



5-1999

A neutral network approach to automated classification of cracks in images of highway pavement using vectorized data

Judy Nicole Edwards

Follow this and additional works at: https://trace.tennessee.edu/utk_gradthes

Recommended Citation

Edwards, Judy Nicole, "A neutral network approach to automated classification of cracks in images of highway pavement using vectorized data. " Master's Thesis, University of Tennessee, 1999.
https://trace.tennessee.edu/utk_gradthes/9814

This Thesis is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a thesis written by Judy Nicole Edwards entitled "A neural network approach to automated classification of cracks in images of highway pavement using vectorized data." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Bruce Whithead, Major Professor

We have read this thesis and recommend its acceptance:

Jack Hansen, Dinesh Mehta

Accepted for the Council:

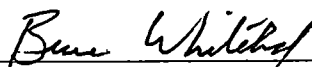
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

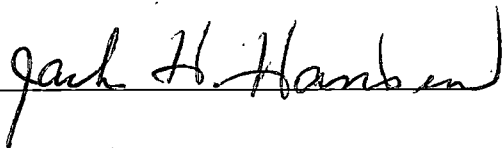
To the Graduate Council:

I am submitting herewith a thesis written by Judy Nicole Edwards entitled "A Neural Network Approach to Automated Classification of Cracks in Images of Highway Pavement Using Vectorized Data." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.



Bruce Whithead, Major Professor

We have read this thesis
and recommend its acceptance:







Associate Vice Chancellor
and Dean of The Graduate School

A Neural Network Approach to Automated Classification
of Cracks in Images of Highway Pavement Using
Vectorized Data

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Judy Nicole Edwards

May 1999

Acknowledgements

I would like to express many thanks to my thesis committee, Dr. Bruce Whitehead, Dr. Jack Hansen, and Dr. Dinesh Mehta, for their advice during the writing of this thesis.

I would also like to thank Dr. Jack Hansen and Dr. Alfonso Pujol, Jr. for providing a comfortable and relaxed environment in which this research could be performed.

Sincere gratitude is due to *The Connecticut Department of Transportation* for providing the imagery used here and in the research leading to this thesis.

I would like to thank all of my family and friends who supported and encouraged me. I would also like to especially thank Marcus March, whose love, faith and support made me strong and kept me focused. Finally and most importantly, I give thanks to my Lord, Jesus Christ, who gave me the strength to continue my education and complete this thesis.

Abstract

The collection and interpretation of highway pavement distress data is an expensive process. This process, in its traditional sense, required many man hours, as technicians have to survey the highway. This process is also quite dangerous and subject to traffic and weather conditions.

Much research has been performed in order to automate the collection and interpretation phases. This research focuses on an alternative methodology for classification of highway pavement distresses. The Learning Vector Quantizer, an artificial neural network architecture, was implemented to classify images of highway pavement according to the longitudinal, transverse, fatigue and block crack types. Data was recorded from vectorized crack segments produced by a segmentation process. The images were recursively broken into smaller blocks, forming a quad-tree type structure. Each block in the image tree was then classified as to the major distress prevalent.

This technique should allow for easier measurement of extent and severity. Also it allows records to be kept on individual pieces of highway for comparison with future monitoring.

The overall results of the classification were satisfactory, with longitudinal and transverse classification outperforming block and fatigue. This was to be expected as block and fatigue crack types are harder to differentiate. With additional research, this approach should prove to be very beneficial to the pavement management field.

Contents

1	Introduction	1
1.1	Previous Classification Techniques	2
1.2	Pavement Condition Rating	6
1.3	Extent and Severity	7
1.4	Purpose of Research	8
2	Pattern Classification	10
2.1	Preclassification techniques	10
2.1.1	Image Enhancement	10
2.1.2	Image Segmentation	13
2.2	Classification Algorithms	13
2.2.1	Bayes Classifier	14
2.2.2	K Nearest Neighbor Classifier	16
2.2.3	Artificial Neural Networks	17
3	Neural Networks	19
3.1	Biological Background	19
3.2	Artificial Neural Networks	21

3.3	Architectures and Learning Algorithms	22
3.3.1	The Perceptron	22
3.3.2	Multi-layer Feed Forward Network and Backpropagation	24
3.3.3	Unsupervised Vector Quantizer	30
3.3.4	Supervised Learning Vector Quantizer	31
3.3.5	LVQ2 Algorithm	32
3.4	Related Research	33
3.5	Technical Plan	38
4	Technical Implementation	39
4.1	The Data	39
4.2	Segmentation Technique	40
4.3	Preprocessing	42
4.3.1	Calculating Initial Parameters	42
4.3.2	Growing/Connecting Cracks and Removing Spurs	46
4.3.3	Multiple Resolution Data	49
4.3.4	Ground Truthing	53
4.4	Classification	54
4.5	Algorithm	59
5	Results	64
6	Conclusion	83
6.1	Recommendations for Future Research	87

Bibliography	88
Appendices	92
A Images Used in Research	93
B Distress Types and Severities	103
Vita	110

List of Figures

2.1	Non-weighted 3x3 kernel for neighborhood averaging.	11
2.2	Weighted 3x3 kernel for neighborhood averaging.	12
2.3	3x3 kernel for neighborhood ranking.	12
2.4	Portion of segmented image: Foreground is white.	14
3.1	Simple human neurons.	20
3.2	Perceptron	23
3.3	Linearly separated classes.	23
3.4	Multil-layer feed foward network.	26
3.5	Vector quantizer.	30
3.6	Variances in orientations of crack segments.	36
4.1	Shorter segments are spurs.	47
4.2	Division of the image. Level 0 is the entire image.	50
4.3	A crack segment that crosses boundaries of block.	51
4.4	A portion of image with overlaid grid.	54
4.5	Unscaled data. It is difficult to determine decision regions in the data because it is so compact	56

4.6	Scaled data from Figure 4.5. Decision regions can now be determined because the data is less compact.	57
4.7	Gaussian Radial Basis Function	59
5.1	Portion of original image.	65
5.2	Segmentation of Figure 5.1. Note how that some of the crack does appear.	65
5.3	Segmentation of figure 5.1. More of the crack is connected, but the noise level is also increased.	66
5.4	Noisy segmentation shown for full image.	67
5.5	Portion of vectorized image used for human interpretation of distress types in the image.	68
5.6	Plot of accuracy as function of the number of iterations used to train the neural network.	70
5.7	Plot of orientation vs. standard deviations of the lengths of the transverse crack type. Anything between 0 and 8 can be considered relatively small.	71
5.8	Plot of orientation vs. standard deviations of the lengths of the miscellaneous crack type. Anything between 0 and 8 can be considered relatively small.	72
5.9	Centers placed in the length dimension of the input of the training data.	73
5.10	Plot of the TE as a function of the number of epochs.	74
5.11	Plot of the accuracy of the classifier as a function of the number of epochs.	75

5.12	Plot of the training samples in the orientation dimensions.	79
5.13	Trained centers placed in the orientation dimensions of the input of figure 5.12.	79
5.14	Plot of the testing samples in the orientation dimensions. Notice how most of the fatigue samples lie in the outer edges of the decision region displayed in Figure 5.13.	80
5.15	Confusion Matrix for the final classification of the testing data.	80
6.1	Portion of image with grids overlaid.	85
6.2	Portion of image showing longer crack segments with shorter crack segments.	86
A.1	Training Image 1	94
A.2	Training Image 2	95
A.3	Training Image 3	96
A.4	Training Image 4	97
A.5	Training Image 5	98
A.6	Testing Image 1	99
A.7	Testing Image 2	100
A.8	Testing Image 3	101
A.9	Testing Image 4	102
B.1	Fatigue crack with alligator/chicken wire pattern typical in fatigue cracking.	104
B.2	Moderate severity Fatigue cracking.	105

B.3	High severity Fatigue cracking.	105
B.4	High severity Fatigue cracking with spalling.	106
B.5	Moderate severity Block cracking.	106
B.6	High severity Block cracking.	106
B.7	Moderate severity Longitudinal cracking.	107
B.8	High severity Longitudinal cracking.	107
B.9	Low severity Transverse cracking.	108
B.10	Moderate severity Transverse cracking.	108
B.11	High severity Transverse cracking.	109

List of Tables

4.1	Table of constant values used in algorithm.	63
5.1	Results for training on the data from the ground truth of original image. The results of human interpretation of the corresponding vectorized image were also provided.	76
5.2	Results for testing on the data from the ground truth of original image. The results of human interpretation of the corresponding vectorized image are also provided.	77
5.3	Results for training with data obtained from the ground truth of the vectorized images.	81
5.4	Results for training with data obtained from the ground truth of the vectorized images.	82

Chapter 1

Introduction

Each year the transportation departments of many states spend millions of dollars investigating the conditions of the roadways in their states. This investigation is necessary to insure the safety of the citizens.

While necessary this investigation is quite costly as there are several parts that require the expertise of several people. One of the most expensive and time-consuming stages in the process is the collection and analysis of pavement distress types. The state hires several technicians who survey sections of the pavement and record the distresses prevalent. When considering the average number of roads and the lengths of these roads in the state, numerous individuals are needed for the job at a significant cost to the state. And this considers only one area of concentration in the investigative process.

The job of surveying and analyzing the road is also tedious and is subject to and dictated by traffic and weather conditions. Data collection accuracy is affected by high traffic volume, lighting, and severe weather conditions.

In the past few years, the transportation agencies have invested much money to alleviate many of the problems associated with the investigative process including reducing the overall cost. One company, ARAN, has designed a vehicle, which can record video footage of highway pavement using several cameras, strategic strobe lighting and various vibration sensitive devices. This solves the problem of using field crews' labor for collection of the pavement conditions.

There have also been several successful attempts at improving the automated classification of distresses in pavement. This is the primary focus of this research. The following sections describe some of the previous research that have been performed and discusses some topics related to pavement distress analysis. A brief description of some of the previous research follows.

1.1 Previous Classification Techniques

Ohio Department of Transportation

Acosta, Figureroa and Mullen [1] developed a video image processing technique for evaluating pavement surface distress for the Ohio Department of Transportation. For the classification phase of this technique they used a rulebased approach. Several distress types were identified for classification, namely:

1. ravelling,
2. bleeding,
3. patching,
4. rutting,

5. settlement,
6. corrugations,
7. wheel track cracking,
8. transverse cracking,
9. longitudinal cracking,
10. edge cracking,
11. and random cracking.

All or some combinations of these distress types can be found in flexible, jointed and composite pavement.

The classification scheme was divided into 5 major steps:

1. depth related cluster classification,
2. cluster preliminary identification,
3. cluster classification,
4. and final classification.

Clusters are adjacent pixels of "foreground" and "maybe foreground." Foreground pixels are those pixels believed to belong to a pavement distress shown in the image, and "maybe foreground" pixels are ones that probably belong to a distress but are still questionable. Several features are extracted from the crack based on the gray values recorded for the pixels in the clusters. Most of the distress types found in the test images were accurately recognized. No quantitative results were provided.

This classification approach involved using a Back Propagation Neural Network for classifying the following distress types:

1. transverse cracks,
2. alligator cracking,
3. block cracking,
4. potholes,
5. and longitudinal cracks.

Images were divided into 32x32 pixel subimages. There were 3 phases to the classification: first preliminary classification, second preliminary classification, and final classification. The first preliminary classification classified each sub-image as non-damaged, pothole, and linear/jointed. This was accomplished by observing the number of distressed pixels P . Subimages were classified according to:

$$P < 40 \text{ non-damaged,}$$

$$P > 550 \text{ pothole,}$$

$$40 \leq P \leq 550, \text{ linear-jointed.}$$

The second preliminary classification distinguished linear subimages from jointed subimages. The Connected Component Labeling algorithm was used. This method traced along the distressed pixels and assigned labels to non-distressed pixels. If the number of different labels in the subimage was ≤ 2 , the subimage was considered

linear. If the number of different labels was at least 3, the subimage was jointed. Percentages of the classification for each subimage from both preliminary classification were used as input to the Back Propagation Neural Network. Final image classification was determined by the neural network. Longitudinal, transverse, alligator, and block crack types were classified with 100 percent accuracy. The classification of potholes achieved an accuracy of 93 percent.

Chou, JaChing and Wende A. O'Neill [5]

This classification method also used a Back Propagation Neural Network. Longitudinal, transverse, combined longitudinal and transverse cracks, right and left diagonal cracks, alligator, and non-distress types were the focus of this approach. The sizes of the images were 482x512 pixels.

Hu's, Bamich and Zeurike moments were calculated using the gray values of the pixels in the image and used as input to the neural network. The classification results were 100 percent accurate.

El Sanhoury, Ibrahim Mahmoud [8]

Sanhoury's work focused on classifying alligator, block, longitudinal and transverse crack types. The feature vectors were composed of three variables: density, inertia ratio, and angle of inertia. These features were extracted from bounding boxes that enclosed the distressed pixels within the image.

Decision trees and simultaneous approaches were implemented. The simultaneous approach computes the Euclidean distance between the feature vector and a prototype vector for each class. The prototype vector is the mean vector for a class. The class

of the prototype vector that yields the smallest Euclidean distance is assigned to the feature vector in question.

The decision tree approach used different aspects of the feature vector in order to determine if the distress type was 1D or 2D. 1D cracks are longitudinal and transverse cracks. 2D cracks are alligator and block crack types. Once it was determined if the type was 1D or 2D, the classifier had to determine what specific type of crack it was.

The accuracy of classification achieved for simultaneous approach overall was 56 percent. The decision tree approach did slightly better with an accuracy of 70 percent. The classifier was able to determine the difference between longitudinal and transverse cracks, but did not differentiate well between alligator and block crack types.

1.2 Pavement Condition Rating

Most transportation departments base the pavement condition rating (PCR) on two parameters: riding quality of the pavement surface and the extent and severity of pavement surface distresses present. The extent and severity are an indicator of the pavement structural condition.

Some distresses result in surface distortion and unevenness, which produces poor riding quality. An example of this is fatigue cracking that has gone unrepaired and has worsened to the point that pieces of the pavement become removed, leaving holes. These holes eventually grow larger and more numerous reducing the riding quality of that particular roadway.

Different distress manifestations may be grouped according to three categories [3]. The primary focus of this research is the crack category. The manifestations

in each category are evaluated in terms of the extent and severity. This along with the riding quality, which is not discussed in this research, constitutes the basis for rating of pavement condition. This rating should reflect the average condition of the pavement.

The rating of the pavement should be carried out on pavement sections that exhibit the same surface types. For example, if there is a strip of highway that is constructed of flexible (asphalt) material, it should not be grouped and rated with a following strip of highway that is possibly Portland cement concrete. This is because different pavement types exhibit distresses differently. Thus, two different types of highway with the similar traffic loading will result in different requiring different forms of maintenance.

The rating should be performed regularly to ensure that maintenance and rehabilitation is conducted as needed.

1.3 Extent and Severity

Severity refers to how bad the problem is. While, extent describes how much of the pavement is effected by the distress manifestation. There are three degrees of severity: low, moderate, and high. These descriptors are based on the width, length, and other parameters as they pertain to the different distress types.

Extent also has three degrees: intermittent, frequent, and extensive. These descriptors are based on the percentage of pavement effected and how often the distress occurs. For different distresses, the extent is defined similarly with a few variations as they pertain to the distress type. The following definitions describe the extent rating:

1. intermittent: less than 20 percent of pavement surface is affected and is in localized areas only,
2. frequent: 20 percent to 50 percent of pavement surface is affected and is evenly spread out over the length of pavement or in localized areas,
3. extensive: more than 50 percent of pavement surface is affected and is evenly spread out over the length of pavement.

