));
    st(1,(ni+4))=ni;
```

```
for i=1:rd/2
   if indexdct((2*i-1),(ni+3))==ni
      l=indexdct((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
         st(1,:)=indexdct((2*i-1),1:(ni+4));
      end
   end
end

for i=1:rh/2
   if indexhaar((2*i-1),(ni+3))==ni
      l=indexhaar((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
         st(1,:)=indexhaar((2*i-1),1:(ni+4));
      end
   end
end

for i=1:rdm/2
   if indexmxmndct((2*i-1),(ni+3))==ni
      l=indexmxmndct((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
         st(1,:)=indexmxmndct((2*i-1),1:(ni+4));
      end
   end
end

for i=1:rhm/2
   if indexmxmnhaar((2*i-1),(ni+3))==ni
      l=indexmxmnhaar((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
         st(1,:)=indexmxmnhaar((2*i-1),1:(ni+4));
      end
   end
end

for i=1:rs/2
   if indexsvd((2*i-1),(ni+3))==ni
      l=indexsvd((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
         st(1,:)=indexsvd((2*i-1),1:(ni+4));
```

```
      end
    end
  end

  for i=1:rw/2
    if indexwav((2*i-1),(ni+3))==ni
      l=indexwav((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
        st(1,:)=indexwav((2*i-1),1:(ni+4));
      end
    end
end

for i=1:rwe/2
    if indexedwav((2*i-1),(ni+3))==ni
      l=indexedwav((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
        st(1,:)=indexedwav((2*i-1),1:(ni+4));
      end
    end
end


for i=1:res/2
    if indexedsvd((2*i-1),(ni+3))==ni
      l=indexedsvd((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
        st(1,:)=indexedsvd((2*i-1),1:(ni+4));
      end
    end
end

for i=1:re/2
    if indexenav((2*i-1),(ni+3))==ni
      l=indexenav((2*i-1),(ni+4));
      l1=ni-l;
      if (abs(l-l1))<st(1,(ni+4))
        st(1,:)=indexenav((2*i-1),1:(ni+4));
      end
    end
end

for i=1:ref/2
```

```
    if edgfor((2*i-1),(ni+3))==ni
        l=edgfor((2*i-1),(ni+4));
        l1=ni-l;
        if (abs(l-l1))<st(1,(ni+4))
            st(1,:)=edgfor((2*i-1),1:(ni+4));
        end
    end
end

imagind(2,1:(ni+3))=st(1,1:(ni+3));
imagind(2,1:(ni+3))


ii=2;
i=1;

while(i<(ni))
    i=i+1;
    % we don't take the row if there is only one image in group.
    q=0;


    for a=3:(ni+2)
        if imagind(i,a)~=0
            q=q+1;
        end
    end
    if (q==0 |q==1)
        i=i+1;
    end

    q2=0;
    for w=3:(ni+1)
        for ww=(w+1):(ni+2)
            if (imagind(i,w)==imagind(i,ww) & imagind(i,w)~=0)
                q2=q2+1;
            end
        end
    end
    if (q2==0)
        i=i+1;
    end


    y(1,:)=imagind(1,1:(ni+2));
    y(2,:)=imagind((i),1:(ni+2));
```

```
      [temp,in,nni]=diffcritmg(y,ni,ng);

      %nni is the new number of images grouped without zeros.
      % this part so that no zero rows
      for k=1:ng
         q=0;
         for a=3:in(1,k)+2
            if temp(3*k,a)~=0
               q=q+1;
            end
         end

         if q~=0
            ii=ii+1
            for jj=3:ni+2
               for j=3:in(1,k)+2
                  if imagind(1,jj)==temp((3*k-2),j)
                     imagind((ii),1:2)=temp(3*k,1:2);
                     imagind((ii),jj)=temp(3*k,j);
                     imagind(ii,(ni+3))=temp(3*k,(in(1,k)+4));
                  end
               end
            end
         end
         save imagind.dat imagind -ascii -double
         % c11, c12 are the criteria used and we don't want to use it again
         c11=imagind(ii,1);
         c12=imagind(ii,2);

         fprintf(fp,'%d\n',imagind(ii,:));
      end
   end
      fclose(fp);
end

save imagind.dat imagind -ascii –double


% This function is used to find nc criteria giving the same grouping according to
% imagind.dat
function [ss]=similarcrit(ni,nc)
%if dct starts with 7
% if haar starts with 8
% if svd starts with 9
% if dct 333 1, it is the same as dct 335 1
```

```
% nc is the number of similar criteria
%fd='s1.dat';
%fid=fopen(fd,'r');        %inorder to open file
%x=fscanf(fid,'%lf',[ni+1 ni+2]);      %it reads the elements into column
%fclose(fid);
load imagind.dat
[ri,ci]=size(imagind);
% This part romoves the zero rows from imagind.dat
for s=2:ri
  %if imagind(s,1)==0
  if imagind(s,1)~=0       % the previous line is not working with multigrouping
    %  x=zeros((s-1),ci);
    % x(1:(s),1:ci)=imagind(1:(s),1:ci);
    ss=s;
  end
end
x=imagind;
% % A is a matrix of all criteria which gives the same grouping according to s1.

load indexdct.dat
load indexhaar.dat
load indexmxmndct.dat
load indexmxmnhaar.dat
load indexsvd.dat
load indexedsvd.dat
load indexenav.dat
load edgfor.dat
load indexwav.dat
load indexedwav.dat



[rd,cd]=size(indexdct);
[rh,ch]=size(indexhaar);
[rdm,cdm]=size(indexmxmndct);
[rhm,chm]=size(indexmxmnhaar);
[rs,cs]=size(indexsvd);
[res,ces]=size(indexsvd);
[re,ce]=size(indexenav);
[ref,cef]=size(edgfor);
[rw,cw]=size(indexwav);
[rwe,cwe]=size(indexedwav);


%A=zeros((ss),(3*nc));
A=zeros((ri),(3*nc));
```

```
size(A)
size(x)
A(:,1:2)=x(:,1:2);
A(:,3)=x(:,(ni+3));
% matrix of criteria doing same function.
a1=zeros((2*(ni-1)*nc),(ni+3));
a=0;
%for i=2:ss
for i=2:ss
  A(i,1:2)=x((i),1:2);
  A(i,3)=0;
  t=3;

  for w=1:rdm/2
    if t<((3*nc)+1)
      l=0;
      for j=3:(ni+2)
        if x(i,j)~=0;
          if x(i,j)~=indexmxmndct((2*w-1),j);
            l=l+1;
          end
        end
      end

      % this part is to make sure that if dct 345,1 is used, 355, 1 isn't used again
      k=0;
      for tt=1:t/3
        yy=num2str(A(i,(3*tt-2)));
        yyyy=yy(1,2);
        yp=yy(1,3);
        xx=num2str(indexmxmndct((2*w-1),1));
        xxxx=xx(1,2);
        xp=xx(1,3);
        yyy=str2num(yyyy);
        xxx=str2num(xxxx);
        ypp=str2num(yp);
        xpp=str2num(xp);

        if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexmxmndct((2*w-1),2)==2)
          k=2;
        end

        if ((A(i,(3*tt-2)))==indexmxmndct((2*w-1),1) & A(i,(3*tt-1))==indexmxmndct((2*w-1),2))
          k=2;
        end
```

```
      if (yyy==xxx & A(i,(3*tt-1))==1 & indexmxmndct((2*w-1),2)==1)
        k=2;
      end
    end
    if (l==0 & k==0)
      t=t+1;
      o=indexmxmndct((2*w-1),1);
      dd=[ num2str(o)];
      A(i,t)=sscanf(dd,'%f');
      t=t+1;
      A(i,t)=indexmxmndct((2*w-1),2);
      t=t+1;
      A(i,t)=indexmxmndct((2*w-1),ni+3);
      a=a+1;
      a1(a,1:(ni+3))=indexmxmndct((2*w-1),1:(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexmxmndct((2*w),1:(ni+3));
    end
  end
end

for w=1:rhm/2

  if t<((3*nc)+1)

    l=0;
    for j=3:(ni+2)
      if x(i,j)~=0;
        if x(i,j)~=indexmxmnhaar((2*w-1),j);
          l=l+1;
        end
      end
    end
    k=0;

    for tt=1:t/3
      yy=num2str(A(i,(3*tt-2)));
      yyyy=yy(1,2);
      yp=yy(1,3);
      xx=num2str(indexmxmnhaar((2*w-1),1));
      xxxx=xx(1,2);
      xp=xx(1,3);
      yyy=str2num(yyyy);
      xxx=str2num(xxxx);
      ypp=str2num(yp);
      xpp=str2num(xp);
```

```
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexmxmnhaar((2*w-1),2)==2)
      k=2;
    end
    if ((A(i,(3*tt-2)))==indexmxmnhaar((2*w-1),1) & A(i,(3*tt-1))==
                                                    indexmxmnhaar((2*w-1),2))
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexmxmnhaar((2*w-1),2)==1)
      k=2;
    end
   end
   if (l==0 & k==0)

    t=t+1;
    o=indexmxmnhaar((2*w-1),1);
    %for haar
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=indexmxmnhaar((2*w-1),2);
    t=t+1;
    A(i,t)=indexmxmnhaar((2*w-1),(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexmxmnhaar((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexmxmnhaar((2*w),1:(ni+3));
   end
  end
end

for w=1:re/2
  if t<((3*nc)+1)
    l=0;
    for j=3:(ni+2)
      if x(i,j)~=0;
        if x(i,j)~=indexenav((2*w-1),j);
          l=l+1;
        end
      end
    end

    % this part is to make sure that if dct 345,1 is used, 355, 1 isn't used again
    k=0;
    for tt=1:t/3
      yy=num2str(A(i,(3*tt-2)));
```

```
      yyyy=yy(1,2);
      yp=yy(1,3);
      xx=num2str(indexenav((2*w-1),1));
      xxxx=xx(1,2);
      xp=xx(1,3);
      yyy=str2num(yyyy);
      xxx=str2num(xxxx);
      ypp=str2num(yp);
      xpp=str2num(xp);
      if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexenav((2*w-1),2)==2)
        k=2;
      end

      if ((A(i,(3*tt-2)))==indexenav((2*w-1),1) & A(i,(3*tt-1))==indexenav((2*w-1),2))
        k=2;
      end

      if (yyy==xxx & A(i,(3*tt-1))==1 & indexenav((2*w-1),2)==1)
        k=2;
      end
    end
    if (l==0 & k==0)
      t=t+1;
      o=indexenav((2*w-1),1);
      dd=[ num2str(o)];
      A(i,t)=sscanf(dd,'%f');
      t=t+1;
      A(i,t)=indexenav((2*w-1),2);
      t=t+1;
      A(i,t)=indexenav((2*w-1),ni+3);
      a=a+1;
      a1(a,1:(ni+3))=indexenav((2*w-1),1:(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexenav((2*w),1:(ni+3));
    end
  end
end

for w=1:rd/2
  if t<((3*nc)+1)
    l=0;
    for j=3:(ni+2)
      if x(i,j)~=0;
        if x(i,j)~=indexdct((2*w-1),j);
          l=l+1;
        end
```

```
    end
  end

  % this part is to make sure that if dct 345,1 is used, 355, 1 isn't used aagin
  k=0;
  for tt=1:t/3
    yy=num2str(A(i,(3*tt-2)));
    yyyy=yy(1,2);
    yp=yy(1,3);
    xx=num2str(indexdct((2*w-1),1));
    xxxx=xx(1,2);
    xp=xx(1,3);
    yyy=str2num(yyyy);
    xxx=str2num(xxxx);
    ypp=str2num(yp);
    xpp=str2num(xp);
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexdct((2*w-1),2)==2)
      k=2;
    end

    if ((A(i,(3*tt-2)))==indexdct((2*w-1),1) & A(i,(3*tt-1))==indexdct((2*w-1),2))
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexdct((2*w-1),2)==1)
      k=2;
    end
  end
  if (l==0 & k==0)
    t=t+1;
    o=indexdct((2*w-1),1);
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=indexdct((2*w-1),2);
    t=t+1;
    A(i,t)=indexdct((2*w-1),ni+3);
    a=a+1;
    a1(a,1:(ni+3))=indexdct((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexdct((2*w),1:(ni+3));
  end
 end
end

for w=1:rh/2
```

```
if t<((3*nc)+1)

  l=0;
  for j=3:(ni+2)
    if x(i,j)~=0;
      if x(i,j)~=indexhaar((2*w-1),j);
        l=l+1;
      end
    end
  end
  k=0;

  for tt=1:t/3
    yy=num2str(A(i,(3*tt-2)));
    yyyy=yy(1,2);
    yp=yy(1,3);
    xx=num2str(indexhaar((2*w-1),1));
    xxxx=xx(1,2);
    xp=xx(1,3);
    yyy=str2num(yyyy);
    xxx=str2num(xxxx);
    ypp=str2num(yp);
    xpp=str2num(xp);
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexhaar((2*w-1),2)==2)
      k=2;
    end
    if ((A(i,(3*tt-2)))==indexhaar((2*w-1),1) & A(i,(3*tt-1))==indexhaar((2*w-1),2))
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexhaar((2*w-1),2)==1)
      k=2;
    end
  end
  if (l==0 & k==0)

    t=t+1;
    o=indexhaar((2*w-1),1);
    %for haar
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=indexhaar((2*w-1),2);
    t=t+1;
    A(i,t)=indexhaar((2*w-1),(ni+3));
```

```
      a=a+1;
      a1(a,1:(ni+3))=indexhaar((2*w-1),1:(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexhaar((2*w),1:(ni+3));
    end
  end
end

for w=1:rs/2
  if t<((3*nc)+1)

    l=0;
    for j=3:(ni+2)
      if x(i,j)~=0;
        if x(i,j)~=indexsvd((2*w-1),j);
          l=l+1;
        end
      end
    end

    k=0;

    for tt=1:t/3
      yy=num2str(A(i,(3*tt-2)));
      yyyy=yy(1,2);
      yp=yy(1,3);
      xx=num2str(indexsvd((2*w-1),1));
      xxxx=xx(1,2);
      xp=xx(1,3);
      yyy=str2num(yyyy);
      xxx=str2num(xxxx);
      ypp=str2num(yp);
      xpp=str2num(xp);
      if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexsvd((2*w-1),2)==2)
        k=2;
      end
      if ((A(i,(3*tt-2)))==indexsvd((2*w-1),1) & A(i,(3*tt-1))==indexsvd((2*w-1),2))
        k=2;
      end

      if (yyy==xxx & A(i,(3*tt-1))==1 & indexsvd((2*w-1),2)==1)
        k=2;
      end
    end
    if (l==0 & k==0)
```

```
      t=t+1;
      o=indexsvd((2*w-1),1);
      dd=[ num2str(o)];
      A(i,t)=sscanf(dd,'%f');
      t=t+1;
      A(i,t)=indexsvd((2*w-1),2);
      t=t+1;
      A(i,t)=indexsvd((2*w-1),(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexsvd((2*w-1),1:(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexsvd((2*w),1:(ni+3));
    end
   end
  end
end

for w=1:res/2
 if t<((3*nc)+1)

   l=0;
   for j=3:(ni+2)
    if x(i,j)~=0;
     if x(i,j)~=indexedsvd((2*w-1),j);
      l=l+1;
     end
    end
   end

   k=0;

   for tt=1:t/3
    yy=num2str(A(i,(3*tt-2)));
    yyyy=yy(1,2);
    yp=yy(1,3);
    xx=num2str(indexedsvd((2*w-1),1));
    xxxx=xx(1,2);
    xp=xx(1,3);
    yyy=str2num(yyyy);
    xxx=str2num(xxxx);
    ypp=str2num(yp);
    xpp=str2num(xp);
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexedsvd((2*w-1),2)==2)
     k=2;
    end
    if ((A(i,(3*tt-2)))==indexedsvd((2*w-1),1) & A(i,(3*tt-1))==indexedsvd((2*w-1),2))
```

```
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexedsvd((2*w-1),2)==1)
      k=2;
    end
  end
  if (l==0 & k==0)

    t=t+1;
    o=indexedsvd((2*w-1),1);
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=indexedsvd((2*w-1),2);
    t=t+1;
    A(i,t)=indexedsvd((2*w-1),(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexedsvd((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexedsvd((2*w),1:(ni+3));
  end
 end
end
end

for w=1:rw/2
 if t<((3*nc)+1)

  l=0;
  for j=3:(ni+2)
    if x(i,j)~=0;
      if x(i,j)~=indexwav((2*w-1),j);
        l=l+1;
      end
    end
  end

  k=0;

  for tt=1:t/3
    yy=num2str(A(i,(3*tt-2)));
    yyyy=yy(1,2);
    yp=yy(1,3);
    xx=num2str(indexwav((2*w-1),1));
    xxxx=xx(1,2);
```

```
    xp=xx(1,3);
    yyy=str2num(yyyy);
    xxx=str2num(xxxx);
    ypp=str2num(yp);
    xpp=str2num(xp);
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexwav((2*w-1),2)==2)
      k=2;
    end
    if ((A(i,(3*tt-2)))==indexwav((2*w-1),1) & A(i,(3*tt-1))==indexwav((2*w-1),2))
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexwav((2*w-1),2)==1)
      k=2;
    end
   end
   if (l==0 & k==0)

    t=t+1;
    o=indexwav((2*w-1),1);
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=indexwav((2*w-1),2);
    t=t+1;
    A(i,t)=indexwav((2*w-1),(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexwav((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexwav((2*w),1:(ni+3));
   end
  end
end
end

for w=1:rwe/2
  if t<((3*nc)+1)

   l=0;
   for j=3:(ni+2)
    if x(i,j)~=0;
      if x(i,j)~=indexedwav((2*w-1),j);
        l=l+1;
      end
    end
   end
```

```
    k=0;

    for tt=1:t/3
      yy=num2str(A(i,(3*tt-2)));
      yyyy=yy(1,2);
      yp=yy(1,3);
      xx=num2str(indexedwav((2*w-1),1));
      xxxx=xx(1,2);
      xp=xx(1,3);
      yyy=str2num(yyyy);
      xxx=str2num(xxxx);
      ypp=str2num(yp);
      xpp=str2num(xp);
      if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexedwav((2*w-1),2)==2)
        k=2;
      end
      if ((A(i,(3*tt-2)))==indexedwav((2*w-1),1) & A(i,(3*tt-1))==indexedwav((2*w-1),2))
        k=2;
      end

      if (yyy==xxx & A(i,(3*tt-1))==1 & indexedwav((2*w-1),2)==1)
        k=2;
      end
    end
    if (l==0 & k==0)

      t=t+1;
      o=indexedwav((2*w-1),1);
      dd=[ num2str(o)];
      A(i,t)=sscanf(dd,'%f');
      t=t+1;
      A(i,t)=indexedwav((2*w-1),2);
      t=t+1;
      A(i,t)=indexedwav((2*w-1),(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexedwav((2*w-1),1:(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexedwav((2*w),1:(ni+3));
    end
  end
end
end
```

```
for w=1:rdm/2
  if t<((3*nc)+1)
    l=0;
    for j=3:(ni+2)
      if x(i,j)~=0;
        if x(i,j)~=indexmxmndct((2*w-1),j);
          l=l+1;
        end
      end
    end

    % this part is to make sure that if dct 345,1 is used, 355, 1 isn't used again
    k=0;
    for tt=1:t/3
      yy=num2str(A(i,(3*tt-2)));
      yyyy=yy(1,2);
      yp=yy(1,3);
      xx=num2str(indexmxmndct((2*w-1),1));
      xxxx=xx(1,2);
      xp=xx(1,3);
      yyy=str2num(yyyy);
      xxx=str2num(xxxx);
      ypp=str2num(yp);
      xpp=str2num(xp);
      if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexmxmndct((2*w-1),2)==2)
        k=2;
      end

      if ((A(i,(3*tt-2)))==indexmxmndct((2*w-1),1) & A(i,(3*tt-1))==indexmxmndct((2*w-1),2))
        k=2;
      end

      if (yyy==xxx & A(i,(3*tt-1))==1 & indexmxmndct((2*w-1),2)==1)
        k=2;
      end
    end
    if (l==0 & k==0)
      t=t+1;
      o=indexmxmndct((2*w-1),1);
      dd=[ num2str(o)];
      A(i,t)=sscanf(dd,'%f');
      t=t+1;
      A(i,t)=indexmxmndct((2*w-1),2);
      t=t+1;
      A(i,t)=indexmxmndct((2*w-1),ni+3);
      a=a+1;
```

```
    a1(a,1:(ni+3))=indexmxmndct((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexmxmndct((2*w),1:(ni+3));
   end
  end
end

for w=1:rhm/2

  if t<((3*nc)+1)

   l=0;
   for j=3:(ni+2)
    if x(i,j)~=0;
     if x(i,j)~=indexmxmnhaar((2*w-1),j);
      l=l+1;
     end
    end
   end
   k=0;

   for tt=1:t/3
    yy=num2str(A(i,(3*tt-2)));
    yyyy=yy(1,2);
    yp=yy(1,3);
    xx=num2str(indexmxmnhaar((2*w-1),1));
    xxxx=xx(1,2);
    xp=xx(1,3);
    yyy=str2num(yyyy);
    xxx=str2num(xxxx);
    ypp=str2num(yp);
    xpp=str2num(xp);
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexmxmnhaar((2*w-1),2)==2)
     k=2;
    end
    if ((A(i,(3*tt-2)))==indexmxmnhaar((2*w-1),1)
        & A(i,(3*tt-1))==indexmxmnhaar((2*w-1),2))
     k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexmxmnhaar((2*w-1),2)==1)
     k=2;
    end
   end
   if (l==0 & k==0)
```

```
      t=t+1;
      o=indexmxmnhaar((2*w-1),1);
      %for haar
      dd=[ num2str(o)];
      A(i,t)=sscanf(dd,'%f');
      t=t+1;
      A(i,t)=indexmxmnhaar((2*w-1),2);
      t=t+1;
      A(i,t)=indexmxmnhaar((2*w-1),(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexmxmnhaar((2*w-1),1:(ni+3));
      a=a+1;
      a1(a,1:(ni+3))=indexmxmnhaar((2*w),1:(ni+3));
    end
  end
end

for w=1:ref/2
  if t<((3*nc)+1)
    l=0;
    for j=3:(ni+2)
      if x(i,j)~=0;
        if x(i,j)~=edgfor((2*w-1),j);
          l=l+1;
        end
      end
    end

    % this part is to make sure that if dct 345,1 is used, 355, 1 isn't used again
    k=0;
    for tt=1:t/3
      yy=num2str(A(i,(3*tt-2)));
      yyyy=yy(1,2);
      yp=yy(1,3);
      xx=num2str(edgfor((2*w-1),1));
      xxxx=xx(1,2);
      xp=xx(1,3);
      yyy=str2num(yyyy);
      xxx=str2num(xxxx);
      ypp=str2num(yp);
      xpp=str2num(xp);

      if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & edgfor((2*w-1),2)==2)
        k=2;
      end
```

```
    if ((A(i,(3*tt-2)))==edgfor((2*w-1),1) & A(i,(3*tt-1))==edgfor((2*w-1),2))
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & edgfor((2*w-1),2)==1)
      k=2;
    end
   end
   if (l==0 & k==0)
    t=t+1;
    o=edgfor((2*w-1),1);
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=edgfor((2*w-1),2);
    t=t+1;
    A(i,t)=edgfor((2*w-1),ni+3);
    a=a+1;
    a1(a,1:(ni+3))=edgfor((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=edgfor((2*w),1:(ni+3));
   end
  end
end


for w=1:re/2
  if t<((3*nc)+1)
   l=0;
   for j=3:(ni+2)
    if x(i,j)~=0;
     if x(i,j)~=indexenav((2*w-1),j);
      l=l+1;
     end
    end
   end

   % this part is to make sure that if dct 345,1 is used, 355, 1 isn't used again
   k=0;
   for tt=1:t/3
    yy=num2str(A(i,(3*tt-2)));
    yyyy=yy(1,2);
    yp=yy(1,3);
    xx=num2str(indexenav((2*w-1),1));
    xxxx=xx(1,2);
    xp=xx(1,3);
```

```
    yyy=str2num(yyyy);
    xxx=str2num(xxxx);
    ypp=str2num(yp);
    xpp=str2num(xp);
    if (yyy==xxx & ypp==xpp & A(i,(3*tt-1))==2 & indexenav((2*w-1),2)==2)
      k=2;
    end

    if ((A(i,(3*tt-2)))==indexenav((2*w-1),1) & A(i,(3*tt-1))==indexenav((2*w-1),2))
      k=2;
    end

    if (yyy==xxx & A(i,(3*tt-1))==1 & indexenav((2*w-1),2)==1)
      k=2;
    end
   end
   if (l==0 & k==0)
    t=t+1;
    o=indexenav((2*w-1),1);
    dd=[ num2str(o)];
    A(i,t)=sscanf(dd,'%f');
    t=t+1;
    A(i,t)=indexenav((2*w-1),2);
    t=t+1;
    A(i,t)=indexenav((2*w-1),ni+3);
    a=a+1;
    a1(a,1:(ni+3))=indexenav((2*w-1),1:(ni+3));
    a=a+1;
    a1(a,1:(ni+3))=indexenav((2*w),1:(ni+3));
   end
  end
end


%crtfn is a matrix of criteria doing the same function
save crtfn.dat A -ascii -double
cor=zeros(a,(ni+3));
%cor(:,1:(ni+3))=a1(1:a,1:(ni+3));
%save noiscorrect.dat cor -ascii -double


% This function is used to develop some criteria and compute the accuracy of each of them
  in case of noisy images

% 1. vQ          2.equal statistical groups and nnet
```

```
function []=system2(nb,ni,g,k,maxz,ns,gs,fi,fil)
% fil indicates if the image is used or the edges in the transforms.
%nb=size(image)
q=0;

d=basis(nb);
dctd= d(1:nb,1:nb);
haard= d((nb+1):(2*nb),1:nb);
for hi=3:1:5            %the first coefficients =0 after average removede
  for mid=3:1:5
    for low=3:1:5
      [fd,fh,fdn,fhn]=sumcoeff1(nb,dctd,haard,ni,hi,mid,low,ns,gs,fi,fil);  %returns the files names
      for v=1:3
        nc=4;
        q=q+1;
        %ddct=grouping(fd,ni,v,g);  %returns target vectors naming each image with
        %[n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fd,fdn,ni,v,nc);     %grouping such as
        statistical property is the same
        [n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fd,fdn,ni,v,nc,g);   %using vector
        quantization
        dd=[num2str(7) num2str(hi) num2str(mid) num2str(low)];
        indexdct((2*q-1),1)= sscanf(dd,'%f');
        indexdct((2*q-1),2)=v;
        indexdct((2*q-1),3:(ni+2))=ddct(1,1:ni);
        indexdct((2*q-1),ni+5)=m1;
        indexdct((2*q-1),ni+6)=m2;
        % The  following function is to find the grouping of noisy images when
        % using nnet.