1.4 Purpose of Research

The purpose of this research is to analyze a new technique of pavement image classification. Classification will focus only on crack manifestations. More specifically, alligator, block, longitudinal, transverse and miscellaneous crack types will be identified. Like many of the previous research techniques, a neural network classifier was implemented. The primary difference is the input data used. The data is extracted from vectorized segments. The crack segments are represented as vectors of points. Data extracted from vectors of (x,y) points that describe the crack segments are used as input to the neural network classifier. These parameters that are extracted provide information to researchers who may want to analyze the data for closer inspection.

The technique also uses a divide and conquer strategy. This strategy divides the pavement image into many sections. Each section is classified as to the major distress type prevalent. This serves two purposes. It should allow for ease in calculating the severity and extent of distress types in the pavement image and allows a record to be kept of individual sections of pavement. This is useful for future research of pavement

and their distresses.

Chapter 2

Pattern Classification

2.1 Preclassification techniques

Before images can be classified, certain techniques must be applied in order to make the image more appropriate for classification. Most images are degraded and must be cleaned up before any further manipulation can be accomplished. Also in order to perform the classification, certain features must be extracted that describe the image. These topics are the subject of the following subsections.

2.1.1 Image Enhancement

There is a gray area between image enhancement and correcting image defects, but for the purposes of this research we will consider them the same. Image enhancement is the process of improving a degraded image. A degraded image is generally one that contains defects. Defects can be introduced into an image in a variety of ways. Usually defects are captured during image acquisition as a result of imperfect detec-

1	1	1
1	1	1
1	1	1

Figure 2.1: Non-weighted 3x3 kernel for neighborhood averaging.

tors, inadequate or non-uniform illumination, or an undesirable viewpoint. The most common defect associated with pavement images is noise as a result of non-uniform lighting and particles in the pavement with varying coefficients of reflectivity. The most common ways of removing noise from the images are neighborhood averaging and ranking.

Neighborhood Averaging

The most common way to perform neighborhood averaging is to replace each pixel by the average of itself and neighboring pixels. This is usually performed using a non-weighted square kernel starting from 3x3 and up. Equation 2.1 describes the calculation used to perform the averaging.

$$P_{x,y}^* = \frac{\sum_{i,j=-m}^{+m} W_{i,j} P_{x+i,y+j}}{\sum_{i,j=-m}^{+m} W_{i,j}} \quad (2.1)$$

Each value in the square neighborhood is multiplied by a weight in the kernel W . Figure 2.1 is an example of a non-weighted 3x3 kernel. In this figure, the central pixel would be replaced by the sum of the weighted values divided by the sum of the weights. The kernel would be applied for each pixel in the image, except those along the edges of the image. For practical purposes, it is assumed that no useful information is found in the edges.

1	2	1
2	4	2
1	2	1

Figure 2.2: Weighted 3x3 kernel for neighborhood averaging.

2	4	2
4	4	4
2	4	2

Figure 2.3: 3x3 kernel for neighborhood ranking.

Noise is reduced, but blurring may and usually does result from the averaging process. This can be overcome by using a weighted kernel as seen in Figure 2.2. Weighted techniques are used to obtain averages which smooth or accent the possible existence of linear features according to where the weights are placed in the kernel.

Neighborhood Ranking

Another technique used to smooth noise is neighborhood ranking. This method ranks pixels in a neighborhood according to their brightness. The median filter (Figure 2.3) is an example of neighborhood ranking.

The median value of the neighborhood pixels are used to replace the central pixel values. There are many types of median filters that may be applied to images. The selection of one is based on the type of the image in question. Median filters do not degrade edges and thus, allow for repeated application if necessary. The mode filter is another example of neighborhood ranking that uses the mode of the neighborhood

pixels as opposed to the median.

2.1.2 Image Segmentation

Image segmentation can be thought of as separating the foreground from the background of an image. The foreground is usually the features and objects in the image that are under examination for classification. Selection of these features or objects is an important prerequisite for understanding an image

One of the most popular methods of segmentation is thresholding. The simplest form of thresholding entails defining a range of brightness (threshold) for an image. Any pixels in the image that belong to this range are labeled as foreground and all other pixel values in the image are assigned to background. This image is usually represented by a binary image. That is, pixels belonging to foreground are assigned a value of 255 and background pixels are assigned zero or vice versa. Thus when displayed, as in Figure 2.4 features show up as white against a black background.

Simple as this technique may seem, the selection of an appropriate threshold is not trivial. This value depends largely on the image in question (i.e. the amount of noise and types of feature being extracted). One method of setting the threshold involves using the histogram of the image. Images utilized in this research were segmented with a new segmentation technique that is discussed in detail by Hansen [10].

2.2 Classification Algorithms

Classification can be thought of as mapping unknown objects to certain classes based on the features of the objects. Two types of classification are parametric and



Figure 2.4: Portion of segmented image: Foreground is white.

non-parametric. Parametric techniques are used if you know or are willing to assume the underlying probability density and distribution of the numeric data. Only the parameters need to be estimated. Non-parametric techniques are based on classifying using the information directly from a data set without regard to classical or statistical assumptions and descriptors. A review of common classification techniques follows.

2.2.1 Bayes Classifier

A Bayes classifier is a statistical classification technique that chooses the most probable class given certain features. Define F as a vector of n arbitrary features $(f_1, f_2, f_n, \dots, f_n)$. Let $C = (c_1, c_2, c_3, \dots, c_m)$ be the set of m classes to which any given feature vector can be classified. F is classified in class c_i if:

$$p(c_i|F) > p(c_j|F) \text{ for all } j \neq i. \quad (2.2)$$

These probabilities are calculated using Bayes rule:

$$p(c_i|F) = \frac{P(c_i)p(F|c_i)}{\sum_{i=1}^m P(c_i)p(F|c_i)} \quad (2.3)$$

The a priori probabilities $P(c_i)$ and the conditional densities $p(F|c_i)$ are assumed to be known but usually are not. The prior probability can be estimated from training data or from previous experiments. However, inferring conditional densities from training data is not trivial. Some assumptions about the distribution of the data must be made and an appropriate density function which best describes the data must be selected. The parameters for the selected density function must then be estimated.

The data are often assumed to be normally distributed and the feature vector represents $n > 1$ features. Thus, a multivariate normal distribution should be a fairly accurate representation of the conditional density of the data. There are several ways to estimate the parameters: mean (μ) and covariance (Σ), which include the Method of Moments and Maximum Likelihood Estimators. The Maximum Likelihood Estimator is the typical method employed.

The maximum likelihood estimate of a parameter is that value which, when substituted into the density function, produces that density for which the joint density of obtaining the entire observed set of samples is maximized [9]. To compute the maximum likelihood estimate of parameter α , the derivative of the logarithm of the density function with respect to α is computed. Setting this derivative to zero and solving for α results in the maximum likelihood estimate of α .

If the random samples $F_1, F_2, F_3, \dots, F_j$ are drawn independently from the popu-

lation then:

$$p(F_1, F_2, F_3, \dots, F_j) = p(F_1) \cdots p(F_j) = \prod_{i=1}^j p(F_i).$$

Given this assumption, the estimated mean μ is as follows:

$$\mu = \frac{1}{N} \sum_{k=1}^N F_k \quad (2.4)$$

and estimated covariance Σ is

$$\Sigma = \frac{1}{N} \sum_{k=1}^N (F_k - \mu)^T \quad (2.5)$$

Having estimated the parameters μ and Σ , the conditional density $p(F|c_j)$ can be computed using:

$$p(F|c_j) = \frac{1}{\sqrt{\det \Sigma} (2\pi)^n} e^{-\frac{(F-\mu)^T \Sigma^{-1} (F-\mu)}{2}} \quad \text{where } n = \text{the number of features} \quad (2.6)$$

2.2.2 K Nearest Neighbor Classifier

The k Nearest Neighbor (k-nn) algorithm is a non-parametric classification technique. That is, it does not require knowledge or any assumptions of the underlying probability density functions of the data. It is usually reasonable to assume sample points that are in close proximity to each other will probably have the same classification. With this assumption, examining the neighbors of the sample point in question should lead to the correct the classification. The k Nearest Neighbors algorithm takes advantage of this assumption and assigns the unclassified sample to the class most heavily represented by its k nearest neighbors.

The error rate of the k-nn technique approaches that of the Bayes classifier as both k and k/n grow infinitely large, where n is the total number of samples in

the reference set. As the number of samples in a small area grows, the percentage from each class also increases. This serves as an approximation of the posterior class conditional probability $P(C|X)$.

In any event, the probability of error for k-nn will never be more than twice the upper bound of the error rate for the Bayes classifier.

2.2.3 Artificial Neural Networks

Artificial Neural Networks (ANN) are pattern classifiers motivated by natural neural networks. Humans make most of their decisions based on the recognition and interpretation of patterns. This was one of the primary motivations for the research of ANN.

ANN have several desirable characteristics, the more relevant ones as they pertain to classification being:

1. They are resistant to noise in the input data.
2. They possess the ability to generalize.
3. They are able to produce accurate solutions when only partial information is available.

These situations are ones commonly faced in the field of pattern recognition and classification.

Artificial Neural Networks also have been shown to outperform statistical pattern classifiers. This is partially due to the fact that the underlying density functions, which are required, for example, by Bayesian classification, are not required for classification by ANN.

Artificial Neural Networks also have the potential for parallel processing. This is a desirable characteristic since pattern classification can be computationally expensive.

ANN have shown to be a promising technique in the field of pattern classification. A considerable amount of research is currently revolving around their use in this field. A description of the technical details and architectures of ANN is presented in Chapter 3.

Chapter 3

Neural Networks

3.1 Biological Background

Knowledge of the biological origins of Artificial Neural Networks is beneficial in developing an understanding of the how one works and why much effort has been devoted to implement them successfully.

A simple neuron in the human brain is a nerve cell with all of its processes. The cell body of the neuron contains the nucleus. Leading into the cell body are one or more dendrites. Dendrites are branched parts of the nerve cell that conduct impulses toward the nucleus, while axons are nerve cell branches that conduct impulses away from the cell body. Bundles of these simple neurons form nerve structures. The point at which two neurons connect or more specifically where the axons of one neuron come into close proximity with the dendrites of another neuron is referred to as a synapse(Figure 3.1). Impulses generated by the first neuron may initiate an impulse in the second neuron if the end of the second nerve fiber is stimulated to some point

above its sensitivity threshold level. When this impulse reaches the termination of the nerve fiber, it may induce an impulse in another nerve cell. This process eventually results in some physiological activity or inhibition.

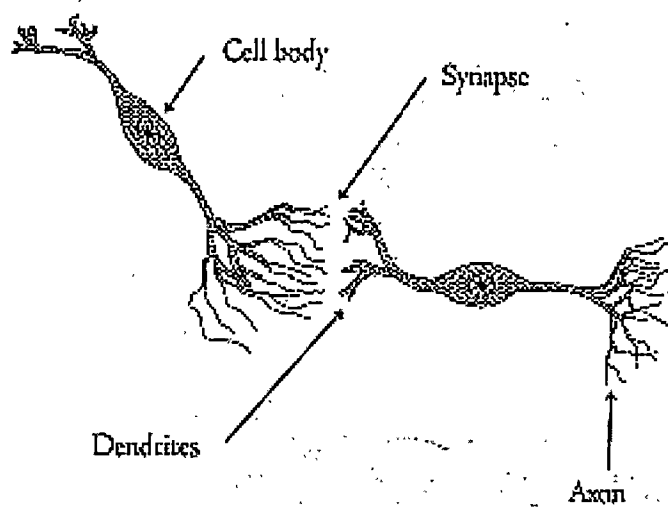


Figure 3.1: Simple human neurons.

The synaptic activity requires further explanation. The signals that come into different synapses are “weighted”. According to the strength of each synapse, some signals will be stronger than others will. Signals may be either of the excitatory (positive) or inhibitory (negative) nature. The effects of the weighted inputs are summed. If the sum of these weighted inputs is equal to or greater than the threshold for the given neuron, the neuron will activate or fire (give output). However, if the opposite holds true the neuron will produce no activation or output. The transmission of these signals may be altered by activity in the nervous system [19]. Certain events create a resistance to the passage of impulses from one neuron to another, while other events that may increase firing activity. This ability to adjust the signals is a

mechanism by which the neuron learns.

3.2 Artificial Neural Networks

Artificial neural networks(ANN), which are based loosely on biological neural networks consist of many simple processing elements(PE) or nodes, that represent biological neurons. The PE's handle several basic functions which include evaluating input signals and determining the strength of each one, calculating the total combined inputs, comparing the total to some specified threshold level and finally determining what the output should be.

These processing elements are connected with each other by weighted interconnections. Weights are adaptive coefficients within the network that are applied to each input signal. Once the inputs to neurons have been weighted and summed, the summation is presented to an activation function. The selection of an appropriate activation function is largely based on the architecture of the ANN being implemented. Output of the overall network is based on the output of these activation functions.

Artificial neural networks possess the ability to "learn." The ability to change weights allows that ANN to modify its behavior in response to inputs. How the ANN "learns" is determined by its learning rule. Generally there are two types of learning: supervised and unsupervised. Supervised learning ANN use a priori knowledge of the desired output for each training input to learn. The input is presented several times to the network. The network outputs an anticipated response, which is compared to the given output. Weights are then adjusted as specified in response to the results of the comparison. Contrarily, unsupervised networks must use self-organizing algorithms to

capture the most salient features in the input space to determine the correct output.

The number of processing elements and how they are arranged, the activation function, and the learning function determine the architecture of the network. There are several architectures that are suitable for the purposes of classification: Multi-Layer Feed Forward, Self Organizing (Vector Quantizer), Piecewise Linear Classifier, and variations of these.

3.3 Architectures and Learning Algorithms

3.3.1 The Perceptron

The Perceptron, one of the most simple and least sophisticated examples of ANN and pattern classifiers, was developed in 1958 by Frank Rosenblatt. A single node perceptron (Figure 3.2) performs classification by computing the weighted sums of the inputs, subtracting a threshold, and passing this result through a hard limiting non-linearity. The output is designated as being one of two classes. An output of 1 indicates class A and output of -1 indicates class B [12, page 58]. This is done by using a step function as the transfer function. The perceptron forms two decision regions separated by a straight line (Figure 3.3). When the input vectors have more than 2 dimensions, the separating line becomes a plane or hyperplane. The equation of the straight line is dependent upon the connection weights and threshold. Perceptrons are usually implemented on two-category linearly separable cases.