        %[xx,srgr]=nnetgroupingsumcoeff(fd,fdn,ddct,ni,v,nc,k);
        indexdct((2*q-1),ni+3)=srgr;
        indexdct((2*q-1),ni+4)=n1;
        indexdct((2*q-1),ni+7)=gap;
        indexdct((2*q),1)= sscanf(dd,'%f');
        indexdct((2*q),2)=v;
        indexdct((2*q),3:(ni+2))=xx(1,1:ni);
        indexdct2((2*q-1),1)= sscanf(dd,'%f');
        indexdct2((2*q-1),2)=v;
        indexdct2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
        indexdct2((2*q),1)= sscanf(dd,'%f');
        indexdct2((2*q),2)=v;
        indexdct2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
        indexdct2((2*q-1),ni+5)=m1;
        indexdct2((2*q-1),ni+6)=m2;
        indexdct2((2*q-1),ni+3)=srgr;
        indexdct2((2*q-1),ni+4)=n1;
```

```
      indexdct2((2*q-1),ni+7)=gap;
      save indexdct2.dat indexdct2 -ascii -double
      save indexdct.dat indexdct -ascii -double

      %dhaar=grouping(fh,ni,v,g);                    % group number
      [n1,dhaar,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fh,fhn,ni,v,nc);
      %[dhaar,jj,xf]=tv(fd,fdn,ni,v,nc);
          %[n1,dhaar,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fh,fhn,ni,v,nc,g);   %using vector
         quantization
      dd=[num2str(8) num2str(hi) num2str(mid) num2str(low)];
      indexhaar((2*q-1),1)= sscanf(dd,'%f');
      indexhaar((2*q-1),2)=v;
      indexhaar((2*q-1),3:(ni+2))=dhaar(1,1:ni);
      indexhaar((2*q-1),ni+5)=m1;
      indexhaar((2*q-1),ni+6)=m2;

      % The  following function is to find the grouping of noisy images when
      % using nnet.
      % It doesn't take care of indexdct2
      [xx,srgr]=nnetgroupingsumcoeff(fh,fhn,dhaar,ni,v,nc,k);
      indexhaar((2*q-1),ni+3)=srgr;
      indexhaar((2*q-1),ni+4)=n1;
      indexhaar((2*q-1),ni+7)=gap;
      indexhaar((2*q),1)= sscanf(dd,'%f');
      indexhaar((2*q),2)=v;
      indexhaar((2*q),3:(ni+2))=xx(1,1:ni);
      indexhaar2((2*q-1),1)= sscanf(dd,'%f');
      indexhaar2((2*q-1),2)=v;
      indexhaar2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
      indexhaar2((2*q),1)= sscanf(dd,'%f');
      indexhaar2((2*q),2)=v;
      indexhaar2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
      indexhaar2((2*q-1),ni+5)=m1;
      indexhaar2((2*q-1),ni+6)=m2;
      indexhaar2((2*q-1),ni+3)=srgr;
      indexhaar2((2*q-1),ni+4)=n1;
      indexhaar2((2*q-1),ni+7)=gap;
      save indexhaar2.dat indexhaar2 -ascii -double
      save indexhaar.dat indexhaar -ascii -double
    end
   end
  end
end

% This part is used to find the wavelet of images, then edges, then cosine transform.
q=0;
```

```
fil='ced';
[rc]=edwavlet(ni,2,1,1,nb,fil,ns);       %This step is only for computation of rc
d=basis(rc);
dctd= d(1:rc,1:rc);
haard= d((rc+1):(2*rc),1:rc);
for hi=3:1:5            %the first coefficients =0 after removing the average
  for mid=3:1:5
    for low=3:1:5
      for a=1:4
        for aa=1:3
          le=2;
          [rc]=edwavlet(ni,le,a,aa,nb,fil,ns);
          [fdcos,fdcosn]=sumedcoefficientcos(rc,dctd,haard,ni,hi,mid,low,ns,gs,fi,fil,a,aa)
          for v=1:3
            nc=4;
            q=q+1;
            %ddct=grouping(fd,ni,v,g);  %returns target vectors naming each image with
            %[n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fd,fdn,ni,v,nc);     %grouping such
            as statistical property is the same
           [n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fdcos,fdcosn,ni,v,nc,g);   %using vector
           quantization
            dd=[num2str(7) num2str(a) num2str(aa) num2str(hi) num2str(mid) num2str(low)];
            indexedwavdct((2*q-1),1)= sscanf(dd,'%f');
            indexedwavdct((2*q-1),2)=v;
            indexedwavdct((2*q-1),3:(ni+2))=ddct(1,1:ni);
            indexedwavdct((2*q-1),ni+5)=m1;
            indexedwavdct((2*q-1),ni+6)=m2;
            % The  following function is to find the grouping of noisy images when
            % using nnet.
            % It doesn't take care of indexdct2
            %[xx,srgr]=nnetgroupingsumcoeff(fd,fdn,ddct,ni,v,nc,k);
            indexedwavdct((2*q-1),ni+3)=srgr;
            indexedwavdct((2*q-1),ni+4)=n1;
            indexedwavdct((2*q-1),ni+7)=gap;
            indexedwavdct((2*q),1)= sscanf(dd,'%f');
            indexedwavdct((2*q),2)=v;
            indexedwavdct((2*q),3:(ni+2))=xx(1,1:ni);
            indexedwavdct2((2*q-1),1)= sscanf(dd,'%f');
            indexedwavdct2((2*q-1),2)=v;
            indexedwavdct2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
            indexedwavdct2((2*q),1)= sscanf(dd,'%f');
            indexedwavdct2((2*q),2)=v;
            indexedwavdct2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
            indexedwavdct2((2*q-1),ni+5)=m1;
            indexedwavdct2((2*q-1),ni+6)=m2;
            indexedwavdct2((2*q-1),ni+3)=srgr;
```

```
          indexedwavdct2((2*q-1),ni+4)=n1;
          indexedwavdct2((2*q-1),ni+7)=gap;
          save indexedwavdct2.datindexedwavdct2 -ascii -double
          save indexedwavdct.dat indexedwavdct -ascii -double

        end
       end
     end
   end
 end
end



% This part is used to find the wavelet of images,  then cosine transform.
q=0;
fil='c';
le=1;
[rc]=edwavlet(ni,le,1,1,nb,fil,ns);       %This step is only for computation of rc
d=basis(rc);
dctd= d(1:rc,1:rc);
haard= d((rc+1):(2*rc),1:rc);
a=1;       % we don't need it
[rc]=trialm(ni,le,a,nb,fil,ns);
for hi=3:1:5            %the first coefficients =0 after removing average
  for mid=3:1:5
    for low=3:1:5
      for aa=1:4
        [fedcos,fedcosn]=sumedcoefficientcos(rc,dctd,haard,ni,hi,mid,low,ns,gs,fi,fil,a,aa)
        for v=1:3
          nc=4;
          q=q+1;
          %ddct=grouping(fd,ni,v,g);  %returns target vectors naming each image with
             %[n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fd,fdn,ni,v,nc);     %grouping such
           as statistical property is the same
         [n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fedcos,fedcosn,ni,v,nc,g);   %using
         vector quantization
         dd=[num2str(7) num2str(a) num2str(aa) num2str(hi) num2str(mid) num2str(low)];
         indexwavdct((2*q-1),1)= sscanf(dd,'%f');
         indexwavdct((2*q-1),2)=v;
         indexwavdct((2*q-1),3:(ni+2))=ddct(1,1:ni);
         indexwavdct((2*q-1),ni+5)=m1;
         indexwavdct((2*q-1),ni+6)=m2;
         % The  following function is to find the grouping of noisy images when
         % using nnet.
         % It doesn't take care of indexdct2
```

```
        %[xx,srgr]=nnetgroupingsumcoeff(fd,fdn,ddct,ni,v,nc,k);
        indexwavdct((2*q-1),ni+3)=srgr;
        indexwavdct((2*q-1),ni+4)=n1;
        indexwavdct((2*q-1),ni+7)=gap;
        indexwavdct((2*q),1)= sscanf(dd,'%f');
        indexwavdct((2*q),2)=v;
        indexwavdct((2*q),3:(ni+2))=xx(1,1:ni);
        indexwavdct2((2*q-1),1)= sscanf(dd,'%f');
        indexwavdct2((2*q-1),2)=v;
        indexwavdct2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
        indexwavdct2((2*q),1)= sscanf(dd,'%f');
        indexwavdct2((2*q),2)=v;
        indexwavdct2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
        indexwavdct2((2*q-1),ni+5)=m1;
        indexwavdct2((2*q-1),ni+6)=m2;
        indexwavdct2((2*q-1),ni+3)=srgr;
        indexwavdct2((2*q-1),ni+4)=n1;
        indexwavdct2((2*q-1),ni+7)=gap;
        save indexwavdct2.dat indexwavdct2 -ascii -double
        save indexwavdct.dat indexwavdct -ascii -double

      end
    end
  end
 end
end


% This part is used to find the singular values
for hi=5:5:10            %the first coefficients =0 after removing average
 for mid=5:5:10
  for low=5:5:10
    [fsvd,fsvdn]=sumcoefficientsvd(nb,ni,hi,mid,low,ns,gs,fi,'ced',1); %returns the files names
    for v=1:3
     q=q+1;
     nc=4;
        %[n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fsvd,fsvdn,ni,v,nc);  %returns target
        vectors naming each image with
     % group number
     %[d,jj,xf]=vecquantgroup2(fd,fdn,ni,v,nc);
         [n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fsvd,fsvdn,ni,v,nc,g);   %using vector
        quantization
     dd=[num2str(9) num2str(hi) num2str(mid) num2str(low)];
     indexsvd((2*q-1),1)= sscanf(dd,'%f');
     indexsvd((2*q-1),2)=v;
     indexsvd((2*q-1),3:(ni+2))=d(1,1:ni);
```

```
      indexsvd((2*q-1),ni+5)=m1;
      indexsvd((2*q-1),ni+6)=m2;
      %[xx,srgr]=nnetgroupingsumcoeff(fsvd,fsvdn,d,ni,v,nc,k);
      indexsvd((2*q-1),ni+3)=srgr;
      indexsvd((2*q-1),ni+4)=n1;
      indexsvd((2*q-1),ni+7)=gap;
      indexsvd((2*q),1)= sscanf(dd,'%f');
      indexsvd((2*q),2)=v;
      indexsvd((2*q),3:(ni+2))=xx(1,1:ni);
      indexsvd2((2*q-1),1)= sscanf(dd,'%f');
      indexsvd2((2*q-1),2)=v;
      indexsvd2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
      indexsvd2((2*q),1)= sscanf(dd,'%f');
      indexsvd2((2*q),2)=v;
      indexsvd2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
      indexsvd2((2*q-1),ni+5)=m1;
      indexsvd2((2*q-1),ni+6)=m2;
      indexsvd2((2*q-1),ni+3)=srgr;
      indexsvd2((2*q-1),ni+4)=n1;
      indexsvd2((2*q-1),ni+7)=gap;
      save indexsvd.dat indexsvd -ascii -double
      save indexsvd2.dat indexsvd2 -ascii -double
    end
   end
  end
end


% This part is used to compute the energy and mean and group images accrodingly
q=0;
[fenrgyav,fenrgyavn]=enrgyavext(nb,ns,ni,gs,fi,fil);
for v=1:2
  % v=1 energy, v=2 average
  q=q+1;
  nc=3;
  [n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fenrgyav,fenrgyavn,ni,v,nc);
  %[n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fenrgyav,fenrgyavn,ni,v,nc,g);   %using vector
quantization
  dd=[num2str(v) num2str(v) num2str(v) num2str(v)];
  indexenav((2*q-1),1)= sscanf(dd,'%f');       %watch how we enter a string into matrix
  indexenav((2*q-1),2)=v;
  indexenav((2*q-1),3:(ni+2))=d(1,1:ni);
  indexenav((2*q-1),ni+5)=m1;
  indexenav((2*q-1),ni+6)=m2;
  [xx,srgr]=nnetgroupingsumcoeff(fenrgyav,fenrgyavn,d,ni,v,nc,k);
  indexenav((2*q-1),ni+3)=srgr;
```

```
  indexenav((2*q-1),ni+4)=n1;
  indexenav((2*q-1),ni+7)=gap;
  indexenav((2*q),1)= sscanf(dd,'%f');
  indexenav((2*q),2)=v;
  indexenav((2*q),3:(ni+2))=xx(1,1:ni);
  indexenav2((2*q-1),1)= sscanf(dd,'%f');
  indexenav2((2*q-1),2)=v;
  indexenav2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
  indexenav2((2*q),1)= sscanf(dd,'%f');
  indexenav2((2*q),2)=v;
  indexenav2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
  indexenav2((2*q-1),ni+5)=m1;
  indexenav2((2*q-1),ni+6)=m2;
  indexenav2((2*q-1),ni+3)=srgr;
  indexenav2((2*q-1),ni+4)=n1;
  indexenav2((2*q-1),ni+7)=gap;
  save indexenav.dat indexenav -ascii -double
  save indexenav2.dat indexenav2 -ascii -double
end

% This part is used to compute the wavelet
q=0;
le=2;
for aa=1:3
  [fwavlet,fwavletn]=sumcoefficwavlet(ni,le,ns,aa,nb)
  q=q+1;
  nc=2;
  v=1;
  [n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fwavlet,fwavletn,ni,v,nc);
  %[n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fwavlet,fwavletn,ni,v,nc,g);   %using vector
quantization
  dd=[ num2str(aa) num2str(aa) num2str(0)];
  indexwav((2*q-1),1)= sscanf(dd,'%f');       %watch how we enter a string into matrix
  indexwav((2*q-1),2)=aa;
  indexwav((2*q-1),3:(ni+2))=d(1,1:ni);
  indexwav((2*q-1),ni+5)=m1;
  indexwav((2*q-1),ni+6)=m2;
  [xx,srgr]=nnetgroupingsumcoeff(fwavlet,fwavletn,d,ni,v,nc,k);
  indexwav((2*q-1),ni+3)=srgr;
  indexwav((2*q-1),ni+4)=n1;
  indexwav((2*q-1),ni+7)=gap;
  indexwav((2*q),1)= sscanf(dd,'%f');
  indexwav((2*q),2)=aa;
  indexwav((2*q),3:(ni+2))=xx(1,1:ni);
  indexwav2((2*q-1),1)= sscanf(dd,'%f');
  indexwav2((2*q-1),2)=aa;
```

```
  indexwav2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
  indexwav2((2*q),1)= sscanf(dd,'%f');
  indexwav2((2*q),2)=aa;
  indexwav2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
  indexwav2((2*q-1),ni+5)=m1;
  indexwav2((2*q-1),ni+6)=m2;
  indexwav2((2*q-1),ni+3)=srgr;
  indexwav2((2*q-1),ni+4)=n1;
  indexwav2((2*q-1),ni+7)=gap;
  save indexwav.dat indexwav -ascii -double
  save indexwav2.dat indexwav2 -ascii -double
end