The weights on the perceptron are updated by the "perceptron" learning rule (PLR) or gradient descent techniques (GDT). Let y equal the output of the network

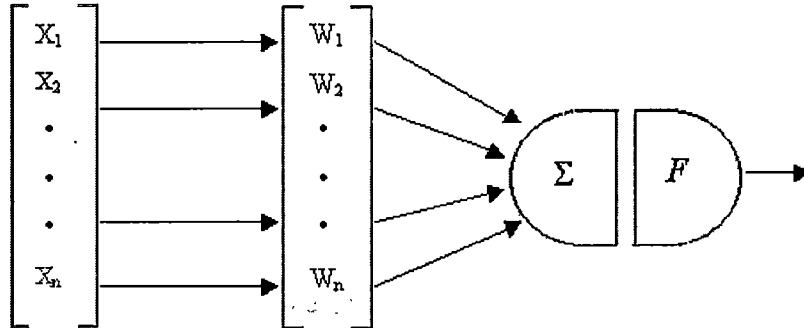


Figure 3.2: Perceptron

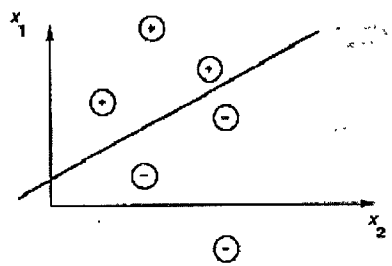


Figure 3.3: Linearly separated classes.

and $y^{desired}$ equal the desired value for the output. The PLR updates the weights according to:

$$\vec{w} = \vec{w} - \vec{w}(y - y^{desired}) * \vec{x} \text{ where } x \text{ is the input vector.} \quad (3.1)$$

Gradient Descent Techniques make changes according to the amount of error. The surface of the error function is a bowl. GDT moves down the surface of this bowl to find the bottom. Once this bottom is reached, the overall error of the network should be at its minimum.

A perceptron using the PLR has severe restrictions. The class of tasks for which it suited is limited to those, which can be solved in a linearly separable manner. While the linear separability of the system does guarantee that the training algorithm will converge to an optimal solution, the number of tasks that can be solved using this network is minute [15, pp. 31–33]. This is not the case for non-linear systems such as multiple layer networks.

3.3.2 Multi-layer Feed Forward Network and Backpropagation

The Multi-layer Feed Forward Network architecture is an extension of the single layer Perceptron. It is one of the most popular and versatile forms of neural classifiers. MLFN in their appropriate form have proven to be universal classifiers capable of forming decision regions of arbitrary complexity [20, page 73].

The Multi-layer Feed Forward Network consists of three or more layers of neurons: an input layer, one or more hidden layers and an output layer. Each of the hidden layers receives its input from the layer directly preceding and sends their output

to the units in the layer directly following it. No interconnections exist within a layer. The output of each hidden unit is a function of the sum of its weighted inputs and an optional bias term. This bias term is included to move the discriminating hyperplane(s) away from the origin. Output from one hidden layer is presented as input to the next layer.

Multiple layer networks can overcome many of the restrictions limiting the single layer perceptron. However, the introduction of a new layer presents the problem of adaption of the weights leading into that layer. The adaptations on the weights connected to the output layer are a function of the error. Since, there are no supervised training values for the hidden layer activations, there is no defined error associated directly with the hidden layer. The backpropagation learning algorithm is a solution to this problem and provides a means of updating the weights between the hidden layer(s) and preceding layer(s). The idea is that the errors from the output layer are back-propagated to the units in the hidden layer(s) to determine the error in the units of the hidden layer.

Define the following to represent a MLFN in Figure 3.4 with one hidden layer:

1. $\vec{x} = \{x_1, x_2, x_3, \dots, x_n\}$ = the input vector having n input features
2. y_j = the j^{th} unit in m hidden layer units
3. $\vec{v}_j = \{v_{j0}, v_{j1}, v_{j2}, \dots, v_{jn}\}$ = the weight vector from the input vector into node j in the hidden layer
4. $y_j^{out} = f_j(\vec{x} \cdot \vec{v}_j)$ = the result of the non linear activation function f_j^{in} of hidden unit j

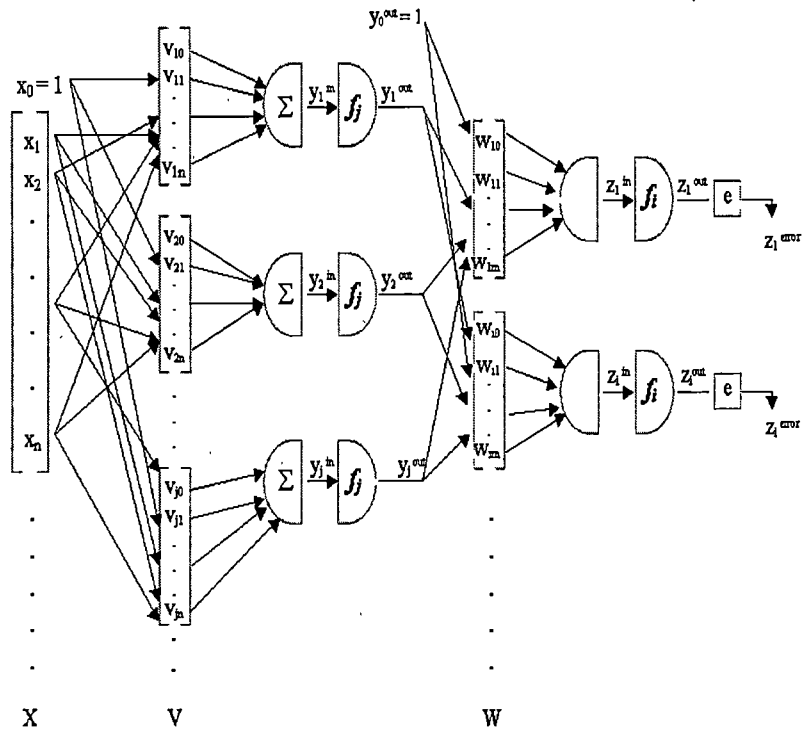


Figure 3.4: Multilayer feed forward network.

5. $\vec{w}_i = \{w_{i0}, w_{i1}, w_{i2}, \dots, w_{im}\}$ = the weight vector from the output vector of the hidden layer into node i on the output layer
6. z_i = the i^{th} output unit
7. $z_i^{out} = f_i(\vec{w}_i \cdot \vec{y}^{out})$ = the result of the activation function of output unit i
8. z_i^{error} = the error associated with output unit i .

Weights are adjusted by:

$$\Delta w_{ij} = -\alpha \frac{\partial (z_i^{error})^2}{\partial w_{ij}} \text{ where } z_i^{error} = z_i^{out} - z_i^{desired}, \quad (3.2)$$

where α is the learning rate constant and $z_i^{desired}$ is the desired output value at output unit i .

Let z_{in} be the dot product of the weights on the outputs of the hidden units:

$$z_i^{in} = \vec{w}_i \cdot \vec{y}^{out}. \quad (3.3)$$

Then we can write:

$$\frac{\partial (z_i^{error})^2}{\partial w_j} = \frac{\partial (z_i^{error})^2}{\partial z_i^{out}} \frac{\partial z_i^{out}}{\partial z_i^{in}} \frac{\partial z_i^{in}}{\partial w_{ij}}. \quad (3.4)$$

With further simplification, equation 3.4 can be rewritten as:

$$\frac{\partial (z_i^{error})^2}{\partial w_j} = \frac{\partial e^2(z_i^{out})}{\partial z_i^{out}} \frac{\partial f_i(z_i^{in})}{\partial z_i^{in}} \frac{\partial (\vec{w}_i \cdot \vec{y}^{out})}{\partial w_{ij}} \text{ or} \quad (3.5)$$

$$\frac{\partial (z_i^{error})^2}{\partial w_j} = \frac{\partial e^2(z_i^{out})}{\partial z_i^{out}} \frac{\partial f_i(z_i^{in})}{\partial z_i^{in}} \frac{\partial (w_{i1}y_1^{out} + w_{i2}y_2^{out} + \dots + w_{ij}y_j^{out} + \dots)}{\partial w_{ij}} \quad (3.6)$$

$$\text{where } e(z^{out}) = z^{out} - z^{desired}$$

By differentiating equation 3.6, the following is obtained:

$$\frac{\partial (z_i^{error})^2}{\partial w_j} = 2e(z_i^{out})e'(z_i^{out})f'_i(z_i^{in})(y_j^{out}) \quad (3.7)$$

or

$$\frac{\partial (z_i^{error})^2}{\partial w_j} = 2(z_i^{error})f'_i(z_i^{in})(y_j^{out}) \quad (3.8)$$

Define:

$$z_i^\epsilon = z_i^{error} f'_i(z_i^{in}) \quad (3.9)$$

and an update rule has been obtained. This update rule results in a gradient descent on the error surface if the weight changes are made according to:

$$\Delta w_{ij} = -\alpha z_i^\epsilon y_j^{out}. \quad (3.10)$$

It seems simple enough to calculate the change in weights that correspond to the output layer. However, it is not obvious what effect units in the hidden layer have on the output error of the network, due to a lack of supervised information for the hidden layer. The error in this layer, however, can be computed by measuring it as a function of the net inputs from the hidden to the output layer. The chain rule is used to calculate this error. The chain rule distributes the error of an output unit to all the hidden units that are connected to it.

The change in the weights corresponding to the hidden layer is defined as:

$$\Delta v_{jk} = -\alpha \frac{\partial \sum_i (z_i^{error})^2}{\partial v_{jk}} = -\alpha \sum_i \frac{\partial (z_i^{error})^2}{\partial v_{jk}}. \quad (3.11)$$

Let y_{in} equal the dot product of the hidden layer weight vector and the input vector, then:

$$\frac{\sum_i \partial (z_i^{error})^2}{\partial v_{jk}} = \sum_i \left(\frac{\partial (z_i^{error})^2}{\partial z_i^{out}} \frac{\partial z_i^{out}}{\partial z_i^{in}} \frac{\partial z_i^{in}}{\partial y_j^{out}} \frac{\partial y_j^{out}}{\partial y_j^{in}} \frac{\partial y_j^{in}}{\partial v_{jk}} \right). \quad (3.12)$$

Simplifying equation 3.12 yields:

$$\frac{\sum_i \partial (z_i^{error})^2}{\partial v_{jk}} = \sum_i \left(\frac{\partial e^2(z_i^{out})}{\partial z_i^{out}} \frac{\partial f_i(z_i^{in})}{\partial z_i^{in}} \right) \quad (3.13)$$

$$\left(\frac{\partial (w_{i0}y_0^{out} + w_{i1}y_1^{out} + \dots + w_{ij}y_j^{out} + \dots)}{\partial y_j^{out}} \frac{\partial f_j(y_j^{in})}{\partial y_{in}} \frac{\partial (v_{j0}x_0 + v_{j1}x_1 + \dots + v_{jk}x_k + \dots)}{\partial v_{jk}} \right)$$

Differentiating equation 3.13 gives:

$$\frac{\sum_i \partial (z_i^{error})^2}{\partial v_{jk}} = \sum_i 2e(z_i^{out})e'(z_i^{out})f'_i(z_i^{in})w_{ij}f'_j(y_j^{in})x_k = \sum_i 2e(z_i^{out})f'_i(z_i^{in})w_{ij}f'_j(y_j^{in})x_k \quad (3.14)$$

Define:

$$z_i^e = 2e(z_i^{out})f'_i(z_i^{in}) \quad (3.15)$$

then the following update rule has been obtained for the weights on the hidden layer:

$$\Delta v_{jk} = -\alpha \sum_i z_i^\xi w_{ij} f_j'(y_j^{in}) x_k \quad (3.16)$$

Theoretically, changes should be made to weights only after a full set of training patterns have been presented (i.e. at the end of a training epoch). In practice, however, the learning rule is applied to the weight after each training example.

Backpropagation can be applied to networks with any number of hidden layers. However, only one layer of hidden units usually suffices to approximate any function with finitely many discontinuities to arbitrary precision [15, page 33]. This is provided that the activation functions of the hidden units are non-linear. Most applications of backpropagation involve a three layer feed forward network using a sigmoid activation function for the units in the hidden layer.

Much success has been found with the back propagation learning technique. There are disadvantages associated with the algorithm. The learning process can be quite cumbersome largely due to the selection of an improper learning rate. If the learning rate is chosen too small, convergence is slow. If it is set too large, the network may oscillate around the solution.

Training failures often result from local minima. The error surface of a neural network contains many hills and valleys. Because the weights are moving down these hills and valleys, it is possible that the network may become trapped in a local minimum even if there is a deeper minimum in close proximity.

Another of complication is the selection of the number of hidden units. If too many hidden units are used, the network may begin to memorize inputs and will fit exactly with the training examples. This is known as "grandmothering" and results

in the network being incapable of generalizing for other input sets. This could also result in the network becoming trapped in local minima.

Ideally, the number of hidden units should be chosen to train the network to generalize correctly. Input values that have not been presented as training examples should be mapped to the correct output value.

3.3.3 Unsupervised Vector Quantizer

The Vector Quantizer (VQ) (Figure 3.5) is an unsupervised neural network architecture invented by T. Kohonen. The VQ learns by way of competitive learning.

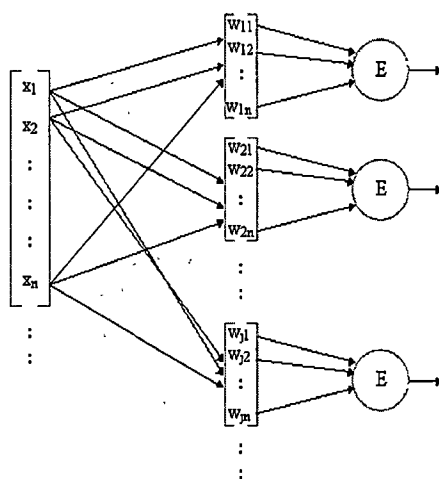


Figure 3.5: Vector quantizer.

Competitive learning is a “competition” among the output layer units to respond to the given input. The network reduces the input into some n clusters that describe redundancies in the data. The unit whose center vector is the closest to the input sample usually in terms of its Euclidean distance is declared “winner”.

In the VQ, there is a layer of input units and a layer of output units. The number of

output units usually represents the number of classes for a given problem. The input units are directly connected to the output units. In one pass through the network, a training example is presented to each output unit. The Euclidean distance between the weight vector (which represents the center of the cluster) and the input vector is computed for each output unit. The output unit obtaining the smallest distance is said to be the winner. Thus, the winning unit is the weight vector which lies closest to the input vector.

Define \vec{w}_j to be the center of unit j of the output layer. Then, the weights on the winning unit are updated by the following learning rule:

$$\vec{w}_j = \vec{w}_j + \alpha(\vec{x} - \vec{w}_j) \text{ where } \alpha \text{ is the learning rate .} \quad (3.17)$$

The learning rate decreases linearly in successive passes over the training data.

3.3.4 Supervised Learning Vector Quantizer

The supervised learning vector quantizer adapted by Kohonen from the unsupervised vector quantizer algorithm is much better suited for applications involving classification [20]. The learning vector quantizer is nothing more than a supervised learning version of the vector quantizer. For each tuple of input presented to the network, there is a corresponding desired classification (output). Like the original VQ algorithm, input tuples are presented to the network, the Euclidean distance between the output units' weights and input vector is calculated. The unit with the smallest distance is chosen as the winner.

Unlike VQ, the output units are labeled as being one of a given set of predefined classifications. The weights on the output units are modified in a similar manner to

the VQ, but now consider the classification (label) of the output unit. Referring again to Figure 3.5, let c_j be the classification label of unit j and $c^{desired}$ be the classification of input vector \vec{x} , then:

$$\vec{w}_j = \vec{w}_j + \alpha(\vec{x} - \vec{w}_j) \text{ if } c_j = c^{desired} \quad (3.18)$$

and

$$\vec{w}_j = \vec{w}_j - \alpha(\vec{x} - \vec{w}_j) \text{ if } c_j \neq c^{desired} \quad (3.19)$$

are the update rules for the weights. This particular training method moves the weight vector of the correctly-classified unit closer to the input vector and moves it away if the unit misclassifies. The learning rate is adjusted in the same manner as the original VQ algorithm.

3.3.5 LVQ2 Algorithm

The LVQ2 algorithm is an improvement of the learning vector quantizer algorithm. The LVQ2 algorithm uses gradient descent to reduce classification error. Weight adaptations can only take place if the following conditions hold true:

1. The closest weight vector W_1 is not of the same class as the input vector, x_i .
2. The next closest vector W_2 is of the same class as the input vector, x_i , and
lastly
3. The input vector x_i must lie within a window defined near the midpoint of vectors W_1 and W_2 .

When these conditions are satisfied the incorrect weight vector W_1 is moved away from the input vector and correct vector W_2 is moved closer to the input.

The selection of the optimal window mentioned in (3) is done by calculating the relative distances d_i and d_j measured from the weight vectors W_1 and W_2 respectively. If the window is too narrow, the training will suffer from low statistical accuracy due to a reduced number of corrections [13]. The vector x_i lies in the window if:

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > 1 - \varepsilon \quad (3.20)$$

The value for ε is determined by the number of training examples.

3.4 Related Research

In 1992, Mohamed Kaseko successfully integrated conventional image processing techniques and artificial neural networks models [11]. His research focused on using ANN as pattern classifiers for image interpretation and classification. Two ANN models were investigated: the multi-layer feed-forward network and the two-stage piecewise linear neural classifier. The two stage piecewise linear neural classifier consists of two stages (1) competitive learning (unsupervised learning) phase to find the approximate location of weight vectors that represent each class and (2) using the vectors from stage 1 as initial vectors for the LVQ2 algorithm. The function of the second stage is to fine tune the position of the weight vectors to achieve more accurate classification results.

Approximately 250 images of asphalt concrete pavement were acquired for use from PASCO USA INC. The images had to be digitized and segmented. From the

segmented images, features were extracted. The features were extracted based on “projection histograms” showing the number of distressed pixels in the binary images. Because the histograms did not describe the position of the cracks relative to some origin, the image was subdivided into tiles. Feature parameters were extracted from the individual tiles.

The tiles were made small enough to minimize the chance of including more than one crack segment in each tile. This was done so that feature parameters were able to clearly distinguish between the different orientations of the crack segments. In Kaseko’s research, the image was subdivided into 256 tiles, resulting in 32×29 pixel-sized tiles.

The parameters are computed based on the four directions of the projections. The directions considered are transverse, longitudinal and the two diagonal directions. The following parameters are computed based on the histograms:

1. x_m = the relative number of distressed pixels as a proportion of the total number pixels in the tile;
2. v_{tr} = variance of the number of distressed pixels per line in the transverse direction;
3. v_{ln} = variance of number of distressed pixels per line in a longitudinal direction;
4. v_{d1} = variance of number of distressed pixels per line in a diagonal direction;
5. v_{d2} = variance of number of distressed pixels per line in the diagonal direction;
6. r_{tr} = the mean number of “runs”, i.e., uninterrupted sequence of distressed pixels, in the transverse direction;

7. r_{ln} = the mean number of "runs" in the longitudinal direction;
8. r_{d1} = the mean number of "runs" in the diagonal direction;
9. r_{d2} = the mean number of "runs" in the other diagonal direction;
10. l_{tr} = the projected crack length in the transverse direction in proportion to the length of the tile;
11. l_{ln} = the projected crack length in the longitudinal direction in proportion to the width of the tile;
12. l_{d1} = the projected crack length in a diagonal direction in proportion to the length of diagonal;
13. l_{d2} = the projected crack length in the other diagonal direction in proportion to the length of the diagonal.

Parameters 2 through 5 provide information on the general orientation of crack segments in the tiles. According to Kaseko, the variance will generally be the highest in the direction of the cracking, as illustrated in Figure 3.6. Parameters 6 through 9 provide information of the number of crack segments in the each tile. These parameters, together with parameters 10 through 13 are used for image classification and computation of crack severities and extents.

After the image had been segmented and the features had been extracted, the next job was tile classification. In this stage, the orientation of any crack segments that existed was determined. The first five feature parameters were used as input to the neural network. Each of the tiles was classified into one of five categories:

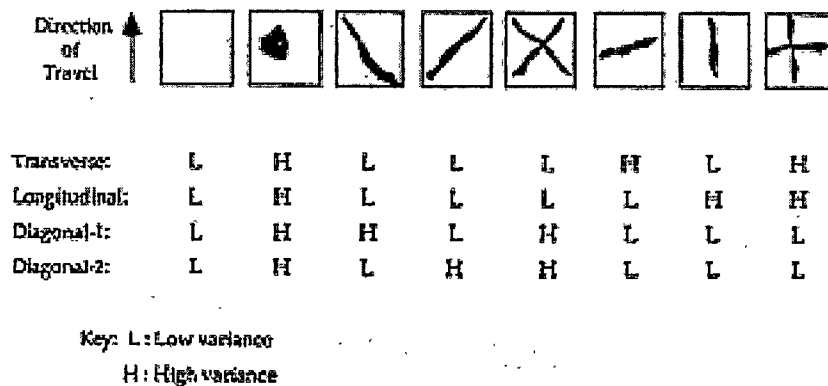


Figure 3.6: Variances in orientations of crack segments.

1. No cracking,
2. Transverse,
3. Longitudinal,
4. Diagonal and
5. "Combination" cracking.

Combination cracking was used to describe a tile that has two or more crack segments with different orientations.

The Multi-Layer feed forward network had five input units (corresponding to five feature parameters) and five output units corresponding to five classes. The data set was generated from 20 images. The number of hidden units was chosen to be five. This was selected after repeated tests. Five hidden units gave the best results in terms of minimizing the sum of square error.

The training set achieved 96 percent classification accuracy; while, the test set

achieved 93 percent accuracy. Traverse and Longitudinal cracks were the most distinguishable, as to be expected. Most misclassification occurred during the classification of diagonal and combination cracking.

The best results for the 2-stage piecewise linear neural network were obtained using 2 units in the competitive learning and the LVQ phase. The 2-stage PLN outperformed the MLFN, but not by very much. The results were 95.7 and 94 percent accurate for training and testing sets respectively.

The output from the tile classification was used as input for the image classification phases. Each 512×464 image is classified by the predominant crack type found in the image. The feature parameters for this phase consist of a two dimensional matrix. The elements of the matrix are based on the output of the tile classification stage. The input parameters are also based on parameters 6 through 13 computed in the feature extraction stage. These parameters were used to classify the image as one of five type distress types:

1. Transverse
2. Longitudinal
3. Alligator-1
4. Alligator-2
5. Block

Again both architectures were used. Sixty-four (64) input units, 3 hidden units, and 5 output units were used in the MLFN. This resulted in classification results of 93.3 and 85.7 percent for training and testing respectively.

The same PLNC used for the tile classification, with the exception that 64 input units were used, was implemented for this phase. This achieved 90.0 and 71.4 percent for training and testing respectively.

3.5 Technical Plan

There are 4 major steps in the proposed technical plan. They are as follows:

1. calculation of crack segment descriptors,
2. connection/Growing of the crack segments,
3. recursive division of images into various block sizes,
4. classification of distress types.

The crack segment descriptors are basic parameters that describe the segments extracted during feature extraction. These will be used to determine the proper growing and connecting of the segments and also later on in the classification stage. Growing/connecting refers to joining segments that were separated during the thinning process. After the final set of crack segments is obtained from step 3, the image is recursively divided into smaller and smaller blocks. This stops at some minimum block size. This step can be thought as forming a quad tree of the image. The average of the crack segment descriptors of the crack segments found in each of the above-mentioned blocks is used as input to the classifier. In step 4, each block will be classified by a pattern classifier, more specifically an artificial neural network.

Chapter 4

Technical Implementation

4.1 The Data

Highway images used to obtain data for this research were supplied by the Connecticut Department of Transportation (CONN DOT). The images were recorded by the ARAN System. The VGA output of the pavement cameras viewing have a resolution of 640×480 which is equivalent to approximately 3.125 mm^2 per pixel.

A total of nine images were used for training and testing purposes; 5 for training and 4 for testing. The images were selected based on how well they represented the different distress types to be classified. The distress types focused on were longitudinal, transverse, fatigue and block cracking. Also images were selected on the basis of how well the distresses showed up. Some of the images that were provided by CONN DOT did not show the distresses well due to initial imaging/processing irregularities. There were also no images that adequately represented fatigue cracking. Diagonal cracks were used to represent fatigue cracking since fatigue cracking consists

or many diagonal cracks [14]. Images that were damaged or had uneven lighting were automatically eliminated as candidates.

Images were approximately 2.5m x 9.5m. representing approximately 10 meters of single lane highway pavement.

To extract features needed for the classification, the images first were segmented. The segmentation technique used for the purposes of this research was the effort of a colleague's master thesis research. A general discussion of it follows. It should be noted that this description is not technically thorough; a more technical description can be found in [7].

4.2 Segmentation Technique

Pavement images are noisy and must be enhanced. Flexible pavement consists of particles with varying reflectivity and this noise often shows up as light areas. To eliminate the light areas, the image was clipped at its median value. The median value of the pixels in the image was calculated. Every pixel image above this median value was replaced by the median value.

Horizontal and vertical enhancement was performed using directionally biased median filters. This was done to enhance linear features in the image. Segmentation was then performed separately on these newly enhanced vertically and horizontally enhanced images with each segmentation image later combined to form one segmentation image.

The images were divided into tiles and a threshold was calculated for each tile in order to segment them. Supervised training was selected to generate the segmentation

parameters necessary to calculate this threshold. Supervised training involved using a human interpreter to select areas in a training image to delineate distressed regions. A binary subimage was generated from the ground truth. An optimum threshold value was determined for the ground truth tile.

Once the optimal threshold for these tiles was selected, estimators that approximated the optimal threshold without the use of the ground truth were obtained. This was done by calculating statistical values for each tile such as the minimum, maximum, and mean. These can be compared with the optimal threshold in order to estimate a tile threshold value.

An all-possible regression analysis is performed on the statistical data for the tiles. This is done to determine the best combination of estimators to predict the optimum threshold value. Once a proper combination is determined, a multiple regression analysis is performed to obtain values for the estimators, which form the independent variables in the regression equation.

The segmentation resulting from the threshold calculated with these parameters is still noisy. An optimization technique is performed on the parameters to further minimize the threshold value while still maintaining an optimal value.

Once the image has been segmented, it is thinned. Areas determined to be crack segments are reduced to a single pixel line width. The centers of the crack segments are found and the segments are eroded to the central line of the segment. After this, the newly thinned segment is traversed and at every n pixels a (x,y) point is recorded. Thus, the crack segments are reduced to a series of (x,y) points for each crack segment according to where they are located in the image. The lower left-hand corner is the origin and point coordinates are calculated according to this.

4.3 Preprocessing

After completing the segmentation of Section 4.2, the data for each crack segment consists of pairs of (x, y) coordinates in two-dimensional space. When visualized, these descriptors provide much information to a human observer as to the nature of a crack. However, if presented to an Artificial Neural Network in this form, the data would mean very little. Most of the cracks in this research can lie almost anywhere in the image. Thus, a description of the whereabouts of the crack would not supply enough information to allow an ANN to yield accurate results.

However, it can be seen from examining the data, that certain parameters that define different crack types can be calculated from this data with simple algorithms. These parameters, more specifically: the variance from linear, orientation, length and width, should provide enough information to the ANN to allow it to distinguish between different crack types [17]. Most crack types have distinguishable characteristics that can be described by one or a combination of these parameters.

4.3.1 Calculating Initial Parameters

The following parameters must be calculated for each crack segment found in the image:

1. variance from linear(VFL),
2. orientation,
3. length,
4. and average width.

These parameters are calculated initially for later use by the preprocessing algorithm.

VFL

The VFL describes how straight the crack segment is and is used for connecting/growing. This is discussed later in this paper. The VFL is the ratio $\frac{\text{point to point path length}}{\text{endpoint to endpoint distance}}$. It is calculated by summing the distance from point to point within the crack segment and dividing it by the Euclidean distance between the two crack endpoints. Let x_1 and y_1 be the coordinates of the first endpoint of any given segment and x_n and y_n be the coordinates of the last endpoint in the segment where there are n nodes(points), then:

$$dx = x_k - x_{k+1} \text{ and } dy = y_k - y_{k+1} \text{ for all } n \text{ points} \quad (4.1)$$

The VFL is now defined as:

$$VFL = \frac{\sum_{k=1}^n \sqrt{dx_k^2 + dy_k^2}}{\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}} \quad (4.2)$$

There is a special case to be considered when calculating the VFL. If the total length from endpoint to endpoint distance is zero, the shape of the segment being investigated approximates a closed path. In this unusual circumstance the VFL is assigned a negative constant less than minus one. For the purpose of this research, the VFL for a closed path was set to -5 . By setting the VFL to -5 for example, a human investigating the data can easily detect the anomaly and determine the segment in question is a closed path. Using -1 for example, would blend in too much with the normal values of the VFL. Normal VFLs have been empirically determined to lie in the range of 1.00 to 1.1, where 1.0 is the straightest and 1.1 is considerably

wavy. By setting the VFL to -5 for example, a human investigating the data can easily detect the anomaly and determine the segment in question is a circle. While this does not occur much in this research, it could prove to be beneficial in future endeavors if highway distresses that have circular characteristics (e.g. potholes...) are added to the list of distress types to be classified.

Orientation

The orientation describes the angle the crack segment forms with respect to a line parallel to the centerline of pavement passing through the beginning of each crack segment. While examining crack types it is apparent that this is one of the most important features. Most cracks can be determined by simply observing the orientation. For example, the two most common cracks found in pavement, longitudinal and transverse, form approximately 0 and 90 degree angles respectively with reference to the pavement centerline. That is, they are parallel and perpendicular with the centerline. Other crack types, namely alligator and block cracking are a combination of these two types of cracks. Thus the orientation of a crack segment is crucial in determining crack types.

Again let (x_1, y_1) and (x_n, y_n) be the coordinates of the first and last endpoints respectively of a crack segment with n points. The orientation is found by:

$$\arctan \frac{y_1 - y_n}{x_1 - x_n}. \quad (4.3)$$

In the event that the difference between the x values of the two coordinates is equal to zero, the orientation is assigned a value of 90 degrees.

The orientation is mapped into the range of 0 and 90 degrees by reflection with

respect to the centerline. Initially crack segments may take on values between 0 and 180 degrees. Any value greater than 180 degrees was mirrored into the first quadrant by:

$$Orientation_{90} = 180 - Orientation \quad (4.4)$$

This is done to obtain meaningful averages. For example, during the averaging phase (which is to be discussed later), a transverse crack recorded at 175 degrees and another transverse crack recorded at 5 degrees will average to 90 degrees. This orientation does not reflect a transverse crack but a longitudinal crack.

This was also done to reduce the amount of variation recorded in the standard deviations calculated values that is used as input to the neural network. The major concern is to obtain the general orientation of the crack. The precise angle of the crack is not important.

Length

The length of the crack is another important variable to consider during classification. Random cracking consists of diminutive transverse and longitudinal cracks. Alligator cracks, while progressing to a higher severity, are composed of random cracks. Block cracking is composed of longer transverse and longitudinal cracks.