% This part to comput max coeff of dct(haar), min,ratio and summation
% max=1, min=2, ratio=3, sum=4;
q=0;
[fmxmndct,fmxmndctn,fmxmnhaar,fmxmnhaarn]=mxmn(nb,ni,ns,gs,fi,fil);  %returns the files
names
for v=1:4
  nc=5;
  q=q+1;
  %ddct=grouping(fd,ni,v,g);  %returns target vectors naming each image with
  [n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fmxmndct,fmxmndctn,ni,v,nc);
%grouping such as statistical property is the same
  %[ddct,jj,xf]=tv(fd,fdn,ni,v,nc);
  %[n1,ddct,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fmxmndct,fmxmndctn,ni,v,nc,g);   %using
vector quantization
  dd=[num2str(7) num2str(0) num2str(0) num2str(v)];
  indexmxmndct((2*q-1),1)= sscanf(dd,'%f');
  indexmxmndct((2*q-1),2)=v;
  indexmxmndct((2*q-1),3:(ni+2))=ddct(1,1:ni);
  indexmxmndct((2*q-1),ni+5)=m1;
  indexmxmndct((2*q-1),ni+6)=m2;
  % The  following function is to find the grouping of noisy images when
  % using nnet.
  % It doesn't take care of indexdct2
  [xx,srgr]=nnetgroupingsumcoeff(fmxmndct,fmxmndctn,ddct,ni,v,nc,k);
  indexmxmndct((2*q-1),ni+3)=srgr;
  indexmxmndct((2*q-1),ni+4)=n1;
  indexmxmndct((2*q-1),ni+7)=gap;
  indexmxmndct((2*q),1)= sscanf(dd,'%f');
  indexmxmndct((2*q),2)=v;
  indexmxmndct((2*q),3:(ni+2))=xx(1,1:ni);
  indexmxmndct2((2*q-1),1)= sscanf(dd,'%f');
  indexmxmndct2((2*q-1),2)=v;
  indexmxmndct2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
```

```
  indexmxmndct2((2*q),1)= sscanf(dd,'%f');
  indexmxmndct2((2*q),2)=v;
  indexmxmndct2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
  indexmxmndct2((2*q-1),ni+5)=m1;
  indexmxmndct2((2*q-1),ni+6)=m2;
  indexmxmndct2((2*q-1),ni+3)=srgr;
  indexmxmndct2((2*q-1),ni+4)=n1;
  indexmxmndct2((2*q-1),ni+7)=gap;
  save indexmxmndct2.dat indexmxmndct2 -ascii -double
  save indexmxmndct.dat indexmxmndct -ascii -double

  %dhaar=grouping(fh,ni,v,g);                          % group number
  [n1,dhaar,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fmxmnhaar,fmxmnhaarn,ni,v,nc);
  %[dhaar,jj,xf]=tv(fd,fdn,ni,v,nc);
  %[n1,dhaar,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fmxmnhaar,fmxmnhaarn,ni,v,nc,g);
%using vector quantization
  dd=[num2str(8) num2str(0) num2str(0) num2str(v)];
  indexmxmnhaar((2*q-1),1)= sscanf(dd,'%f');
  indexmxmnhaar((2*q-1),2)=v;
  indexmxmnhaar((2*q-1),3:(ni+2))=dhaar(1,1:ni);
  indexmxmnhaar((2*q-1),ni+5)=m1;
  indexmxmnhaar((2*q-1),ni+6)=m2;
  % The  following function is to find the grouping of noisy images when
  % using nnet.
  % It doesn't take care of indexdct2
  [xx,srgr]=nnetgroupingsumcoeff(fmxmnhaar,fmxmnhaarn,dhaar,ni,v,nc,k);

  indexmxmnhaar((2*q-1),ni+3)=srgr;
  indexmxmnhaar((2*q-1),ni+4)=n1;
  indexmxmnhaar((2*q-1),ni+7)=gap;
  indexmxmnhaar((2*q),1)= sscanf(dd,'%f');
  indexmxmnhaar((2*q),2)=v;
  indexmxmnhaar((2*q),3:(ni+2))=xx(1,1:ni);
  indexmxmnhaar2((2*q-1),1)= sscanf(dd,'%f');
  indexmxmnhaar2((2*q-1),2)=v;
  indexmxmnhaar2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
  indexmxmnhaar2((2*q),1)= sscanf(dd,'%f');
  indexmxmnhaar2((2*q),2)=v;
  indexmxmnhaar2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
  indexmxmnhaar2((2*q-1),ni+5)=m1;
  indexmxmnhaar2((2*q-1),ni+6)=m2;
  indexmxmnhaar2((2*q-1),ni+3)=srgr;
  indexmxmnhaar2((2*q-1),ni+4)=n1;
  indexmxmnhaar2((2*q-1),ni+7)=gap;
  save indexmxmnhaar2.dat indexmxmnhaar2 -ascii -double
  save indexmxmnhaar.dat indexmxmnhaar -ascii -double
```

```
end

% This part is used to compute the edges of images, divide them into subimages, compute mean,
histogram and the standard deviation.
% We are using th four edge techniques

%note that we changed the directory for edges of images
q=0;
dtctedg(ni,ns,nb);
for dve=1:(log10(nb)/log10(2))-1;
  dv=2^dve;
  [fd,fdn]=edgestandevtion(nb,dv,ni,ns,gs,fi);
  for v=1:4
    % v=1 edge by canny v=2 edge by perwitt v=3 edge by zerocrossing v=4 edge by roberts
    q=q+1;
    nc=4;
    [n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fd,fdn,ni,v,nc);
    %[n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fd,fdn,ni,v,nc,g);   %using vector quantization
    dd=[num2str(1) num2str(v) num2str(v) num2str(v)];
    edgfor((2*q-1),1)= sscanf(dd,'%f');
    edgfor((2*q-1),2)=v;
    edgfor((2*q-1),3:(ni+2))=d(1,1:ni);
    edgfor((2*q-1),ni+5)=m1;
    edgfor((2*q-1),ni+6)=m2;
    [xx,srgr]=nnetgroupingsumcoeff(fd,fdn,d,ni,v,nc,k);
    edgfor((2*q-1),ni+3)=srgr;
    edgfor((2*q-1),ni+4)=n1;
    edgfor((2*q-1),ni+7)=gap;
    edgfor((2*q),1)= sscanf(dd,'%f');
    edgfor((2*q),2)=v;
    edgfor((2*q),3:(ni+2))=xx(1,1:ni);
    edgfor2((2*q-1),1)= sscanf(dd,'%f');
    edgfor2((2*q-1),2)=v;
    edgfor2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
    edgfor2((2*q),1)= sscanf(dd,'%f');
    edgfor2((2*q),2)=v;
    edgfor2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
    edgfor2((2*q-1),ni+5)=m1;
    edgfor2((2*q-1),ni+6)=m2;
    edgfor2((2*q-1),ni+3)=srgr;
    edgfor2((2*q-1),ni+4)=n1;
    edgfor2((2*q-1),ni+7)=gap;
    save edgfor.dat edgfor -ascii -double
    save edgfor2.dat edgfor2 -ascii -double
  end
end
```

```
q=0;
for hi=1:5:10              %the first coefficients =0 after removing the average
  for mid=5:5:10
    for low=5:5:10
      for a=1:4
        [fedsvd,fedsvdn]=sumedcoefficientsvd(nb,ni,hi,mid,low,ns,gs,fi,'ced',a) %returns the files
names
        for v=1:3
          q=q+1;
          nc=4;
            %[n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fedsvd,fedsvdn,ni,v,nc);  %returns
            target vectors naming each image with
          % group number
          %[d,jj,xf]=vecquantgroup2(fd,fdn,ni,v,nc);
             [n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fedsvd,fedsvdn,ni,v,nc,g);   %using vector
            quantization
          dd=[num2str(1) num2str(hi) num2str(mid) num2str(low)];
          indexedsvd((2*q-1),1)= sscanf(dd,'%f');
          indexedsvd((2*q-1),2)=v;
          indexedsvd((2*q-1),3:(ni+2))=d(1,1:ni);
          indexedsvd((2*q-1),ni+5)=m1;
          indexedsvd((2*q-1),ni+6)=m2;
          %[xx,srgr]=nnetgroupingsumcoeff(fedsvd,fedsvdn,d,ni,v,nc,k);
          indexedsvd((2*q-1),ni+3)=srgr;
          indexedsvd((2*q-1),ni+4)=n1;
          indexedsvd((2*q-1),ni+7)=gap;
          indexedsvd((2*q),1)= sscanf(dd,'%f');
          indexedsvd((2*q),2)=v;
          indexedsvd((2*q),3:(ni+2))=xx(1,1:ni);
          indexedsvd2((2*q-1),1)= sscanf(dd,'%f');
          indexedsvd2((2*q-1),2)=v;
          indexedsvd2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
          indexedsvd2((2*q),1)= sscanf(dd,'%f');
          indexedsvd2((2*q),2)=v;
          indexedsvd2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
          indexedsvd2((2*q-1),ni+5)=m1;
          indexedsvd2((2*q-1),ni+6)=m2;
          indexedsvd2((2*q-1),ni+3)=srgr;
          indexedsvd2((2*q-1),ni+4)=n1;
          indexedsvd2((2*q-1),ni+7)=gap;
          save indexedsvd.dat indexedsvd -ascii -double
          save indexedsvd2.dat indexedsvd2 -ascii -double
        end
      end
    end
  end
```

```
end

q=0;
for aaa=1:4
  for aa=1:3
    [fwavleted,fwavletedn]=sumcoeffiedwavlet(ni,2,aa,aaa,nb)  %returns the files names
    q=q+1;
    nc=2;
    v=1;
    [n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=groupstat(fwavleted,fwavletedn,ni,v,nc);  %returns
target vectors naming each image with
    % group number
    %[d,jj,xf]=vecquantgroup2(fd,fdn,ni,v,nc);
    %[n1,d,srgr,xf,m1,m2,xx,gap,bot,upp]=vcquant(fwavleted,fwavletedn,ni,v,nc,g);   %using
vector quantization
    dd=[num2str(aaa) num2str(aa) num2str(0) num2str(0)];
    indexedwav((2*q-1),1)= sscanf(dd,'%f');
    indexedwav((2*q-1),2)=aa;
    indexedwav((2*q-1),3:(ni+2))=d(1,1:ni);
    indexedwav((2*q-1),ni+5)=m1;
    indexedwav((2*q-1),ni+6)=m2;
    [xx,srgr]=nnetgroupingsumcoeff(fwavleted,fwavletedn,d,ni,v,nc,k);
    indexedwav((2*q-1),ni+3)=srgr;
    indexedwav((2*q-1),ni+4)=n1;
    indexedwav((2*q-1),ni+7)=gap;
    indexedwav((2*q),1)= sscanf(dd,'%f');
    indexedwav((2*q),2)=aa;
    indexedwav((2*q),3:(ni+2))=xx(1,1:ni);
    indexedwav2((2*q-1),1)= sscanf(dd,'%f');
    indexedwav2((2*q-1),2)=aaa;
    indexedwav2((2*q-1),3:(ni+2))=xf(1,1:ni);   %image number
    indexedwav2((2*q),1)= sscanf(dd,'%f');
    indexedwav2((2*q),2)=aaa;
    indexedwav2((2*q),3:(ni+2))=xf(2,1:ni);    % group number
    indexedwav2((2*q-1),ni+5)=m1;
    indexedwav2((2*q-1),ni+6)=m2;
    indexedwav2((2*q-1),ni+3)=srgr;
    indexedwav2((2*q-1),ni+4)=n1;
    indexedwav2((2*q-1),ni+7)=gap;
    save indexedwav.dat indexedwav -ascii -double
    save indexedwav2.dat indexedwav2 -ascii -double
  end
end
```

```
function []=votemg(ni,ss,ng)
% This function is used to find the final decision for grouping of images
% from the majority decision of all criteria doing the same grouping.
% When using crtfn2, we vote according to partial accuracy, i.e the images wanted
% from the criteria at certain stage
% We have two options either full accuracy (ni+3) or partial crtfn(i,3*j)
% This function is used when we have more than 2 groups
% ng is the number of groups
% We are using the function maximum, if 2 groups have equal votes will take the first 1
load imagind.dat
load crtfn2.dat
load indexdct.dat
load indexhaar.dat
load indexmxmndct.dat
load indexmxmnhaar.dat
load indexsvd.dat
load indexedsvd.dat
load indexenav.dat
load edgfor.dat
load indexwav.dat
load indexedwav.dat


[ri,ci]=size(imagind)
[rc,cc]=size(crtfn2)
[rd,cd]=size(indexdct);
[rh,ch]=size(indexhaar);
[rdm,cdm]=size(indexmxmndct);
[rhm,chm]=size(indexmxmnhaar);
[rs,cs]=size(indexsvd);
[res,ces]=size(indexsvd);
[re,ce]=size(indexenav);
[ref,cef]=size(edgfor);
[rw,cw]=size(indexwav);
[rwe,cwe]=size(indexedwav);

ss=ri;
nos=zeros((1+ng)*ss,ci);
% This matrix is similar to imagind, for each noisy image, corresponding to grouping
% of imagind we put 1 or 2 according to the majority votes.
% For each row, we follow it by 2 rows showing how many votes 1 and 2 respectively.