The length of the crack is found by taking the Euclidean distance from (x_1, y_1) to (x_n, y_n) of a crack segment.

$$Length = \sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2} \quad (4.5)$$

Average Width

The width of a crack is more used for determining the severity of a crack type than for classification. The average width of the crack segment is found by averaging the widths recorded at each point during the thinning process.

$$AverageWidth = \frac{\sum_{k=1}^n width_{p_n}}{n} \text{ where } p_n \text{ is the } nth \text{ point} \quad (4.6)$$

Making note of the width could also prove to be beneficial when this research incorporates crack width to complete extent and severity portion of the crack classification.

4.3.2 Growing/Connecting Cracks and Removing Spurs

Crack information may be lost or incomplete for a number of reasons. A crack may not be entirely distinguished during the segmentation phase. Cracks have the tendency to change density and width. The threshold determined for the segmentation of the pavement image may be set such that some of crack with a lower density may not be distinguished. For this reason there may be an interruption in the crack segment. Also for the specific purposes of this research, crack segments were also broken up during the image vectorization phase. While traversing the path of the segments, if an angle θ degrees was incurred, an endpoint was inserted at that point and a new crack segment was formed. Even so, not all cracks are contiguous initially, yet upon examination of the crack segments, it is intuitively obvious to the human observer that the segments form one crack. Separate segments in the data could describe a single noncontiguous crack and thus need to be "connected" or "grown" to form one crack segment

“Connecting” cracks is the joining of two or more crack segments that share common endpoints and have similar initial parameters. “Growing” refers to joining cracks that lie within a certain proximity of one another and also have similar initial parameters. The task of connecting/growing cracks is accomplished by comparing certain properties of the crack. Before any connecting/growing can take place, the segments are examined to identify some that may be “spurs”. Spurs are short segments sharing an endpoint with 2 or more other segments and are of a very small length (Figure 4.1). These spurious segments are a side effect of the thinning phase.

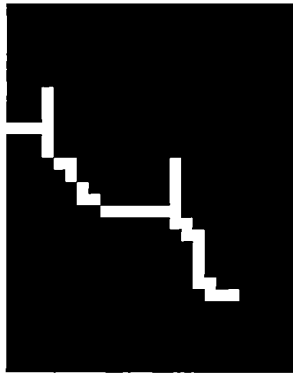


Figure 4.1: Shorter segments are spurs.

If any segment has a common endpoint with two or more segments, it is marked as a possible spur. Later on in the connecting/growing process, the length of the marked segment is compared against some maximum spur length. If the segment is less than this threshold and has not been connected with some other segment, it is removed from the data.

Once this has been determined, the distance separating each crack is examined. Define $S = \{S_1, S_2, S_3, S_4, \dots, S_n\}$ to be the set of n segments in a given image. Let d be the Euclidean distance between S_i and S_n and l_i and l_n be the lengths for S_i

and S_n respectively. Also let $L = l_i + l_n + d$. If the ratio of the d to L is less than 20 percent, the two segments are likely candidates for connection or growing to each other. It was empirically determined that the separating distance(d) should not make up more than 20 percent of the total segment length of the two segments once they have been connected. Also this distance, should not be more that 15 *cms*. This is to ensure that segments are properly connected.

All segment pairs that are candidates for connection or growing are recorded and tentatively joined. The tentatively connected segments form the set $TS = \{TS_1, TS_2, TS_3, \dots, TS_j, \dots\}$. The segment, TS_j having the smallest VFL is chosen as the "winning" candidate. However, this does not guarantee that any connecting/joining will occur. If the winning segment's VFL is less than some specified threshold, meaning the newly formed segment will not be too wavy, the two segments are permanently joined. Otherwise, the segments remain as they were. This is performed for each segment and its succeeding segments.

Segments are deleted from the data when necessary and new segments recorded. Parameters and information are updated. This entire process is performed until it is determined that no remaining segments can be joined. Using big O notation, the order of this entire process was $O(n^3)$.

If no remaining segments can be joined, the lengths of all segments present after the joining process are checked against some minimum length. The minimum length determined by tests was set at .63 *cms* for use in this research. Any segments shorter than this are eliminated from the data. This new set of segments serves as the data from which global input values used by the classifier are obtained.

4.3.3 Multiple Resolution Data

The principle purpose of this research is to identify, classify, and quantify distress types in images of highway pavement. This approach uses an ANN classifier. Information from the images must be presented to the classifier. In the preceding section, certain parameters that describe the crack segments in an image were extracted to obtain this information. It would be difficult to present each segment and its data to the classifier. Also, it is likely that the classifier would not produce accurate results using input in this form.

Rather than present the classifier with each segment's information, the input is aggregated over various block sizes. The average and standard deviation of all the segments' parameters is calculated and used as input. It is not expected that the averages computed over the entire image will lend insight into the distress types found in the entire image. However, examining different portions of the image and the information local to each portion could prove beneficial to classification. By subdividing the images into blocks, the distress types found in the image can be localized. Information aggregating the segments found in a given block can lead to the accurate classification of the distress type found there and ultimately the entire image. Recording the position of the block relative to the entire image and distress type prevalent can provide an easy route to calculating the extent and severity of the of distress types found in an image. This method allows examination of exactly where the distress types are positioned and what percentage of the highway they encompass.

The division of the image also provides additional data. Each block is treated as if were an image itself. This is very important, because it provides more training

examples for each crack type. This solves the problem of not having many training images. This also reduces the large data storage requirements of using many training images.

The original images were recursively divided into smaller blocks (Figure 4.2). The first set of divisions (level 1 divisions) divides the rectangle image vertically into three or four approximately square blocks based on the size of the image. The following divisions divide each of those blocks into four blocks. The twelve blocks that result from the division comprise the level 2 divisions. This is repeated for each block in level 2 leading to level 3. The divisions formed nearly square quadrants in each block in each level past level 1.

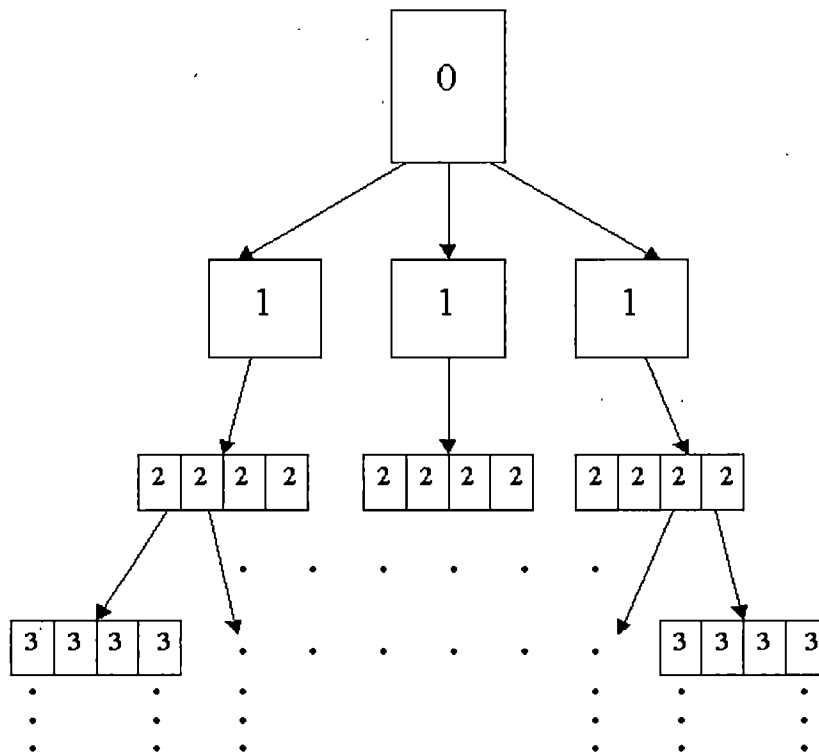


Figure 4.2: Division of the image. Level 0 is the entire image.

Any portion of a segment found in a block is considered local to that block. If the segment is not encapsulated entirely by the block, new endpoints must be inserted on the boundary(ies) the segment crosses. The segment is then bound by these new endpoints. A segment may cross the boundary(ies) of a block in several locations as shown in Figure 4.3.

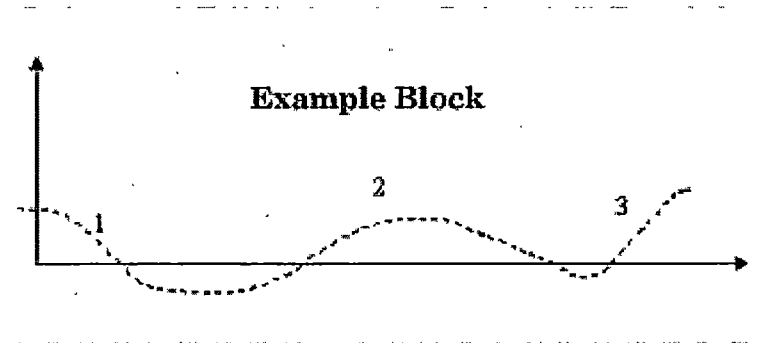


Figure 4.3: A crack segment that crosses boundaries of block.

If Figure 4.3 is used as an example, this segment would create three segments within the example block. The "initial" parameters for each of these new segments must be recalculated. Division occurs until information has been extracted from the smallest block allowed. The minimum block size chosen approximately $30 \times 30\text{cms}$. These dimensions were chosen because engineers and surveyors in the field usually do not consider information from areas of highway smaller than this.

Information extracted from each block is used as input to our classifier. The mean and standard deviation of the segments' parameters local to the block in question are calculated. For each block the following are obtained:

1. weighted average VFL of all segments,
2. standard deviation of VFL for all segments,

3. weighted average orientation of all segments,
4. standard deviation of orientation for all segments,
5. weighted average length of all segments,
6. standard deviation of length for all segments,
7. weighted average of the average width of all segments, and
8. standard deviation of the average width of all segments.

The average were weighted by the lengths of the segments local to the blocks. The longer the crack segment, the more influence it had on the average. This was performed because surveyors in the field pay more attention to longer cracks in their rating of the pavement. Weighting was carried out by the following means:

```

For(I=1 to number of segments)
    Find minimum length of all segments
For(I=1 to number of segments)
    Weight[I] = (length[I]/minimum length)+0.5
For(I=1 to number of segments){
    Average = Average + Weight[I]*input_parameter
    N=N+1
}
Weighted_Average=Average/N

For(I=1 to number of segments){

```

```

Std_Dev1 = (input_parameter-average)2
For(J=0 J<Weight[I])
    Std_Dev = Std_Dev + Std_Dev1
}
Std_Dev = sqrt(Std_Dev/N)

```

4.3.4 Ground Truthing

The input examples used for training must be assigned a classification. This is necessary because the ANN classifier learns in a supervised mode. The classification is obtained by means of human observation of the ground truth image from which the input was obtained. Training input examples are assigned a classification based on the area the observed example represents. This classification takes the form of an integer from 1 to 5, which represents the four types of distresses and a miscellaneous category:

1. Transverse
2. Longitudinal
3. Alligator
4. Block
5. Miscellaneous

This was accomplished by examining the image with a overlaid grid (Figure 4.4).

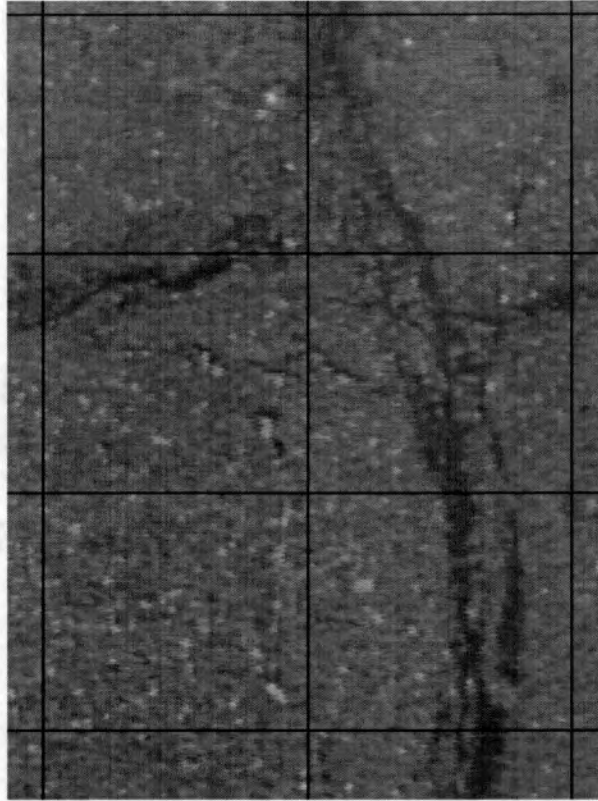


Figure 4.4: A portion of image with overlaid grid.

4.4 Classification

The Learning Vector Quantizer(LVQ) was selected as the classifier to be used in this research. It is one of the simplest architectures whose main purpose is to perform classification in the supervised mode. Because each distress type has certain characteristics in common as they pertain to the input parameters, clusters can easily be formed from the data. The LVQ is also a relatively simple architecture to understand and train. It requires fewer iterations to train than most architectures. The neural network classification was implemented using a Sparc work station. The software was provided by the University of Tennessee Space Institute. The software, titled Ebtide,

allowed the display up of to three dimensions in the input space of the data, as well as the training center nodes. Ebtide also provided a means of determining how well the neural network was converging to a correct classification [2].

The parameters selected as input to the classifier for each block discussed in section 4.3.1 were the average orientation and its corresponding standard deviation and the average length and its corresponding standard deviation. The average VFL, average width and their standard deviations were not used as input features. This is because the VFL for all the crack segments were between 1.0 and 1.06. When tested in the classifier, these values being so close, did not appear to lend discriminating information to the classifier. The width also does not appear to lend much information as to what type the sample belongs.

Certain parameters in the feature vectors were scaled. The values of the average orientation took the range of 0 to 90 degrees. The standard deviation also took the same range. It is not ideal that the deviation be that high, yet it is still possible. The average length and corresponding standard deviation took on the range from approximately .03m to 10m. The numbers are based on the smallest crack segment allowed and the length of the image used for input. Because there is a considerable difference between the values that the orientation variables can take and length variables, the length variables were scaled. They were scaled to lie between 0 and 100, approximately the range of the orientation variables. This was done by applying equation 4.7:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} 100 \quad (4.7)$$

where x_{min} and x_{max} represented the minimum and maximum values for feature x .

Scaling the data did help the performance of the classifier. However it proved to be more beneficial for analyzing the training process. Scaling allowed classifications to be observed as regions in the input space (Figures 4.5 and 4.6).

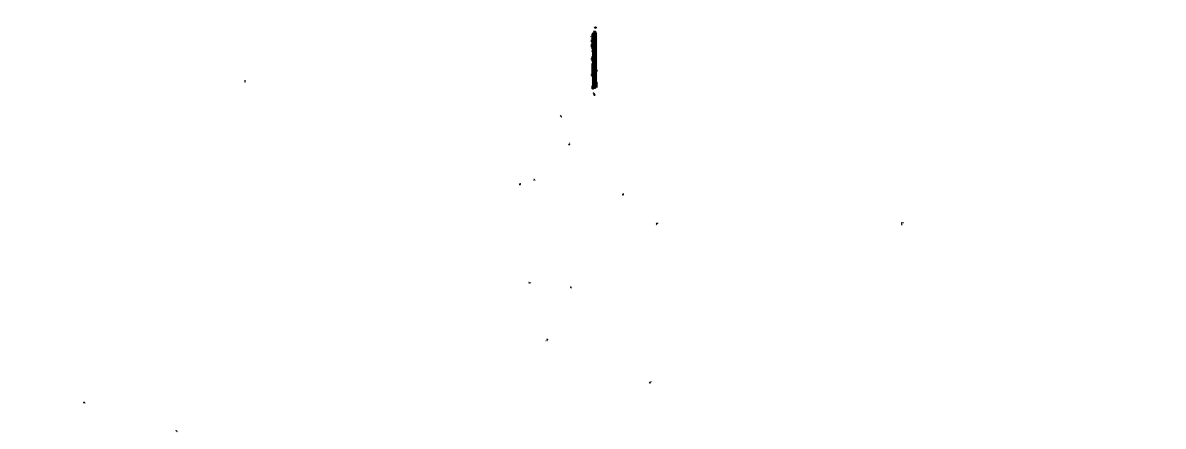


Figure 4.5: Unscaled data. It is difficult to determine decision regions in the data because it is so compact

Recalling from section 3.3.4, certain parameters must be initialized to allow the LVQ to train. These parameters such as the number of output nodes (center nodes for the purposes of this research) and learning rate constant were to be determined empirically. The initial center vectors were selected randomly, but due to poor performance of the classifier, they were changed to the mean values of their corresponding classes with the exception of those initial vectors that pertained to the miscellaneous crack type.

Miscellaneous cracks are short cracks that may take on the orientations of any of the other four classes. More specifically, they are cracks that are approximately 10 *cms* or less and are usually lone cracks. They are usually the result of some

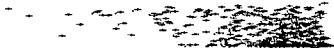


Figure 4.6: Scaled data from Figure 4.5. Decision regions can now be determined because the data is less compact.

random noise, improper connection, or the results of block lines chopping off the crack segments extremely close to their endpoints. This was included as a distress because all cracks had to be assigned to some class and these short cracks did not entirely fit the other classes.

It is true that severe cases of fatigue cracking are composed of short crack segment. However, the fact that a crack segment may be fatigue crack should be reflected in the standard deviation. The standard deviation of miscellaneous cracks is normally 0 or some very small number. These short lone cracks do not provide any useful information to the PCR. Because of this, there are only four centers assigned to the miscellaneous class. Each center is initialized to one of the other four classes.

There was also a small amount of noise injected into the initial centers to offset them from their means. This is performed by equation 4.8. The overall concept of assigning the initial center vectors to those of the means provides a reasonable initial state for training. The designer of the LVQ algorithm suggests this [22].

$$initial_center = input + \left(\frac{std_deviation}{n^{class}} \right) * (random_num - 0.5) \quad (4.8)$$

where n^{class} is equal to the number of examples in the class and

input is the input value with the specified class

Recalling from Chapter 3, the LVQ algorithm learns by attracting a center whose label correctly classifies the input and repelling a center whose label is incorrect. A repulsion factor, β was added to the architecture, so that a losing weight vector was updated with the following rule:

$$\vec{w}_j = \vec{w}_j - \beta * \alpha(\vec{x} - \vec{w}_j) \text{ if } c_j \neq c^{desired} \quad (4.9)$$

This appeared to improve the convergence of the algorithm.

Once the network had been trained a final set of center vectors were in place for unsupervised classification. Testing and final classification is accomplished by presenting new unclassified examples to the network.

A confidence test was added to the final classification phase. This was performed by placing a gaussian radial basis function (equation 4.10) around each center.

$$f(x) = e^{(-\frac{1}{2} \frac{d^2}{\sigma^2})} \quad (4.10)$$

Each unclassified input sample was presented to each center. The center obtaining the highest output from equation 4.10 was chosen as the winner and that input sample was assigned the classification of the winning center vector. The higher the confidence, the more likely the classification was correct and conversely. The gaussian radial basis function is symmetric about its center (Figure 4.7). It takes on its highest

value at the center. So, input samples that lie extremely close to the center will take on very high confidence values.

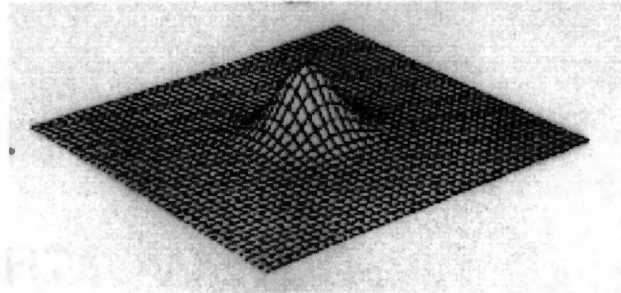


Figure 4.7: Gaussian Radial Basis Function

4.5 Algorithm

1. Calculate Initial Parameters

VFL, Orientation, Length, Width

2. Mark Potential Spurs

```
For(I=1 to number of segments){
```

```
  For(J=1 to number of segments){
```

```
    If(I != J){
```

```
      Determine if segment I and segment J have common endpoint
```

```
    }
```

```
  }
```

```
  If(segment I has 2 or more common endpoints)
```

```
    Mark as possible spur
```

3. Grow/Connect Crack Segments

```
Do{
  For(I=1 to number of segments){
    For(J=I+1 to number of segments-1){
      Find endpoints of segment I and segment J that produce
      Smallest separating distance(d)
      Ratio = length of segment I + length of segment J + d
      If(ratio < max_ratio and d < .15)
        Tentatively join segment I to segment J
    }
    For(J=I+1 to number of segments-1)
      Find pair of tentatively joined segments with smallest VFL
      This is "winner"
      If(VFL of winning segment pair < max_VFL){
        Permanently join segments
        Update parameters for newly formed segment
      }
  }until(no more segments can be joined)
  For(I=1 to number of segments){
    If(segment I not connected and marked as possible spur
      and < maximum spur length)
      Discard segment
  }
  For(I=1 to number of segments){
```

```
If(segment I < min_length)
```

```
  Discard segment
```

```
}
```

4. Create Multiple Resolution Data

```
Divide original image 3 or 4 times according to width
```

```
Determine if segments exist in each block
```

```
If(no segments)
```

```
  Discard block
```

```
If(segments cross boundaries of block){
```

```
  Insert endpoints at intersection
```

```
  Recalculate parameters for each segment in block
```

```
}
```

```
Calculate weighted mean and standard deviation of all segments
```

```
In each block
```

```
Do{
```

```
  For(each block obtained from previous division){
```

```
    Divide 4 times
```

```
    If(block < min_block_size)
```

```
      Stop
```

```
    Determine if segments exist in each block
```

```
    If(no segments)
```

```
      Discard block
```

```
    If(segments cross boundaries of block){
```

```
      Insert endpoints at intersection
```

Recalculate parameters for each segment in block

}

Calculate weighted mean and standard deviation of all segments

In each block

} until(min block size reached)

Write out mean and standard deviation for each block to a file

(This is input to neural network)

5. Ground truth Data

Assign classification to each block in the images according to the major distress prevalent

1 - Transverse

2 - Longitudinal

3 - Fatigue

4 - Block

5 - Miscellaneous

6. Scale length parameters of input set so values will be in range of 0-100

7. Train and test Supervised Learning Vector Quantizer Neural Network

Initialized centers to means of sample values of each class from

Supervised training data (For misc. crack type set each of 4

centers to one of the mean vectors of the other 4 crack types.)

Train

For(I=1 to number of training epochs){

Present each training example to each center(output) node

```

Take Euclidean distance of center node and input
Select node with smallest distance as "winner"
If(class of node = desired class)
    Move center closer to input sample
If(class of node != desired class)
    Move center away from input sample
}

Repeat until best accuracy achieved

```

Test

```

Present each unclassified input sample to each center
Choose class of the center node that receives highest confidence

```

Table 4.1 is a listing of the constants and their corresponding values used in this algorithm.

Table 4.1: Table of constant values used in algorithm.

max_ratio	0.20
max_VFL	1.06
max_spur_length	0.01
min_length	0.0635
min_block_size	$\approx 0.30 \times 0.30$

Chapter 5

Results

The success of any classification scheme is due largely in part to the segmentation of the image. Segmentation played a very crucial role in this research. Because of this, considerable attention was placed on improving the segmentation achieved. For example, Figure 5.1, an original image, is recognized to a human, as a well defined crack. However the segmentation algorithm as shown in Figure 5.2 did not pick up all of it.

Humans can determine even faint linear features and often interpret crack continuance where little or no image density evidence exists. In order for the segmentation algorithm to pick up the lighter cracks (Figure 5.3), the threshold had to be increased. But increasing the threshold increases the noise levels as is shown in the full image Figure 5.4. This noise is detrimental to the performance of the classifier.

After segmentation was optimized by the procedures previously discussed, the performance of the ANN was evaluated by comparing it to the performance of a human observer classifying the vectorized images that resulted from the segmentation.

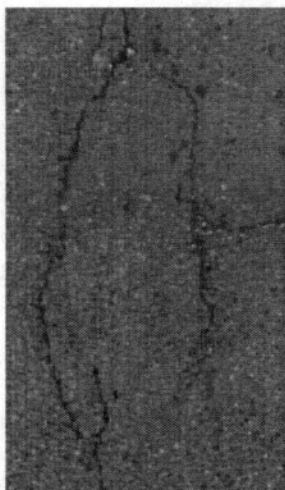


Figure 5.1: Portion of original image.

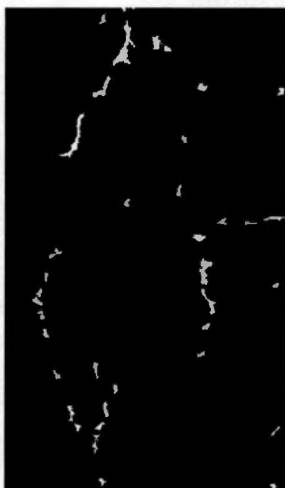


Figure 5.2: Segmentation of Figure 5.1. Note how that some of the crack does appear.

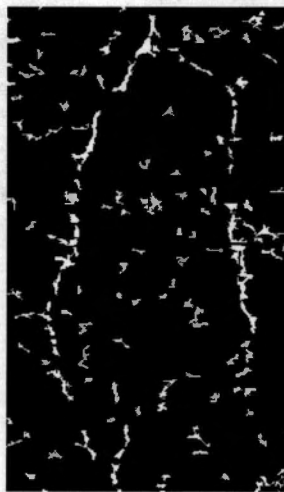


Figure 5.3: Segmentation of figure 5.1. More of the crack is connected, but the noise level is also increased.



Figure 5.4: Noisy segmentation shown for full image.

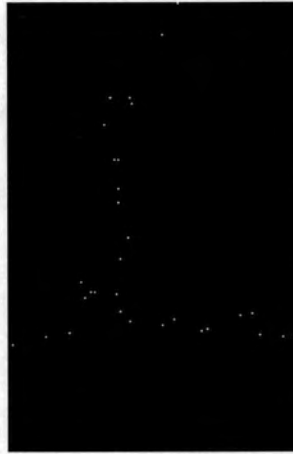


Figure 5.5: Portion of vectorized image used for human interpretation of distress types in the image.

The vectorized image was one in which the cracks were reduced to points as shown in partial image Figure 5.5. A human interpreter then used the samples classify crack types. Since input parameters for the neural networks were actually extracted from this image, a direct comparison between human interpretation and the neural network classifier could be made based on the same data. Therefore, information on how well the neural network was doing with the information presented to it could be obtained. Both the human interpreter and neural network were given partial data. The human interpreter did not have the benefit of the numeric segment descriptors. Both the human and the neural network did not have the benefit of original image. Training and testing of the neural network was performed on the data supplied by the ground truth of the original images. A discussion of the parameters necessary for this training and the results of the training follows.

The Supervised Learning Vector Quantizer requires that certain parameters be

set in order to achieve optimum classification accuracy. The proper number of center nodes, learning rate and number of epochs had to be determined. Combinations of these values were tested during the training phase. The selection of these parameters was based on the accuracy of final classification and a term called the Total Squared Quantization Error (TE). This error at any epoch i is defined as:

$$TE(i) = \sum_{k=1}^n (c_{winner}(i) - x_k)^2 \text{ where } n \text{ is total number of training examples. (5.1)}$$

The TE is not a true measure of the error of the LVQ because the LVQ's error is determined by how accurate the final output is. However, it did provide a means to determine if the network was converging while training. The set of parameters that produced the smallest classification error and allowed the network to converge at a steady rate was chosen.

Repeated training and testing of the network was performed in order to determine the proper number of centers. The longitudinal, transverse, fatigue, and block classes were represented by five centers. The miscellaneous class was represented by 4 centers. Figure 5.6 shows a plot of the accuracy as a function of the number of nodes.

Under normal circumstances, the number of centers would have a great effect on the accuracy of the network. Initializing too few centers can cause the network to underfit the data, and too many can cause the network to overfit. It can be seen that the number of center nodes does not have a great effect on the overall classification accuracy. However, upon close inspection of the individual classes, it is observed that the miscellaneous classification improves when either 1 center or 10 centers are used. To understand why this is so requires examination of the data.

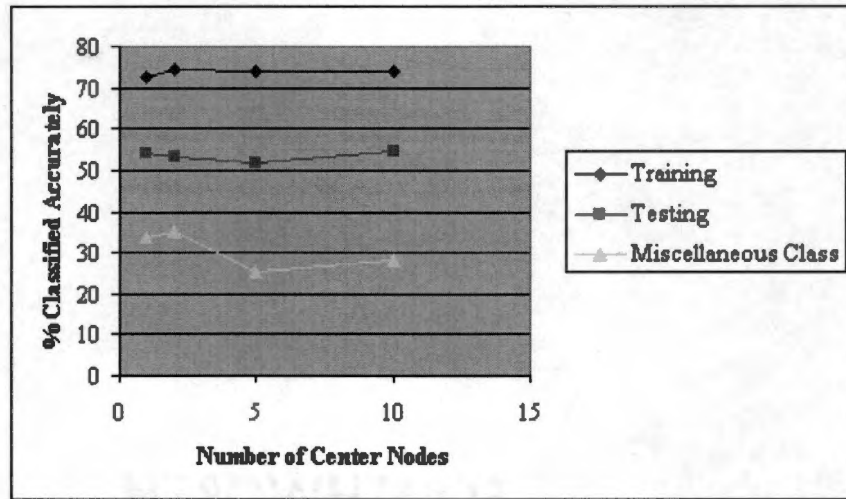


Figure 5.6: Plot of accuracy as function of the number of iterations used to train the neural network.

Miscellaneous cracks are small crack segments resulting from improper connecting of the crack segments (discussed in Section 4.3.2) or a block intersecting a crack segment near an endpoint of the segment. Their orientations will take on the values of the orientations of the other four classes. The lengths are ≈ 10 cms or less. Any segment ≈ 10 cms or less, except for those belonging to fatigue cracking, are assigned to the miscellaneous class.

The problem generally arises from the segmentation issue that was discussed previously in this section. A crack that shows up in the original image may not show up well in the segmentation. A human interpreter has no knowledge of this unless he is examining the segmented binary image. So, crack segments are being labeled as one of the other four classes when the actual calculated parameters really support the miscellaneous class. The results of this is “speckled” data. The term “speckled”

is used because the other classes are mixed into the miscellaneous class region in the length dimension of the input. Taking Figures 5.7 and 5.8 for example, samples in the length dimension of the examples in the transverse class should have a length of at least 6. However some fall below this number and lie in the miscellaneous class region (Figure 5.8). Because of this, a good decision region can not be formed for the miscellaneous class.