% We make a cost function, the coefficient equal the number of total images
% grouped correctly by the criteria
for i=2:ri-1
```

```
for jj=3:ci
   if imagind(i,jj)~=0
      g=zeros(1,ng);
      for j=1:cc/3
         for k=1:ng
            if crtfn2(i,(3*j-2))~=0
               for ii=1:rd/2
                  if (indexdct((2*ii-1),1)==crtfn2(i,(3*j-2))
                        & indexdct((2*ii-1),2)==crtfn2(i,(3*j-1)))
                     if indexdct(2*ii,jj)==k
                        %g1=g1+(indexdct((2*ii-1),(ni+3)));
                        g(1,k)=g(1,k)+crtfn2(i,(3*j));
                        if (indexdct((2*ii-1),(ni+3)))==ni
                           g(1,k)=1000;
                        end
                     end
                  end
               end

               for ii=1:rh/2
                  if (indexhaar((2*ii-1),1)==crtfn2(i,(3*j-2))
                        & indexhaar((2*ii-1),2)==crtfn2(i,(3*j-1)))
                     if indexhaar(2*ii,jj)==k
                        %g1=g1+(indexhaar((2*ii-1),(ni+3)));
                        g(1,k)=g(1,k)+crtfn2(i,(3*j));
                        if (indexhaar((2*ii-1),(ni+3)))==ni
                           g(1,k)=1000;
                        end

                     end
                  end
               end

               for ii=1:rdm/2
                  if (indexmxmndct((2*ii-1),1)==crtfn2(i,(3*j-2))
                & indexmxmndct((2*ii-1),2)==crtfn2(i,(3*j-1)))
                     if indexmxmndct(2*ii,jj)==k
                        %g1=g1+(indexmxmndct((2*ii-1),(ni+3)));
                        g(1,k)=g(1,k)+crtfn2(i,(3*j));
                        if (indexmxmndct((2*ii-1),(ni+3)))==ni
                           g(1,k)=1000;
                        end
                     end
                  end
               end
```

```
for ii=1:rhm/2
   if (indexmxmnhaar((2*ii-1),1)==crtfn2(i,(3*j-2))
 & indexmxmnhaar((2*ii-1),2)==crtfn2(i,(3*j-1)))
      if indexmxmnhaar(2*ii,jj)==k
         %g1=g1+(indexmxmnhaar((2*ii-1),(ni+3)));
         g(1,k)=g(1,k)+crtfn2(i,(3*j));
         if (indexmxmnhaar((2*ii-1),(ni+3)))==ni
            g(1,k)=1000;
         end
      end
   end
end

for ii=1:re/2
   if (indexenav((2*ii-1),1)==crtfn2(i,(3*j-2))
 & indexenav((2*ii-1),2)==crtfn2(i,(3*j-1)))
      if indexenav(2*ii,jj)==k
         %g1=g1+(indexenav((2*ii-1),(ni+3)));
         g(1,k)=g(1,k)+crtfn2(i,(3*j));
         if (indexenav((2*ii-1),(ni+3)))==ni
            g(1,k)=1000;
         end
      end
   end
end

for ii=1:ref/2
   if (edgfor((2*ii-1),1)==crtfn2(i,(3*j-2)) & edgfor((2*ii-1),2)==crtfn2(i,(3*j-1)))
      if edgfor(2*ii,jj)==k
         %g1=g1+(indexhaar((2*ii-1),(ni+3)));
         g(1,k)=g(1,k)+crtfn2(i,(3*j));
         if (edgfor((2*ii-1),(ni+3)))==ni
            g(1,k)=1000;
         end
      end
   end
end

 for ii=1:res/2
   if (indexedsvd((2*ii-1),1)==crtfn2(i,(3*j-2))
& indexedsvd((2*ii-1),2)==crtfn2(i,(3*j-1)))
      if indexedsvd(2*ii,jj)==k
         %g1=g1+(indexhaar((2*ii-1),(ni+3)));
         g(1,k)=g(1,k)+crtfn2(i,(3*j));
         if (indexedsvd((2*ii-1),(ni+3)))==ni
            g(1,k)=1000;
```

```
            end
          end
        end
      end

      for ii=1:rw/2
        if (indexwav((2*ii-1),1)==crtfn2(i,(3*j-2))
            & indexwav((2*ii-1),2)==crtfn2(i,(3*j-1)))
          if indexwav(2*ii,jj)==k
            %g1=g1+(indexhaar((2*ii-1),(ni+3)));
            g(1,k)=g(1,k)+crtfn2(i,(3*j));
            if (indexwav((2*ii-1),(ni+3)))==ni
              g(1,k)=1000;
            end
          end
        end
      end

      for ii=1:rwe/2
        if (indexedwav((2*ii-1),1)==crtfn2(i,(3*j-2))
            & indexedwav((2*ii-1),2)==crtfn2(i,(3*j-1)))
          if indexedwav(2*ii,jj)==k
            %g1=g1+(indexhaar((2*ii-1),(ni+3)));
            g(1,k)=g(1,k)+crtfn2(i,(3*j));
            if (indexedwav((2*ii-1),(ni+3)))==ni
              g(1,k)=1000;
            end
          end
        end
      end


      for ii=1:rs/2
        if (indexsvd((2*ii-1),1)==crtfn2(i,(3*j-2))
            & indexsvd((2*ii-1),2)==crtfn2(i,(3*j-1)))
          if indexsvd(2*ii,jj)==k
            %g1=g1+(indexsvd((2*ii-1),(ni+3)));
            g(1,k)=g(1,k)+crtfn2(i,(3*j));
            if (indexsvd((2*ii-1),(ni+3)))==ni
              g(1,k)=1000;
            end
          end
        end
      end
    end
  end
```

```
            nos(((ng+1)*i-ng),1)=555;
            nos(((ng+1)*i-ng),2)=555;            % to show that this row is grouping
            [m,I]=max(g);                        % Take care if there is a tie, take
            % the first one
            for u=1:ng
               nos(((ng+1)*i-ng),jj)=I;
               nos(((ng+1)*i-(ng-u)),jj)=g(1,u);
            end
            save nos.dat nos -ascii -double
         end
      end
   end
end
save nos.dat nos -ascii -double

% This part is to show the error, the different between imagind and nos
dif=zeros(ss,ni);
dif(1,:)=imagind(1,3:(ni+2));
for i=2:ss
   for j=3:(ni+2)
      dif(i,(j-2))=imagind(i,j)-nos(((ng+1)*i-ng),j);
   end
end
save dif.dat dif -ascii -double

% This function is used to put in crtfn.dat the number of correct images
% with respect to a certrain criteria for certain number of images.
function []=votepart(ni)
% ni : number of images

load imagind.dat
load crtfn.dat
[ri,ci]=size(imagind);
[rc,cc]=size(crtfn);
crtfn2=zeros(rc,cc);


for i=2:rc
  for j=1:cc/3
    c=0;
    v=zeros(2,ni);
    for jj=3:(ni+2)

      if imagind(i,jj)~=0;
        c=c+1;
```

```
      v(1,c)=imagind(1,jj);
      v(2,c)=imagind(i,jj);
     end
    end
    cr=crtfn(i,((3*j)-2));
    bn=crtfn(i,((3*j)-1));
    [a,nc]=grpcrt(cr,bn,ni,v);
    crtfn2(i,((3*j)-2))=cr;
    crtfn2(i,((3*j)-1))=bn;
    crtfn2(i,(3*j))=nc;
  end
end
save crtfn2.dat crtfn2 -ascii –double



function [wr,vtn]=votesgim(ni,ng,im)
% im is the image required
% nv is the number of votes
% This function is used to find the final decision for single image
% from the majority decision of all criteria doing the same grouping and selecting
% the optimum number of votes.
% When using crtfn2, we vote according to partial accuracy, i.e the images wanted
% from the criteria at certain stage
% We have two options either full accuracy (ni+3) or partial crtfn(i,3*j)
% This function is used when we have more than 2 groups
% ng is the number of groups
% We are using the function maximum, if 2 groups have equal votes will take the first 1

load imagind.dat
load crtfn2.dat
load indexdct.dat
load indexhaar.dat
load indexmxmndct.dat
load indexmxmnhaar.dat
load indexsvd.dat
load indexedsvd.dat
load indexenav.dat
load edgfor.dat
load indexwav.dat
load indexedwav.dat

[ri,ci]=size(imagind)
[rc,cc]=size(crtfn2)
[rd,cd]=size(indexdct);
[rh,ch]=size(indexhaar);
```

```
[rdm,cdm]=size(indexmxmndct);
[rhm,chm]=size(indexmxmnhaar);
[rs,cs]=size(indexsvd);
[res,ces]=size(indexsvd);
[re,ce]=size(indexenav);
[ref,cef]=size(edgfor);
[rw,cw]=size(indexwav);
[rwe,cwe]=size(indexedwav);

ss=ri;
nosim=zeros((1+ng)*ss,1);
% This matrix is similar to imagind, for each noisy image, corresponding to grouping
% of imagind we put 1 or 2 according to the majority votes.
% For each row, we follow it by 2 rows showing how many votes 1 and 2 respectively.

% We make a cost function, the coefficient equal the number of total images
% grouped correctly by the criteria
dec=zeros(cc/3,2);   % matrix (number of votes, number of wrong decisions)

for t=1:cc/3
 for i=2:ri-1
   jj=im+2;
   if imagind(i,jj)~=0
     g=zeros(1,ng);
     for j=1:t
       for k=1:ng
         if crtfn2(i,(3*j-2))~=0
           for ii=1:rd/2
             if (indexdct((2*ii-1),1)==crtfn2(i,(3*j-2)) & indexdct((2*ii-1),2)==crtfn2(i,(3*j-1)))
               if indexdct(2*ii,jj)==k
                 %g1=g1+(indexdct((2*ii-1),(ni+3)));
                 g(1,k)=g(1,k)+crtfn2(i,(3*j));
                 if (indexdct((2*ii-1),(ni+3)))==ni
                   g(1,k)=1000;
                 end
               end
             end
           end

           for ii=1:rh/2
             if (indexhaar((2*ii-1),1)==crtfn2(i,(3*j-2)) & indexhaar((2*ii-1),2)==crtfn2(i,(3*j-1)))
               if indexhaar(2*ii,jj)==k
                 %g1=g1+(indexhaar((2*ii-1),(ni+3)));
                 g(1,k)=g(1,k)+crtfn2(i,(3*j));
                 if (indexhaar((2*ii-1),(ni+3)))==ni
                   g(1,k)=1000;
```

```
      end

    end
  end
end

for ii=1:rdm/2
          if (indexmxmndct((2*ii-1),1)==crtfn2(i,(3*j-2)) & indexmxmndct((2*ii-
         1),2)==crtfn2(i,(3*j-1)))
    if indexmxmndct(2*ii,jj)==k
      %g1=g1+(indexmxmndct((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (indexmxmndct((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
end

for ii=1:rhm/2
  if (indexmxmnhaar((2*ii-1),1)==crtfn2(i,(3*j-2))
 & indexmxmnhaar((2*ii-1),2)==crtfn2(i,(3*j-1)))
    if indexmxmnhaar(2*ii,jj)==k
      %g1=g1+(indexmxmnhaar((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (indexmxmnhaar((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
end

for ii=1:re/2
  if (indexenav((2*ii-1),1)==crtfn2(i,(3*j-2)) & indexenav((2*ii-1),2)==crtfn2(i,(3*j-1)))
    if indexenav(2*ii,jj)==k
      %g1=g1+(indexenav((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (indexenav((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
end

for ii=1:ref/2
  if (edgfor((2*ii-1),1)==crtfn2(i,(3*j-2)) & edgfor((2*ii-1),2)==crtfn2(i,(3*j-1)))
```

```
    if edgfor(2*ii,jj)==k
      %g1=g1+(indexhaar((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (edgfor((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
end

for ii=1:res/2
  if (indexedsvd((2*ii-1),1)==crtfn2(i,(3*j-2))
& indexedsvd((2*ii-1),2)==crtfn2(i,(3*j-1)))
    if indexedsvd(2*ii,jj)==k
      %g1=g1+(indexhaar((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (indexedsvd((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
end

for ii=1:rw/2
  if (indexwav((2*ii-1),1)==crtfn2(i,(3*j-2)) & indexwav((2*ii-1),2)==crtfn2(i,(3*j-1)))
    if indexwav(2*ii,jj)==k
      %g1=g1+(indexhaar((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (indexwav((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
end

for ii=1:rwe/2
  if (indexedwav((2*ii-1),1)==crtfn2(i,(3*j-2))
& indexedwav((2*ii-1),2)==crtfn2(i,(3*j-1)))
    if indexedwav(2*ii,jj)==k
      %g1=g1+(indexhaar((2*ii-1),(ni+3)));
      g(1,k)=g(1,k)+crtfn2(i,(3*j));
      if (indexedwav((2*ii-1),(ni+3)))==ni
        g(1,k)=1000;
      end
    end
  end
```

```
        end


        for ii=1:rs/2
          if (indexsvd((2*ii-1),1)==crtfn2(i,(3*j-2)) & indexsvd((2*ii-1),2)==crtfn2(i,(3*j-1)))
            if indexsvd(2*ii,jj)==k
              %g1=g1+(indexsvd((2*ii-1),(ni+3)));
              g(1,k)=g(1,k)+crtfn2(i,(3*j));
              if (indexsvd((2*ii-1),(ni+3)))==ni
                g(1,k)=1000;
              end
            end
          end
        end
      end
      g
      nosim(((ng+1)*i-ng),1)=555;
      % to show that this row is grouping
      [m,I]=max(g);                    % Take care if there is a tie, take
      % the first one
      for u=1:ng
        nosim(((ng+1)*i-ng),1)=I;
        nosim(((ng+1)*i-(ng-u)),1)=g(1,u);
      end
      save nosim.dat nosim -ascii -double
    end
  end
end
save nosim.dat nosim -ascii -double

% This part is to show the error, the different between imagind and nos
acc=zeros(ss,1);
for i=2:ss
   acc(i,1)=imagind(i,(im+2))-nosim(((ng+1)*i-ng),1);
end
acc(1,1)=im;
acc(ss+1,1)=t;
py=0;   % the number of wrong stages grouped wrongly.
for i=2:ss
  if acc(i,1)~=0
    py=py+1;
  end
end
dec(t,1)=t;
dec(t,2)=py;
```

```
  [k,m]=min(dec);
  wr=k(1,2);       % number of wrong decisions.
  vtn=m(1,2);       % number of votes
end
save dec.dat dec -ascii –double



% This function receives images grouped into ng groups (y)
% It scan all criteria to find out which one gives best result in case of noisy
function [temp,in,ni]=diffcritmg(y1,nii,ng)
%ng is the number of groups
% nii is the original number of images, when we remove zeros, it becomes ni
% arrang images group one then group 2 and remove zeros

load indexdct2.dat          % images of group one, then group 2
% image number in first row
load indexhaar2.dat
load indexsvd2.dat
load indexedsvd2.dat
load indexenav2.dat
load indexmxmndct2.dat          % images of group one, then group 2
load indexmxmnhaar2.dat
load edgfor2.dat
load indexedwav2.dat
load indexwav2.dat