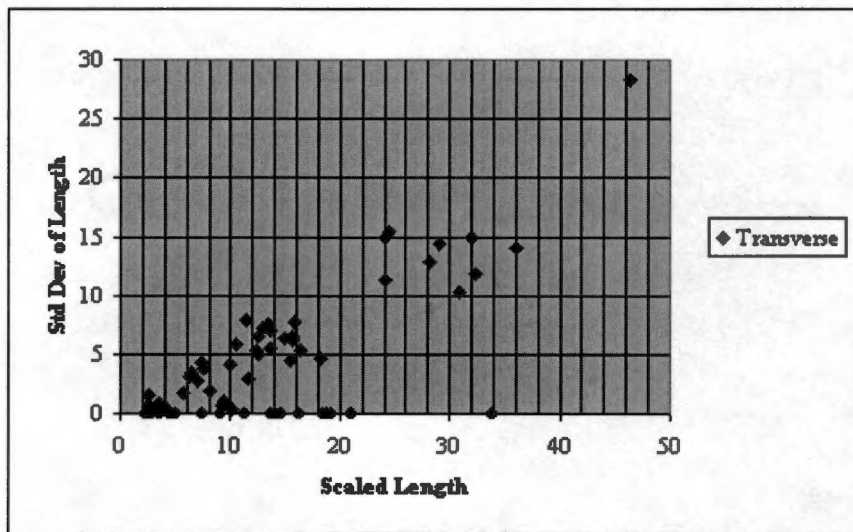


Figure 5.7: Plot of orientation vs. standard deviations of the lengths of the transverse crack type. Anything between 0 and 8 can be considered relatively small.

This does not allow the miscellaneous class to train well. The neural network trains by attracting and repelling the centers according to the accuracy of the classification. The miscellaneous centers are often repelled from their region. This is due to the fact that there are samples of other classification in their region that push them away. Even when the miscellaneous centers are initialized so that they lie directly in their region, they are repelled. So, when initializing the number of centers to 1 for each

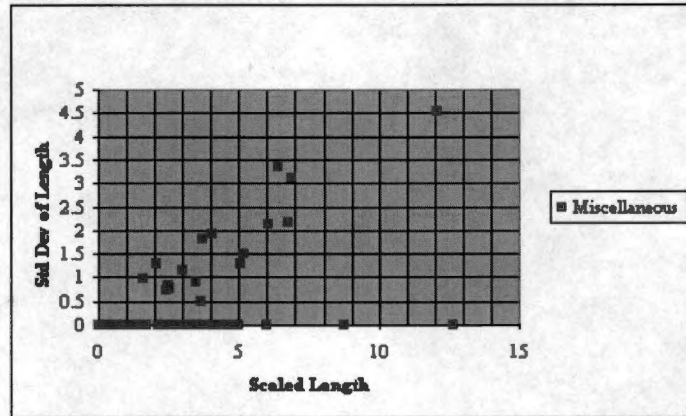


Figure 5.8: Plot of orientation vs. standard deviations of the lengths of the miscellaneous crack type. Anything between 0 and 8 can be considered relatively small.

of the other classes, the miscellaneous class has a greater chance of capturing more samples because it competes with fewer centers of the other classes. When initializing the number of centers to 10 for each of the other classes, the centers of the other classes sit right on top of each other and do not disperse. This allows some of the random centers namely, the one located near the transverse class region, to train better.

Five centers were allowed for each of the other four classes. This seemed to represent the dispersion and shape of the data(Figure 5.9).

The learning rate was chosen to be 0.01. A small learning rate was required because the training was very sensitive. Choosing too large of a training rate caused the centers to either oscillate in the different regions or to leave the regions altogether. This small learning rate allowed the centers to move into the ideal locations in a smooth and steady manner.

The number of epochs was chosen to be 1000. The nature of the LVQ allows it to converge rapidly. Figure 5.10 shows a plot of the TE as a function of the

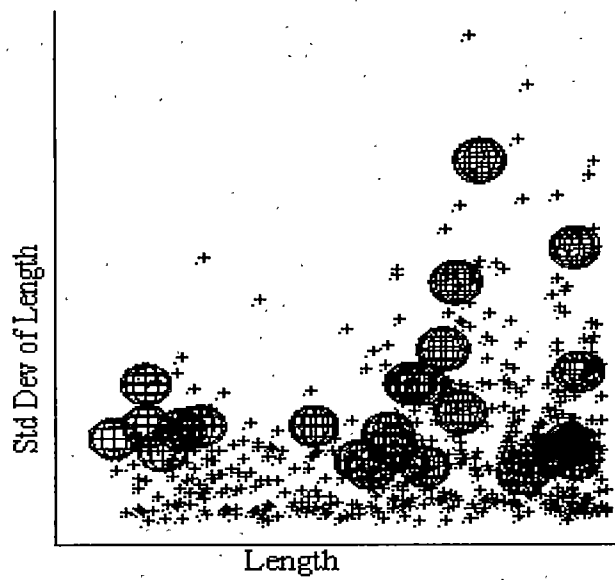


Figure 5.9: Centers placed in the length dimension of the input of the training data.

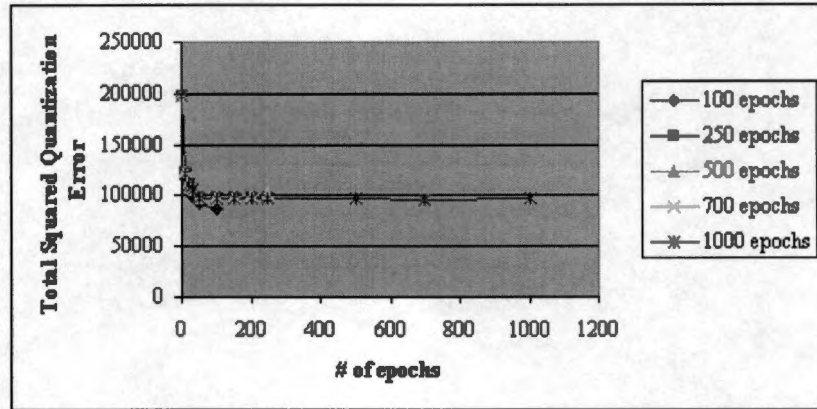


Figure 5.10: Plot of the TE as a function of the number of epochs.

number of iterations. Usually, selection of the number of iterations is also important to the outcome of the classifications. This is because the learning rate is a function of the number of iterations. Recall from Section 3.4 that the learning rate decreases linearly. The learning rate in this research was decremented each epoch by $\text{learning_rate_constant}/\text{epoch_to_stop}$. Figure 5.11 is a plot of the classification accuracy as a function of the number of epochs. There seems to be no significant difference between the outcomes. However the miscellaneous class does seem to perform better when a lower number of epochs is specified. This is because the other centers have not optimally placed themselves. It seems that the reason that the number of epochs to train does not play a considerable role in this research is because the centers are either in or very close to their corresponding decision regions

Tables 5.1 displays the outcome of training and testing with parameters discussed above based on the groundtruth of the original image. For training the neural network performed crack type classification best for longitudinal, transverse, and block cracking producing respectively correct values of 86.3, 90.9, and 78.0 percent. Fatigue

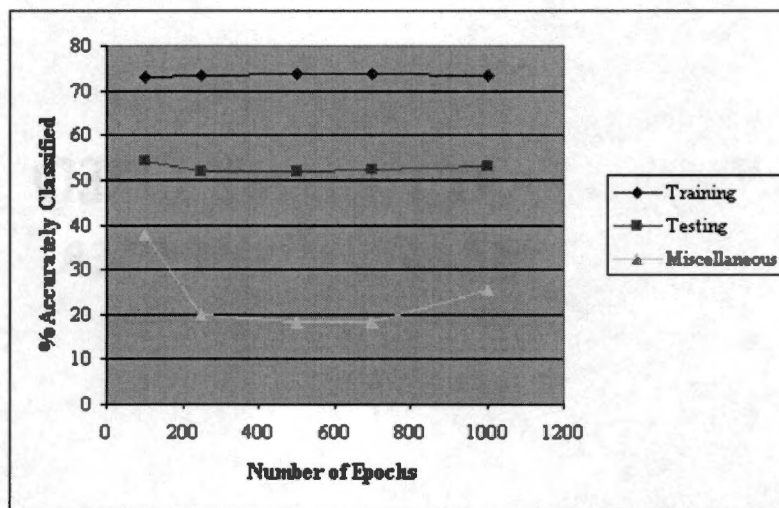


Figure 5.11: Plot of the accuracy of the classifier as a function of the number of epochs.

Table 5.1: Results for training on the data from the ground truth of original image. The results of human interpretation of the corresponding vectorized image were also provided.

Type	Num.	Neural Network		Human Interpreter	
		Num. Correct	% Correct	Num. Correct	% Correct
Fatigue	132	76	57.6%	70	49.6%
Block	100	78	78.0%	71	74.5%
Longitudinal	249	224	90.0%	207	78.5%
Transverse	73	63	86.3%	73	100.0%
Miscellaneous	71	18	25.4%	63	86.7%
Total	625	459	73.4%	484	77.4%

and miscellaneous cracking performed the worse.

The test results based on the ground truth of the original image (Table 5.2) showed similar results with transverse, longitudinal, and block cracking having the best results, namely, 74.0, 70.0, and 67.7 percent correct. Fatigue cracking had a low accuracy of 43.8 percent and the miscellaneous category was 8.4 percent correct.

An explanation for the miscellaneous class has already been provided. Most of the error associated with the other classes can be attributed to 1 of 2 events. The first event is simpler to explain. Samples that lie in between decision regions on the outer edges were more prone to be misclassified. Centers are not found in these regions as can be seen in Figure 5.9. This is because centers are repelled from these areas due to the “speckled” nature of data in these regions.

Table 5.2: Results for testing on the data from the ground truth of original image. The results of human interpretation of the corresponding vectorized image are also provided.

Type	Num.	Neural Network		Human Interpreter	
		Num. Correct	% Correct	Num. Correct	% Correct
Fatigue	105	46	43.8%	58	55.2%
Block	65	44	67.7%	44	67.7%
Longitudinal	203	142	70.0%	148	72.9%
Transverse	96	71	74.0%	70	72.9%
Miscellaneous	119	10	8.4%	76	63.9%
Total	588	313	53.2%	396	67.3%

The other error can be attributed to mislabeling of the input samples, either “accidentally” or “unknowingly.” A human interpreter ground truthing the data could mislabel the input samples “accidentally” because of lack of attention paid while ground truthing or a number of other reasons. The “unknowingly” mislabeling is caused by the segmentation problem discussed earlier. It seems that crack segments that show up in the original image may not show up in the segmented image. The segmentation also detected objects that do not show up as linear feature in the original image. So thus, when one is assigning classification for a particular segment in the image(that does not show up in the segmentation) data could actually support another crack type(that may or may not show up in the original image).

At first glance these results do not seem to be highly accurate. However earlier in

the chapter we discussed the fact that a human interpreter also would have difficulties classifying the images using the vectorized images from which information for the neural network was extracted. The human interpreter using this data, obtained results comparable to the neural network on both the training and testing cases. The human interpreter did perform significantly better when classifying the miscellaneous cracks. A human interpreter can determine shorter crack segments on sight. However, as explained previously in this chapter, the neural network could not find the optimum decision region of the miscellaneous class. During testing, the neural network did not perform as well as the human interpreter in distinguishing the fatigue class. Those samples in the testing data did not lie in the fatigue crack region that was determined in the training. Most of the examples fell along the edges of that region. Figure 5.12 shows a plot of the samples in their orientation dimension and Figure 5.13 shows the centers placed in their decision regions. The centers in the middle belong mostly to the fatigue class. However it can be seen that most of the samples in Figure 5.14 lie along the edge of this decision region. This is further verified by Figure 5.15, which is a class confusion matrix for the test samples.

For the sake of comparison, training of the neural network was also performed on the data obtained from ground truthing the vectorized images. This was done to get some idea of how much the result may have been affected by the loss of information inherent in the vectorization. It should be noted that one would not normally ground truth using vectorized images. It is desirable that the ground truth accurately reflect what it is in the original pavement images.

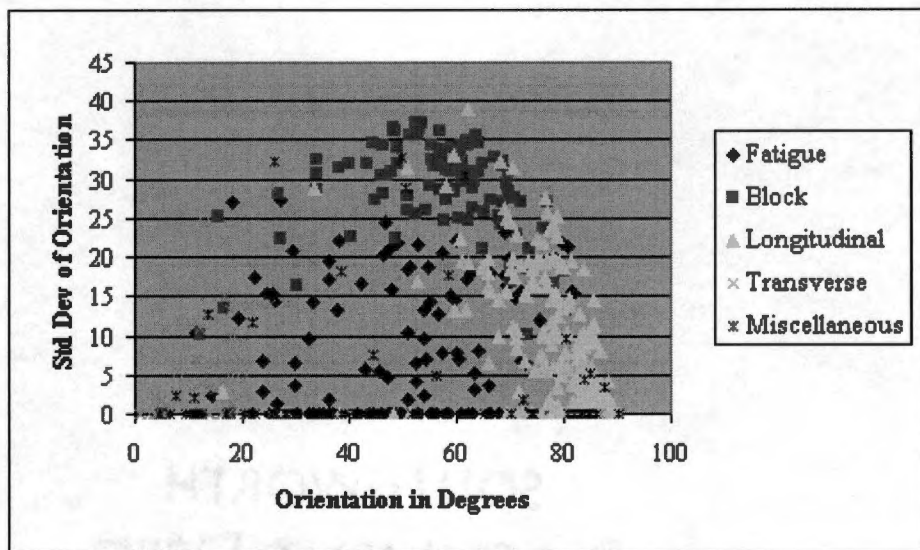


Figure 5.12: Plot of the training samples in the orientation dimensions.

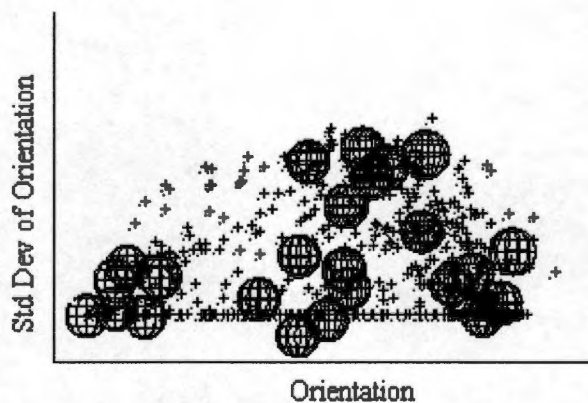


Figure 5.13: Trained centers placed in the orientation dimensions of the input of figure 5.12.