[rd,cd]=size(indexdct2);      % they are obtained from system2
[rh,ch]=size(indexhaar2);
[rdm,cdm]=size(indexmxmndct2);      % they are obtained from system2
[rhm,chm]=size(indexmxmnhaar2);
[rs,cs]=size(indexsvd2);
[res,ces]=size(indexedsvd2);
[re,ce]=size(indexenav2);
[ref,ce]=size(edgfor2);
[rw,cw]=size(indexwav2);
[rwe,cwe]=size(indexedwav2);

yy(1:2,1:2)=y1(1:2,1:2);

j=2;
kg=0;
in=zeros(1,ng);       %vector to compute the number of elements in each group
for u=1:ng
  for jj=3:(nii+2)
    if y1(2,jj)==u
```

```
      j=j+1;
      yy(1,j)=y1(1,jj);
      yy(2,j)=u;
    end
  end
  in(1,u)=j-2-kg;
  kg=kg+in(1,u);
end

ni=j-2;
inn=0;
for t=1:ng
  if (in(1,t)~=1 & in(1,t)~=0)
    inn=inn+1;
  end
end

in
[m,I]=max(in);
if (inn==0 & ni==1)
  %temp1=zeros(3,(in+4));
  %temp2=zeros(3,(ni-in+4));
  temp=zeros((3*ng),(m+4));
else

  %temp1 matrix is 3 rows matrix.
  %first row is the images number of group one.
  %second row is ones.
  %third row is the grouping of this group according to criterion in first 2 cells
  %ni=number of images resulting from the previous criterion ( They have 1 and 2)

  temp=zeros((3*ng),(m+4));

  lg=0;
  for k=1:ng
    temp((3*k-2):(3*k-1),1:2)=yy(1:2,1:2);
    temp((3*k-2):(3*k-1),3:(in(1,k)+2))=yy(1:2,((lg+3):(lg+in(1,k)+2)));
    lg=lg+in(1,k);

    % we choose the criterion that can extract images at each stage(small number of\
    % images in group 1 or 2)
    temp11=zeros(2,in(1,k)+4);
    temp(3*k,(in(1,k)+3))=nii;
    for i=1:rd/2
      l1=0;
      l2=0;
```

```
o=indexdct2((2*i-1),1);
dd=[ num2str(o)];
temp11(1,1)=sscanf(dd,'%f');
temp11(1,2)=indexdct2((2*i-1),2);
temp11(2,1)=sscanf(dd,'%f');
temp11(2,2)=indexdct2((2*i-1),2);

for j=3:in(1,k)+2
  for jj=3:nii+2
    if temp(3*k-2,j)==indexdct2((2*i-1),jj)
      temp11(1,j)=temp(3*k-2,j);
      temp11(2,j)=indexdct2(2*i,jj);
    end
  end
end
l=zeros(1,ng);          %number of elements in each group
for kk=3:(in(1,k)+2)
  for kkk=1:ng
    if temp11(2,kk)==kkk
      l(1,kkk)=l(1,kkk)+1;
    end
  end
end
oo=0;
for o=1:ng
  if l(1,o)~=0
    oo=oo+1;
  end
end

mx=max(l);
mn=min(l);
difr=(mx-mn);
% this function is to find the number of correct noisy images.
vc=zeros(1,in(1,k));
vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
cr=temp11(1,1);
bn=temp11(1,2);
[a,nc]=grpcrt(cr,bn,nii,vc);
temp11(2,(in(1,k)+4))=nc;
if (oo>=2)

  if temp(3*k,(in(1,k)+4))<nc
    if (temp(3*k,(in(1,k)+3))>difr)
      cc=0;
      temp11(2,(in(1,k)+3))=difr;
```

```
      %for y=1:ng
      % if (l(1,y)==0)
      %  cc=cc+1;
      %end
      %end

      temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
      save temp.dat temp -ascii -double

    end
   end
  end
end


% do the same thing for haar
for i=1:rh/2
  l1=0;
  l2=0;
  o=indexhaar2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=indexhaar2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=indexhaar2((2*i-1),2);

  for j=3:in(1,k)+2
    for jj=3:nii+2
      if temp(3*k-2,j)==indexhaar2((2*i-1),jj)
        temp11(1,j)=temp(3*k-2,j);
        temp11(2,j)=indexhaar2(2*i,jj);
      end
    end
  end
  l=zeros(1,ng);
  for kk=3:(in(1,k)+2)
    for kkk=1:ng
      if temp11(2,kk)==kkk
        l(1,kkk)=l(1,kkk)+1;
      end
    end
  end
  oo=0;
  for o=1:ng
    if l(1,o)~=0
      oo=oo+1;
```

```
    end
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
  bn=temp11(1,2);
  [a,nc]=grpcrt(cr,bn,nii,vc);
  temp11(2,(in(1,k)+4))=nc;
  if (mn~=0 |oo~=0)
    if temp(3*k,(in(1,k)+4))<nc
      if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
        temp11(2,(in(1,k)+3))=difr;
        %for y=1:ng
        % if (l(1,y)==0)
        %  cc=cc+1;
        %end
        %end

        temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
        save temp.dat temp -ascii -double

      end
    end
  end
end

for i=1:rdm/2
  l1=0;
  l2=0;
  o=indexmxmndct2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=indexmxmndct2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=indexmxmndct2((2*i-1),2);

  for j=3:in(1,k)+2
    for jj=3:nii+2
      if temp(3*k-2,j)==indexmxmndct2((2*i-1),jj)
        temp11(1,j)=temp(3*k-2,j);
        temp11(2,j)=indexmxmndct2(2*i,jj);
```

```
      end
    end
  end
  l=zeros(1,ng);
  for kk=3:(in(1,k)+2)
    for kkk=1:ng
      if temp11(2,kk)==kkk
        l(1,kkk)=l(1,kkk)+1;
      end
    end
  end
  oo=0;
  for o=1:ng
    if l(1,o)~=0
      oo=oo+1;
    end
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
  bn=temp11(1,2);
  [a,nc]=grpcrt(cr,bn,nii,vc);
  temp11(2,(in(1,k)+4))=nc;
  if (mn~=0 |oo~=0)
    if temp(3*k,(in(1,k)+4))<nc
      if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
        temp11(2,(in(1,k)+3))=difr;
        %for y=1:ng
        % if (l(1,y)==0)
        %  cc=cc+1;
        %end
        %end

        temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
        save temp.dat temp -ascii -double

      end
    end
  end
end
```

```
for i=1:rhm/2
  l1=0;
  l2=0;
  o=indexmxmnhaar2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=indexmxmnhaar2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=indexmxmnhaar2((2*i-1),2);

  for j=3:in(1,k)+2
    for jj=3:nii+2
      if temp(3*k-2,j)==indexmxmnhaar2((2*i-1),jj)
        temp11(1,j)=temp(3*k-2,j);
        temp11(2,j)=indexmxmnhaar2(2*i,jj);
      end
    end
  end
  l=zeros(1,ng);
  for kk=3:(in(1,k)+2)
    for kkk=1:ng
      if temp11(2,kk)==kkk
        l(1,kkk)=l(1,kkk)+1;
      end
    end
  end
  oo=0;
  for o=1:ng
    if l(1,o)~=0
      oo=oo+1;
    end
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
  bn=temp11(1,2);
  [a,nc]=grpcrt(cr,bn,nii,vc);
  temp11(2,(in(1,k)+4))=nc;
  if (mn~=0 |oo~=0)
    if temp(3*k,(in(1,k)+4))<nc
      if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
```

```
      temp11(2,(in(1,k)+3))=difr;
      %for y=1:ng
      % if (l(1,y)==0)
      %  cc=cc+1;
      %end
      %end

      temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
      save temp.dat temp -ascii -double

    end
   end
  end
end
% for svd
for i=1:rs/2
 l1=0;
 l2=0;
 o=indexsvd2((2*i-1),1);
 dd=[ num2str(o)];
 temp11(1,1)=sscanf(dd,'%f');
 temp11(1,2)=indexsvd2((2*i-1),2);
 temp11(2,1)=sscanf(dd,'%f');
 temp11(2,2)=indexsvd2((2*i-1),2);

 for j=3:in(1,k)+2
  for jj=3:nii+2
   if temp(3*k-2,j)==indexsvd2((2*i-1),jj)
    temp11(1,j)=temp(3*k-2,j);
    temp11(2,j)=indexsvd2(2*i,jj);
   end
  end
 end
 l=zeros(1,ng);
 for kk=3:(in(1,k)+2)
  for kkk=1:ng
   if temp11(2,kk)==kkk
    l(1,kkk)=l(1,kkk)+1;
   end
  end
 end
 oo=0;
 for o=1:ng
  if l(1,o)~=0
   oo=oo+1;
  end
```

```
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
  bn=temp11(1,2);
  [a,nc]=grpcrt(cr,bn,nii,vc);
  temp11(2,(in(1,k)+4))=nc;
  if (mn~=0 |oo~=0)
    if temp(3*k,(in(1,k)+4))<nc
      if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
        temp11(2,(in(1,k)+3))=difr;
        %for y=1:ng
        % if (l(1,y)==0)
        %  cc=cc+1;
        %end
        %end

        temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
        save temp.dat temp -ascii -double

      end
    end
  end
end

% for svd
for i=1:rw/2
  l1=0;
  l2=0;
  o=indexwav2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=indexwav2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=indexwav2((2*i-1),2);

  for j=3:in(1,k)+2
    for jj=3:nii+2
      if temp(3*k-2,j)==indexwav2((2*i-1),jj)
        temp11(1,j)=temp(3*k-2,j);
        temp11(2,j)=indexwav2(2*i,jj);
```

```
    end
   end
  end
  l=zeros(1,ng);
  for kk=3:(in(1,k)+2)
   for kkk=1:ng
    if temp11(2,kk)==kkk
     l(1,kkk)=l(1,kkk)+1;
    end
   end
  end
  oo=0;
  for o=1:ng
   if l(1,o)~=0
    oo=oo+1;
   end
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
  bn=temp11(1,2);
  [a,nc]=grpcrt(cr,bn,nii,vc);
  temp11(2,(in(1,k)+4))=nc;
  if (mn~=0 |oo~=0)
   if temp(3*k,(in(1,k)+4))<nc
    if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
     temp11(2,(in(1,k)+3))=difr;
     %for y=1:ng
     % if (l(1,y)==0)
     %  cc=cc+1;
     %end
     %end

     temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
     save temp.dat temp -ascii -double

    end
   end
  end
end
```

```
% for svd
for i=1:rwe/2
 l1=0;
 l2=0;
 o=indexedwav2((2*i-1),1);
 dd=[ num2str(o)];
 temp11(1,1)=sscanf(dd,'%f');
 temp11(1,2)=indexedwav2((2*i-1),2);
 temp11(2,1)=sscanf(dd,'%f');
 temp11(2,2)=indexedwav2((2*i-1),2);

 for j=3:in(1,k)+2
  for jj=3:nii+2
   if temp(3*k-2,j)==indexedwav2((2*i-1),jj)
    temp11(1,j)=temp(3*k-2,j);
    temp11(2,j)=indexedwav2(2*i,jj);
   end
  end
 end
 l=zeros(1,ng);
 for kk=3:(in(1,k)+2)
  for kkk=1:ng
   if temp11(2,kk)==kkk
    l(1,kkk)=l(1,kkk)+1;
   end
  end
 end
 oo=0;
 for o=1:ng
  if l(1,o)~=0
   oo=oo+1;
  end
 end
 mx=max(l);
 mn=min(l);
 difr=(mx-mn);

 % this function is to find the number of correct noisy images.
 vc=zeros(1,in(1,k));
 vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
 cr=temp11(1,1);
 bn=temp11(1,2);
 [a,nc]=grpcrt(cr,bn,nii,vc);
 temp11(2,(in(1,k)+4))=nc;
 if (mn~=0 |oo~=0)
```

```
   if temp(3*k,(in(1,k)+4))<nc
    if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
      temp11(2,(in(1,k)+3))=difr;
      %for y=1:ng
      % if (l(1,y)==0)
      %  cc=cc+1;
      %end
      %end

      temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
      save temp.dat temp -ascii -double

    end
   end
  end
end


for i=1:res/2
  l1=0;
  l2=0;
  o=indexedsvd2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=indexedsvd2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=indexedsvd2((2*i-1),2);

  for j=3:in(1,k)+2
    for jj=3:nii+2
      if temp(3*k-2,j)==indexedsvd2((2*i-1),jj)
        temp11(1,j)=temp(3*k-2,j);
        temp11(2,j)=indexedsvd2(2*i,jj);
      end
    end
  end
  l=zeros(1,ng);
  for kk=3:(in(1,k)+2)
    for kkk=1:ng
      if temp11(2,kk)==kkk
        l(1,kkk)=l(1,kkk)+1;
      end
    end
  end
  oo=0;
  for o=1:ng
```

```
   if l(1,o)~=0
     oo=oo+1;
   end
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
  bn=temp11(1,2);
  [a,nc]=grpcrt(cr,bn,nii,vc);
  temp11(2,(in(1,k)+4))=nc;
  if (mn~=0 |oo~=0)
    if temp(3*k,(in(1,k)+4))<nc
      if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
        temp11(2,(in(1,k)+3))=difr;
        %for y=1:ng
        % if (l(1,y)==0)
        %  cc=cc+1;
        %end
        %end

        temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
        save temp.dat temp -ascii -double

      end
    end
  end
end


% do the same thing for haar
for i=1:ref/2
  l1=0;
  l2=0;
  o=edgfor2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=edgfor2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=edgfor2((2*i-1),2);

  for j=3:in(1,k)+2
```

```
  for jj=3:nii+2
    if temp(3*k-2,j)==edgfor2((2*i-1),jj)
      temp11(1,j)=temp(3*k-2,j);
      temp11(2,j)=edgfor2(2*i,jj);
    end
  end
end
l=zeros(1,ng);
for kk=3:(in(1,k)+2)
  for kkk=1:ng
    if temp11(2,kk)==kkk
      l(1,kkk)=l(1,kkk)+1;
    end
  end
end
oo=0;
for o=1:ng
  if l(1,o)~=0
    oo=oo+1;
  end
end
mx=max(l);
mn=min(l);
difr=(mx-mn);

% this function is to find the number of correct noisy images.
vc=zeros(1,in(1,k));
vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
cr=temp11(1,1);
bn=temp11(1,2);
[a,nc]=grpcrt(cr,bn,nii,vc);
temp11(2,(in(1,k)+4))=nc;
if (mn~=0 |oo~=0)
  if temp(3*k,(in(1,k)+4))<nc
    if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
      temp11(2,(in(1,k)+3))=difr;
      %for y=1:ng
      % if (l(1,y)==0)
      %  cc=cc+1;
      %end
      %end

      temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
      save temp.dat temp -ascii -double

    end
```

```
    end
  end
end



for i=1:re/2
  l1=0;
  l2=0;
  o=indexenav2((2*i-1),1);
  dd=[ num2str(o)];
  temp11(1,1)=sscanf(dd,'%f');
  temp11(1,2)=indexenav2((2*i-1),2);
  temp11(2,1)=sscanf(dd,'%f');
  temp11(2,2)=indexenav2((2*i-1),2);

  for j=3:in(1,k)+2
    for jj=3:nii+2
      if temp(3*k-2,j)==indexenav2((2*i-1),jj)
        temp11(1,j)=temp(3*k-2,j);
        temp11(2,j)=indexenav2(2*i,jj);
      end
    end
  end
  l=zeros(1,ng);
  for kk=3:(in(1,k)+2)
    for kkk=1:ng
      if temp11(2,kk)==kkk
        l(1,kkk)=l(1,kkk)+1;
      end
    end
  end
  oo=0;
  for o=1:ng
    if l(1,o)~=0
      oo=oo+1;
    end
  end
  mx=max(l);
  mn=min(l);
  difr=(mx-mn);

  % this function is to find the number of correct noisy images.
  vc=zeros(1,in(1,k));
  vc(1,1:in(1,k))=temp11(1,3:(in(1,k)+2));
  cr=temp11(1,1);
```

```
    bn=temp11(1,2);
    [a,nc]=grpcrt(cr,bn,nii,vc);
    temp11(2,(in(1,k)+4))=nc;
    if (mn~=0 |oo~=0)
     if temp(3*k,(in(1,k)+4))<nc
      temp(3*k,(in(1,k)+4))
      nc
      if (temp(3*k,(in(1,k)+3))>difr &mn~=0)
       temp11(2,(in(1,k)+3))=difr;
       %for y=1:ng
       % if (l(1,y)==0)
       %  cc=cc+1;
       %end
       %end

       temp(3*k,1:(in(1,k)+4))=temp11(2,1:(in(1,k)+4));
       save temp.dat temp -ascii -double

      end
     end
    end
   end
  end
end
```

LIST OF REFERENCES

1. Abdelwahab, M. M., and Mikhael, W. B., "Neural Network Pattern Recognition Employing Multicriteria Extracted From Signal Projections in Multiple Transform Domains", Proceedings of International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong, 2001, pp. 40-43.
2. Abdelwahab, M. M., and Mikhael, W. B., "Multistage Classification/Recognition Employing Vector Quantization Coding and Criteria Extracted from Nonorthogonal and Preprocessed Signals Representation", Applied Optics, Vol. 43, No.2, January 2004, pp.416-424
3. Allali, M., and DeBrunner, V., "Using Wavelet-Based Indices For Detecting Abrupt Changes in Signals", Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, Vol. 1, Nov. 2001, pp.726-729.
4. Atlas, L., Connor, J., Park, D., El-Sharkawi, M., Marks II, R., Lippman, A., Cole, R., and Muthusamy, Y., "A Performance Comparison of Trained Multi-Layer Perceptrons and Trained Classification Tree", IEEE International Conference on Systems, Man and Cybernetics, 14-17 Nov. 1989, pp.915-920.
5. Aydin, N., and Markus, H. S.,"Directional Wavelet Transform in the Context of Complex Quadrature Doppler Signals", IEEE Signal Processing Letters, Vol. 7, No.10, October 2000, pp.278-280.

6. Benediksson, J. A., Swain, P. H., and Ersoy, O. K., "Neural Network approaches versus statistical methods in classification of multisource remote sensing data", IEEE Transaction on Geoscience and Remote Sensing, July 1990, Vol. 28, pp.540-551.
7. Berg, A. P., and Mikhael, W. B., "A Survey of Mixed Transform Techniques for Speech And Image Coding", Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, ISCAS '99, Vol.4, 30 May-2 June 1999, pp.106-109.
8. Berg, A. P., and Mikhael, W. B., "An Efficient Structure and Algorithm for Image Representation Using Non-orthogonal Basis Images", IEEE Transactions on CAS II: Analog and Digital Signal Processing, Vol. 44, No. 10, Oct. 1997, pp. 818-828.
9. Biao, H., Ying, L., and Licheng, J., "Segmentation and Recognition of Bridges in High Resolution SAR Images", Proceedings of International Conference on Radar, 2001 CIE, pp.479-482.
10. Bishof, H., Schneider, W., and Pinz, A. J., "Multispectral classification of LANDSAT-images using neural networks", IEEE Trans. on Geoscience and Remote Sensing, May 1992, Vol. 30, pp. 482-490.
11. Briem, J. G., Benediktsson, J. A., and Sveinsson, J. R., "Use of Multiple Classifiers in Classification of data from multiple data sources", Geoscience and Remote Sensing Symposium, IGARSS'01, IEEE 2001, Vol.2, pp. 382-384.
12. Brodley, C. E., Friedl, M. A., and Strahler, A. H., "New Approaches to Classification in Remote Sensing Using Homogeneous and Hybrid Decision Trees to Map Land Cover", International Symposium for Geoscience and Remote Sensing, Vol. 1: 27-31, May 1996, pp. 532-534.

13. Bruzzone, L., Prieto, D. F., and Serpico, S. B., "A Neura-Statistical Approach to Multitemporal and Multisource Remote-Sensing Image Classification", IEEE Transactions on Geoscience and Remote Sensing, Vol.37, No.3, May 1999, pp.1350-1359.
14. Chan, J. C., Huang, C., and DeFries, R., "Enhanced Algorithm Performance for Land Cover Classification from Remotely Sensed Data Using Bagging and Boosting", IEEE Transaction on Geoscience and Remote Sensing, Vol. 39, No. 3, March 2001, pp.693-695.
15. Chang, T., and Kuo, C.C.J., "Texture Analysis and Classification with Tree-Structured Wavelet Transform", IEEE Transactions on Image Processing, Vol.2, No. 4, October 1993, pp. 429-441.
16. Chapa, J.O., and Rao, R. M., " Algorithms For Designing Wavelets To Match A Specified Signal", IEEE Transactions on Signal Processing, Vol. 48, No. 12, December 2000, pp.3395-3406.
17. Chapman, R. A., Norman, D. M., Zahirniak, D. R., Rogers, S. K., and Oxley, M. E., "Classification of correlation signatures of spread spectrum signals using neural networks', Proceedings of IEEE Conference of Aerospace and Electronic, 1991, Vol. 1, pp.485-491.
18. Chellappa, R., Wilson, C. L., and Sirohey, S., "Human and machine recognition of faces", a survey, Technical Report CAR-TR-731, CS-TR33339, University of Maryland, August 1994.

19. Chen, B., and. Varshney, P., K., "A Byesian Sampling Approach to Decision Fusion Using Hierarchical Models", IEEE Transaction on Signal Processing, Vol. 50, No.8, August 2002, pp.1809-1818.
20. Chung, K., Kee, S. C., and Kim, S. R., "Face recognition using principal component analysis of Gabor filter responses", Proceedings of 1999 International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-time Systems, pp. 53 -57.
21. Cohen, A., and Matei, B., "Compact Representation of Images by Edge Adapted Multiscale Transforms", International Conference on Image Processing, 2001, Vol. 1, pp.8-11.
22. Cosman, P. C., Oehler, K. L., Riskin, E. A., and Gray, R. M., "Using Vector Quantization for Image Processing", Proceedings of the IEEE, Vol. 81, No. 9, September 1993, pp.1326-1341.
23. Dapena, A. and Ahalt, S., " A Hybrid DCT-SVD Image-Coding Algorithm", IEEE Transaction on Circuits and Systems for Video Technology, Vol. 12, No.2, February 2002, pp.114-121.
24. Duin, R. P. W., Loog, M., Haeb- Umbac, R., "Multi-Class Linear Feature Extraction By Nonlinear PCA", Proceedings of 15th International Conference on Pattern Recognition, 2000, Vol.2, pp.398-401.
25. Elliott, D. F., and Rao, K. R., "Fast Transforms; Algorithms, Analysis, Applications", Academic Press, INC.1982.
26. Fogel, D. B., "An Information Criterion for Optimal Neural Network Selection", IEEE Transaction on Neural Network, Vol. 2, No.5, September 1991, pp.490-497.

27. Foltyniewiez, R., "Automatic Face Recognition via Wavelet and Mathematical Morphology", Proceedings of ICPR' 96, pp.13-17.

28. Fontenla- Romero, O., Alonso-Betanzos, A., and Guijarro-Berdinas, B., "Adaptive Pattern Recognition in the Analysis of Cardiotocographic Records", IEEE Transactions on Neural Networks, Vol.12, No.5, September 2001, pp.1188-1195.

29. Fukunaga, K., "Introduction to Statistical Pattern recognition", Academic Press. INC. 1990.

30. Gao, Q., Förster, P., Möbus, K. R., and Moschytz, G. S., "Fingerprint Recognition Using CNNS: Fingerprint Preprocessing", The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001, Vol.2, 6-9 May 2001, pp. 433-436.

31. Gao, Y., and M. K. H. Leung, M. K. H., "Face Recognition Using Line Edge Map," IEEE Trans. on Pattern Analysis And Machine Intelligence, Vol. 24, pp.764-779, 2002.

32. Gelfand, S. B., Ravishankar, C. S., and. Delp, E. J., "An Iterative Growing and Pruning Algorithm for Classification Tree Design", IEEE Transaction on Pattern analysis and Machine Intelligence, Vol. 13, No.2, February 1991, pp.163-174.

33. Gersho, A. and Gray, R. M., "Vector Quantization and Signal Compression", Kluwer Academic Publishers, 1993.

34. Giacinto, G., Roli, F., Fumera, G., "Unsupervised Learning of Neural Network Ensembles for Images Classification", IEEE International Joint Conference on Neural Networks, 2000, Vol. 3, pp.155-159.

35. Giusti, N., Masulli, F., Sperduti, A., "Theoretical and Experimental Analysis of a Two-Stage System for Classification", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 7, July 2002, pp. 893-904.

36. Go, J., Han, G., Kim, H., and Lee, C., "Multigradient: A New Neural Network Learning Algorithm For Pattern Classification", IEEE Transactions on Geoscience and Remote Sensing, Vol. 39, No.5, May 2001, pp.986-993.
37. Golchin, F., and Paliwal, K. K., "A Hybrid Coder for Code-Excited Linear Predictive Coding of Images", International Symposium on Signal Processing and its Applications, ISSPA, Gold Coast, Australia, 25-30 August, 1996, pp.49-52.
38. Gold, B., "Speech and audio signal processing: processing and perception of speech and music", New York: John Wiley, 2000.
39. Goumas, S. K., Zervakis, M. E., and Stavrakakis, G. S., "Classification of Washing Machines Vibration Signals Using Discrete Wavelet Analysis for Feature Extraction", IEEE Transactions on Instrumentation and Measurement, Vol. 51, No. 3, June 2002, pp.497-508.
40. Gray, R. M., Oehler, K.L., Perlmutte, K.O., Ohlsen, R.A., "Combining Tree-Structured Vector Quantization with Classification and Regression Trees", Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, Nov. 1993, pp.1494-1498.
41. Ha, T.M. and Bunke, H., "Off-Line Handwritten Numeral Recognition by Perturbation Method", IEEE Transaction on Pattern Analysis and Machine Intelligence, May 1997, Vol. 19, pp.535-539.
42. Haddadnia, J., Faez, K., and Ahmadi, M., "A Neural Based Human Face Recognition System Using An Efficient Feature Extraction Method With Pseudo Zernike Moment", Journal Of Circuits, System and Computers, 2002, Vol. 11, No. 3, pp.283-304.

43. Hansen, L., K., and Salamon, P., "Neural Network Ensembles", IEEE Transactions on Pattern Analysis and Machine Intelligence, October 1990, Vol. 12, No.10, pp.993-1001.
44. Hara, Y., Atkins, R.G., Yueh, S.H., Shin, R.T., and Kong, J.A., "Application of Neural Networks to Radar Image Classification", IEEE Transactions on Geoscience and Remote Sensing, January 1994, Vol. 32, No. 1, pp. 100-109.
45. Hatzichristos, G., and Fargues, M. P., "A Hierarchical Approach To The Classification of Digital Modulation Types in Multipath Environments", 35th Asilomar conference on Signals, Systems and Computers, 2001, vol.2, pp.1494-1498.
46. Haykin, S., and Deng, C., "Classification of Radar Clutter Using Neural Networks", IEEE Transactions on Neural Networks, Vol. 2, No. 6, November 1991, pp.589-600.
47. Herman, P. D., and Khazenie, N., "Classification of Multispectral Remote Sensing Data Using a Back-Propagation Neural Network", IEEE Transactions on Geoscience and Remote Sensing, Vol. 30, pp. 81-88, Jan. 1992.
48. Ho, T. K., Hull, J.J., and Srihari, S.N.," Decision Combination in Multiple Classifier Systems", IEEE Transactions on Pattern Analysis Machine Intelligence, Vol. 16, No. 1, pp. 66-75, Jan. 1994.
49. Hsieh, I. and Fan, K., "Multiple Classifier for Color Flag and Trademark Image Retrieval", IEEE Transactions on image processing, June 2001, Vol. 10, No.6, pp. 938-950.
50. Huang, W., and Mariani R., "Face Detection And Precise Eyes Location", Proceedings of 15th International Conference on Pattern Recognition, 2000, Vol.4. pp. 722-727.

51. Jain, A. K., Prabhakar, S. and Hong, L. "A Mulichannel Approach to Fingerprint Classification", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 21, No.4, April 1999, pp.348-359.
52. Jaisimha, M. Y., Goldschneider, J. R., Mohr, A. E., Riskin, E. A., Haralick, R. M., "On Vector Quantization for Fast Facet Edge Detection", IEEE International Conference on Acoustics, Speech, and Signal Processing, 1994. ICASSP-94, 19-22 Apr 1994, Vol.5, pp. V/37-V/40.
53. Jiali, Z., Jinwei, W., and Siwei, L., " A Multi-Moment Pattern Recognition Method Based On Neural Networks", Proceedings of the 5th International Conference on Signal Processing, 2000, Vol. 3, pp.1675-1678.
54. Joshi R. L., Jafarkhani H., Kasner J., Fischer T., Farvardin N., Marcellin M. W. and Bamberger, R., "Comparison of Different Methods of Classification in Suband Coding of Images", IEEE Transactions on Image Processing, Nov. 1997, Vol. 6, No.11, pp. 1473-1486.
55. Juang, B.H., and Katagiri, S., "Discriminant Learning for Minimum Error Classification", IEEE Transactions on Signal Processing, 1992, Vol. 40, No. 12, pp.3043-3054.
56. Kawatani, T., "Handwritten Kanji Recognition Using Combined Complementary Classifiers in a Cascade Arrangement", Proceedings of the Fifth International Conference on Document Analysis and Recognition, 1999. ICDAR ', 20-22 Sep 1999, pp. 503-506
57. Kijsirikul, B., Chongkasemwongse, K., "Decision Tree Pruning Using Backpropagation Neural Networks", Proceedings of International Joint Conference on Neural Networks, IJCNN '01, 2001, Vol.3, pp.1876-1880.

58. Kil, D. K., and Shin, F. B., "Automatic Road-Distress Classification and Identification Using a Combination of Hierarchical Classifiers and Expert Systems-Subimage and Object Processing", Proceedings of International Conference on image Processing, 1997, 26-29 Oct. 1997, Vol.2, pp.414-417.

59. Kittler, J., Hatef, M., Duin, R.P.W., and Matas, J., "On Combining Classifiers", IEEE Transaction on Pattern Analysis and Machine Intelligence, March 1998, Vol. 20, pp. 226-239.

60. Kohir, V. V., and Desai, U. B., "Face Recognition", IEEE International Symposium on Circuit and Systems, May 28-31, 2000, Geneva, Swizerland.

61. Kosaka, T., and Omatu, S., "Classification of Italian Liras Using the LVQ Method", Neural Network, 2000, LJCNN, Proceedings of IEEE-INNS-ENNS International Joint Conference on, Vol.3. pp. 145-148.

62. Kundur, D., Hatzinakos, D., and Leung, H., "Robust Classification of Blurred Imagery", IEEE Transactions on Image Processing, Vol.9, No. 2, February 2000, pp. 243-255.

63. Kupinski, M. A., Edwards, D. C., Giger, M. L., and Metz, C. E., "Ideal Observer Approximation Using Bayesian Classification Neural Networks" IEEE Transactions on Medical Imaging, Vol.20, No.9, September 2001, pp.886-899.

64. Lazarevic, A., Obradovic, Z., "Effective Pruning of Neural Network Classifier Ensembles", Proceedings of International Joint Conference on Neural Networks, IJCNN '01, 2001, Vol.2, pp.796-801.

65. Leondes, C. T.,"Image Processing and Pattern Recognition", (Academic Press, 1998).

66. Lu, Y.," Knowledge Integrations in a Multiple Classifier System", Applied Intelligence, April 1996, Vol.6, no.2, pp.75-86.

67. Luo, X., and Mirchandani, G., "An Integrated Framework for Image Classification", Proceedings of 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing, Istanbul, Turkey, Vol. 1, pp. 620 -623.

68. Lyons, M. J., Budynek, J., and Akamatsu, S., "Automatic Classification of Single Facial Images", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.21, No.12, December 1999, pp.1357-1362.

69. McGuire, P., and Eleuterio, G. M. T. D', "Eigenpaxels and a Neural-Network Approach to Image Classification", IEEE Transactions on Neural Networks, Vol. 12, No.3, May 2001, pp.625-635.

70. Meer, P., and Georgescu, B., "Edge Detection with Embedded Confidence", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.23, No.12, December 2001, pp.1351-1365.

71. Mikhael, W. B., and Abdelwahab, M. M., "Multi Criteria Multi Transform Neural Network", Circuits, Systems and Signal Processing, Vol. 21 No.5 pp. 451-460 Sep.-Oct. 2002.

72. Mikhael, W. B., and Abdelwahab, M. M., "A Novel Intelligent Classifier/Recognizer Employing Vector Quantization Coding of Non Orthogonal Signal and Preprocessed Signal Representations", ICNNSP'03 Conference, Nanjng china, Dec. 14-17 2003,

73. Mikhael, W. B., and Abdelwahab, M. M., "Accurate Classification of Signals Employing a Multistage Binary Grouping Structure", Submitted to IEEE Transaction on Pattern Analysis and machine Intelligence.

74. .Mikhael, W.B., Abdelwahab, M. M., and Krishnan, V. "Self Designing Intelligent Signal Processing System Capable of Evolutionary learning for classification/Recognition of

one and multidimensional Signal", UCF Patent Disclosure August 24, 2001, Provisional obtained.

75. Mikhael, W.B., Krishnan, V., and Abdelwahab, M. M., Projection of Sampled Signals in Different Transform Domains With Useful Applications, The Fourth International Conference on Sampling Theory and Its Applications, Orlando, FL, May 2001. (Invited Paper and Plenary Talk).
76. Mikhael, W. B., and Ramaswamy, A., "Resolving Images in Multiple Transform Domains with Applications", Digital Signal Processing, Vol.5, 1995, pp. 81-90.
77. Mikhael. W. B., and Ramaswamy, A., "An Efficient Representation of Nonstationary Signals Using Mixed-Transforms with Application to Speech", IEEE Transactions on Circuit and Systems-II: Analog and Digital Signal Processing, Vol. 42, No. 6, June 1995, pp393-401.
78. Mikhael, W. B., and Ramaswamy, A., and Pourparviz, G., "A Direct Approach for Representing Non-Stationary Signals Using Mixed Transforms", Proceedings of the 34th Midwest symposium on Circuits and Systems, 1991, pp.27-30.
79. Mikhael, W.B., and Spanias, A., "Accurate Representations of Time Varying Signals Using Mixed Transforms with Applications to Speech", IEEE Transactions on CAS, Vol. 36, No. 2, Feb. 1989, pp.329-331.
80. Naga, K.O., "Face Recognition by Distribution Specific Feature Extraction", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2000, Vol.1, 2000, pp.278-285.

81. Patel, J. N., Khokhar, A. A., and Jamieson, L. H., "Scalability of 2-D Wavelet Transform Algorithms: Analytical and Experimental Results on MPPs", IEEE Transactions on Signal Processing, Vol. 48, No.12, December 2000, pp.3407-3419.

82. Pattichis, C. S., Pattichis, M. S., and Micheli-Tzanakou, E., "Medical Imaging Fusion Applications: An Overview", Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, 2001, Vol. 2, Nov. 2001, pp.1263-1267.

83. Pattichis, M. S., Panayi, G., Bovik, A. C. and Hsu, S. "Fingerprint Classification Using an AM-FM Model", IEEE Transaction on Image Processing, Vol. 10, No.6, June 2001, pp.951-954.

84. Prampero, P.S., and Carvalho, A.C., "Recognition of Vehicles Silhouette using Combination of Classifiers", International Joint Conference on Neural Networks, (IJCNN'98), 1998, pp. 1723-1726.

85. Rice, B. F., "Introduction to Automatic Signal Classification", International conference on signal processing applications and technology 1997.

86. Roberts, A., and Yearworth, M., "A Comparison of Pre-Processing Transforms for Neural Network Classification of Character Images", International Conference on Image Processing and its Applications, 7-9 Apr 1992, pp.189-192.

87. Rowe, N. C., and Grewe, L. L., "Change Detection for Linear Features in Aerial Photographs Using Edge-Finding", IEEE Transactions on Geoscience and Remote Sensing, Vol. 39, No. 7, July 2001, pp.1608-1612.

88. Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., and Sutter, B. W., "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function",

IEEE Transaction on Neural Network, Letters, Vol.1, No. 4, December 1990, pp. 296-298.

89. Sandberg, I. W., "General Structures for Classification", IEEE Transaction on Circuits and Systems-1: Fundamental Theory and application, May 1994, Vol. 41, No. 5, pp. 372-376.
90. Sarlashkar, M. N., Bodruzzaman, M., and Malkani, M.J., "Feature Extraction Using Wavelet Transform For Neural Network Based Image Classification", Proceedings of the Thirtieth Southeastern Symposium on System Theory, 8-10 March,1998, pp.412-416.
91. Senior, A., " A Combination Fingerprint Classifier", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No.10, October 2001, pp.1165-1174.
92. Shimshoni, Y., and Intrator, N., "Classification of Seismic Signals by Integrating Ensemble of Neural Networks", IEEE Transactions on Signal Processing, Vol. 46, No. 5, May 1998, pp.1194-1201.
93. Simard, M., Saatchi, S. S., and Grandi, G. D., "The Use of Decision Tree and Multiscale Texture for Classification of JERS-1 SAR Data over Tropical Forest", IEEE Transactions on Geoscience and Remote Sensing, Vol. 38, No.5, September 2000, pp.2310-2321.
94. Sinha, S. K., and Karry, F., "Classification of Underground Pipe Scanned Images Using Feature Extraction and Neuro-Fuzzy Algorithm", IEEE Transaction on Neural Networks, Vol. 13, No.2, March 2002, pp.393-401.
95. Srivastava, A., Han, E. and Kumar, V., "Parallel Formulations of Decision-Tree Classification Algorithms", Proceedings of International Conference on Parallel Processing, 1998, pp237-244.

96. Suen, C.Y., Kim, J., Kim, K., Xu, Q., and Lam, L., "Handwriting Recognition-The Last Frontiers", Proceedings of 15th International Conference on Pattern recognition, 2000, Vol.4, pp.1-10.

97. Terry, P. J., and Vu, D., "Edge Detection Using Neural Networks", The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, 1993, 1-3 Nov 1993, Vol.1, pp.391-395.

98. Tilki, J. F., and Beex, A. A., "Image Data Compression Using Multiple Bases Representation", Proceedings of the 26th Southeastern Symposium on System Theory, 1194, pp.457-461.

99. Tran, T. D., "The BinDCT: Fast Multiplierless Approximation of the DCT", IEEE Signal Processing Letters, Vol.7, No.6, June 2000, pp.141-144.

100. Tu, p., and Chung, J., "A New Decision-Tree Classification Algorithm for Machine Learning", Proceedings of the 1992 IEEE International Conference on Tools with AI Arlington VA, Nov.1992, pp.370-377.

101. Türkoğlu, I., and Arslan, A., "Optimisation Of The Performance Of Neural Network Based Pattern Recognition Classifiers With Distributed Systems", Proceedings of Eighth International Conference on Parallel and Distributed Systems, 2001, pp.379-382.

102. Ueda, N., "Optimal Linear Combination of Neural Network for Improving Classification Performance", IEEE Transaction on Pattern Analysis and Machine Intelligence, February 2000, Vol. 22, No.2, pp.207-214.

103. Vailaya, A., Figueiredo, M. A. T., Jain, A. K., and Zhang, H., "Image Classification of Content-Based Indexing", IEEE Transactions on Image Processing, Vol. 10, No. 1, pp. 117-130, Jan. 2001.

104. Valverde, F.L., Guil, N., Munoz, J., Nishikawa, R., and Doi, k., "An Evaluation Criterion for Edge Detection Techniques In Noisy Images", Proceeding of international Conference on Image Processing, 2001, Vol. 1, pp.766-769.

105. Verma, B., and Zakos, J., "A Computer-Aided Diagnosis System for Digital Mammograms Based on Fuzzy-Neural and Feature Extraction Techniques", IEEE Transactions on Information Technology in Biomedicine, Vol.5, No.1, March 2001, pp.46-54.

106. Wu, Y., and Tai, S., "Medical Image Compression by Discrete Cosine Transform Spectral Similarity Strategy", IEEE Transaction on Information Technology in Biomedicine, Vol. 5, No. 3, September 2001, pp. 236-243.

107. Xu, L., Krzyzak, A., and Suen, C. Y., "Methods For Combining Multiple Classifiers And Their Applications To Handwriting Recognition", IEEE Transactions on Systems, Man and Cyb.22, vol. 22, May-June, 1992, pp. 418-435.

108. Yu, X., Dent, D., and Osborn, C., "Classification of Ballistocardiography Using Wavelet Transform and Neural Networks", 18th Annual International Conference of IEEE Engineering in Medicine Biology Society, Amsterdam 1996, pp.937-938.

109. Zeng, Y., and Starzyk, J., "Statistical Approach to Clustering In Pattern Recognition", Proceedings of the 33rd Southeastern Symposium on System Theory, Mar 2001, pp. 177 - 181.

110. Zhang, G. P., "Neural Networks for Classification: A Survey", IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Vol. 30, No.4, November 2000, pp.451-462.

111. Zhang, P., Chen, L., and Kot, A. C., "A Novel Hybrid Classifier for Recognition of Handwritten Numerals", IEEE International Conference on Systems, Man and Cybernetics, 2000, Vol.4, pp.2709-2714.

112. Zhang, Y., Zheng, J., Liu, H., and Jiao, L., "An Efficient Method of Radar Target Classification", Proceedings of International Conference on Radar, 2001, pp.502-505.

113. Zhou, W., "Verification of The Nonparametric Characteristics of Backpropagation Neural Networks for Image Classification", IEEE Transactions on Geoscience and Remote Sensing, March 1999, Vol. 37, No. 2, pp. 771-779.