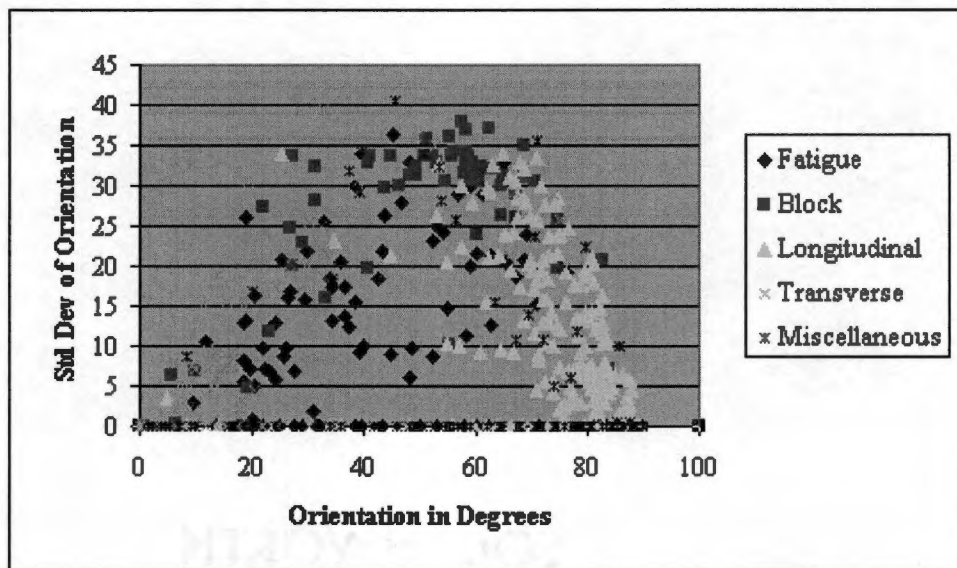


Figure 5.14: Plot of the testing samples in the orientation dimensions. Notice how most of the fatigue samples lie in the outer edges of the decision region displayed in Figure 5.13.

Type	Fatigue	Block	Long.	Transverse	Misc.
Fatigue	0	14	15	24	4
Block	1	0	6	7	5
Long.	19	29	0	2	8
Transverse	6	9	4	0	4
Misc.	25	7	53	23	0

Figure 5.15: Confusion Matrix for the final classification of the testing data.

As seen in Tables 5.3 and 5.4, these results are significantly better than those obtained from training and testing with data supplied from the ground truths of the original data. However, the fatigue class in the testing samples exhibited the same problem as it did earlier when using the data supplied by the ground truths of the original images. Again referring to Figure 5.14, the lower correct percentages for the fatigue cracking type is attributable to the fatigue sample falling along the edge of the fatigue decision region. Recommendations for improvements and future research are presented in the following chapter.

Table 5.3: Results for training with data obtained from the ground truth of the vectorized images.

Type	Num.	Num. Correct	% Correct
Fatigue	78	62	79.4%
Block	77	69	89.6%
Longitudinal	251	220	87.6%
Transverse	78	78	100.0%
Miscellaneous	141	108	76.6%
Total	625	537	85.9%

Table 5.4: Results for training with data obtained from the ground truth of the vectorized images.

Type	Num.	Num. Correct	% Correct
Fatigue	92	28	30.4%
Block	62	52	83.9%
Longitudinal	197	158	80.2%
Transverse	95	85	89.5%
Miscellaneous	142	77	54.2%
Total	588	400	68.0%

Chapter 6

Conclusion

The results of this research demonstrate that the proposed classification technique has potential as a classifier in an automated Pavement Management System. The approach was able to distinguish the different distress types with reasonable accuracy.

A Learning Vector Quantizer was implemented to classify pavement images. Classification was performed using simple parameters that were extracted from vectorized crack segments. Classification of the original grey scale image was taken to be ground truth for the training and testing of the neural network. Recalling from Chapter 5, human interpretation of the vectorized images was used to measure the performance of the classifier. This was done to determine how well the neural network could perform with the given information. The results achieved by the neural network and the human interpreter were comparable.

In the training phase, the neural network achieved an overall accuracy of 73.4 percent. For the same phase, the human interpreter achieved an accuracy of 77.4 percent. For testing, the neural network achieved an overall accuracy of 53.2 percent

and the human and likewise, the human achieved 67.3 percent accuracy. These results are reasonable, but show that more research needs to be invested in order to achieve a higher accuracy. The following sections discuss areas that need to be explored in order to improve the general classifier.

Segmentation

Based on the results from the two different ground truths, it can be seen that the segmentation plays a vital role in the classification. It can also be seen that the neural network technique is very sensitive to the results of segmentation. Humans are looking for essentially linear features within the image and can determine the most faint ones. Because of this, it is very possible to see something that the segmentation technique does not pick up.

In general, the segmentation technique used to produce data for this research is well-established technique. However, there may not be the most appropriate technique for this particular research.

Fatigue Cracking

From the results, it can be seen that the fatigue cracking class had the worse classification results. Fatigue cracking was not trained and tested on data that well represented this class. Most of the images provided did not have examples of fatigue cracking. Diagonal cracks were used to represent the fatigue distress. A region of alligator cracks contains several cracks, mostly diagonal, along with a network of fainter cracks [14].

In the future, images that contain actual examples of fatigue cracking should be

used to train and test the classifier. However, the results from this research are not to be dismissed. They demonstrate the feasibility of classifying fatigue cracks with this method. It should also be noted that fatigue cracking is one of the most difficult distress types to classify.

Gridding Technique

Grids were overlaid on the images to perform the ground truth. The sizes of the grids were based on the division of the image. There was a small amount of error that resulted from the gridding system. Because the grid values were converted from decimal values to pixel integer values, some of the grids did not line up with each other(Figure 6.1). This caused some of the crack segments to appear as if they were lying in another block.

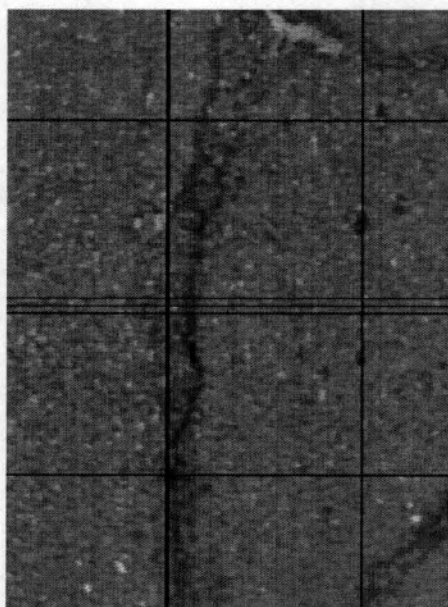


Figure 6.1: Portion of image with grids overlaid.

Weighted Averaging

A weighted average was calculated for each of the input parameters in each block. This weighted average was more appropriate than a simple mean. However, this approach still suffers from being more influenced by shorter crack segments. Take Figure 6.2 for example, the predominate crack types is longitudinal, but there is a smaller crack segment in the block as well. This crack segment was counted towards the average as well. Though it had limited influence, it still affected the average in a way that reduces the effect of the longer crack segment. This weighted average needs to be investigated. technique used to

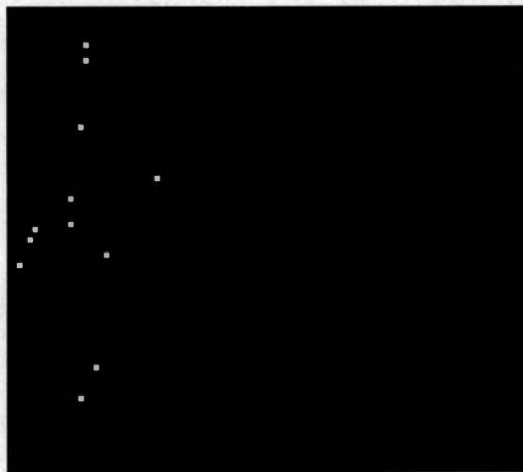


Figure 6.2: Portion of image showing longer crack segments with shorter crack segments.

6.1 Recommendations for Future Research

This approach was proposed for several reasons, one of the most important being the ease of calculating extent and severity of the distress types. This portion of this research stills needs to be explored. The block boundary information is stored in a file along with the input parameters for that block. Using this information and the quad tree type structure, it should not be a problem to calculate the extent and severity.

Currently the classification approach can not distinguish sealed cracks from unsealed cracks. Sealed cracks are those that have been repaired. On flexible pavement, cracks are often filled with a black polymer. The gray average is recorded for each crack segment. It might be possible to incorporate this data to add sealed cracks to the distress classes. This is important, because a state transportation agency does not want to invest planning time to repair a section of highway that has already been repaired.

We have identified four distress types to classify. However, there are many more distress types that can exist in highway pavement. Potholes are an example of another type of distress. It may be possible to incorporate the VFL in order to distinguish potholes. The feasibility of classifying other distress types should be researched in order to increase the robustness of this technique.

BIBLIOGRAPHY

Bibliography

- [1] J. A. Acosta, J. L. Figueroa, and R. L. Mullen. Implementation of the video image processing technique for evaluating pavement surface distress in the state of ohio. Technical report, Ohio Department of Transportation, U. S. Department of Transportation and Federal Highway Administration, 1994.
- [2] Bryan Keith Ary. Data mining over mismatched domains. Master's thesis, University of Tennessee Space Institute, 1997.
- [3] G. J. Chong and W. A. Phang. *Flexible Pavement Condition Rating-Guidelines for Municipalities*. The Research and Development Branch Ministry of Transportation of Ontario, Ontario, 1989.
- [4] Chia-Pei J. Chou and Tz-Kai Liau. Development of automated algorithms for pavement condition survey. *Transportation Research Board*, 1(1536):103–109, 1996.
- [5] JaChing Chou and Wende A. O'Neill. Pavement distress classification using neural networks. *IEEE International Conference on Systems, Man and Cybernetics*, 1:397–401, 1994.

- [6] T. M. Cover and P. E. Hart. "Nearest Neighbor Pattern Classification". *IEEE Transactions on Information Theory*, IT-13(1):21-27, 1967.
- [7] Sean Paul DeMerchant. Segmentation and nonlinear parameter selection. Master's thesis, University of Tennessee Space Institute, 1998.
- [8] Ibrahim Mahmoud El Sanhoury. Computer-based segmentation and interpretation of pavement surface distress images. Master's thesis, Massachusetts Institute of Technology, 1990.
- [9] Richard Johnsonbaug Gose, Earl and Steve Jost. *Pattern Recognition and Image Analysis*. Prentice Hall PTR, New Jersey, 1996.
- [10] Jack A. Hansen, Sean P. DeMerchant, and Alfonso Pujol. An adaptive segmentation technique for automate crack detection. In *Proceedings of the Transportation Research Board 78th Annual Meeting*, pages 1-22, 1999.
- [11] Mohamed Said Kaseko. *A Neural Network-Based Methodology for Automated Distress Classification of Highway Pavement Images*. PhD thesis, University of California, Irvine, 1992.
- [12] Tarun Khanna. *Foundations of Neural Networks*. Addison-Wesley Publishing Co., Massachusetts, 1990.
- [13] Teuvo Kohonen. Improved versions of learning vector quantization. *International Joint Conference on Neural Networks*, I:545-550, 1990.
- [14] H.N. Koutsopoulos and A.B. Downey. Primitive-based classifications of pavement cracking images. *Journal of Transportation Engineering*, 119(6):402-418, 1993.

- [15] Patrick van der Smagt Krose. *An Introduction to Neural Networks*. The University of Amsterdam, Amsterdam, 1996.
- [16] Zhen-Ping Lo and B. Bavarian. A neural piecewise linear classifier for pattern classification. *International Joint Conferences on Neural Networks*, 1:263–268, 1991.
- [17] David S. Mahler, Zuhair B. Kharoufa, Edward K. Wong, and Leonard G. Shaw. Pavement distress using image processing techniques. *Microcomputers in Civil Engineering*, 6:1–14, 1991.
- [18] William Mendenhall and Terry Sincich. *Statistics for Engineering and the Sciences*. Macmillan Publishing Co., New York, 1988.
- [19] Marilyn McCord Nelson and W. T. Illington. *A Practical Guide to Neural Networks*. Addison-Wesley Publishing Co., Massachusetts, 1991.
- [20] Abhijit S. Pandya and Robert B. Macy. *Pattern Recognition with Neural Networks in C++*. CRC Press LLC, Florida, 1996.
- [21] John C. Russ. *The Image Processing Handbook Second Edition*. CRC Press Inc., Florida, 1995.
- [22] Kohonen Teuvo. An introduction to neural computing. *Neural Networks*, 1:3–16, 1988.

APPENDICES

Appendix A

Images Used in Research



Figure A.1: Training Image 1

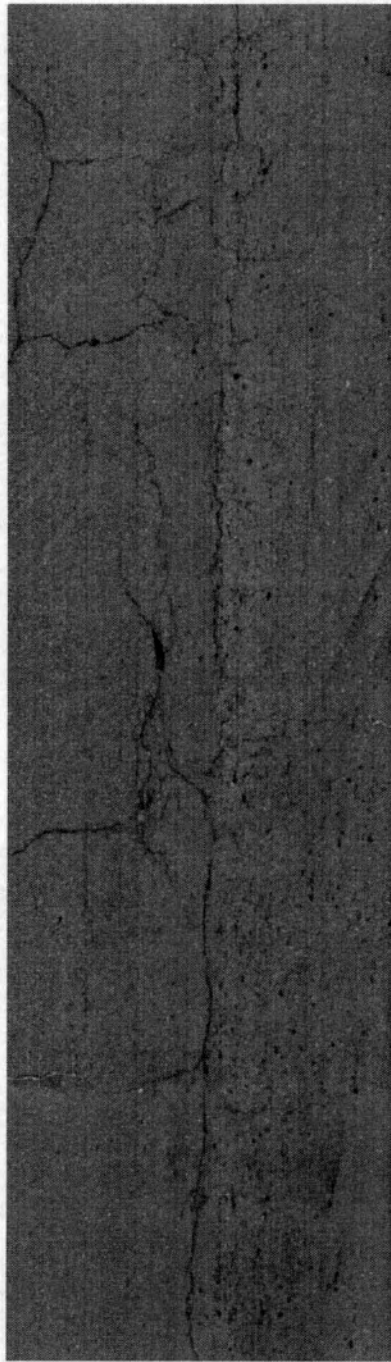


Figure A.2: Training Image 2

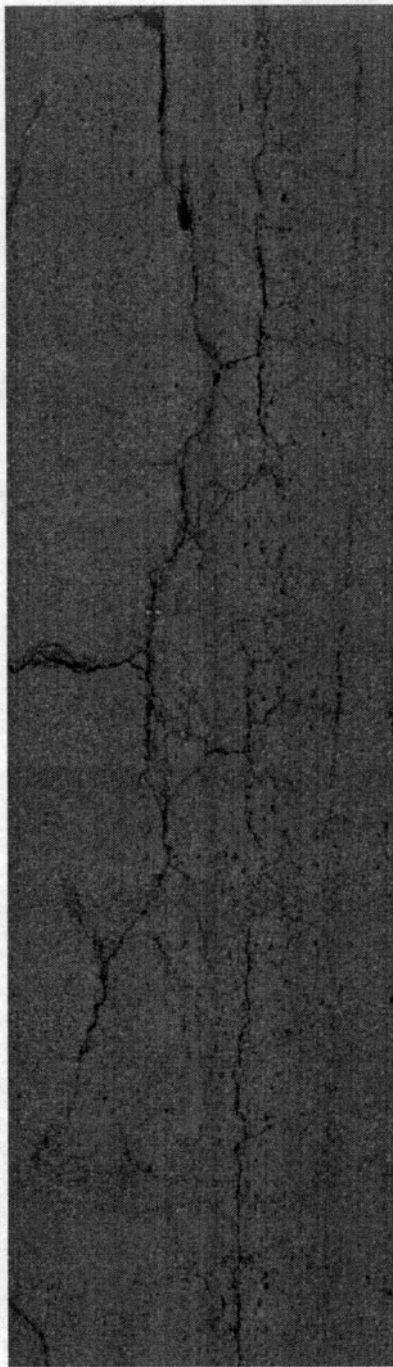


Figure A.3: Training Image 3

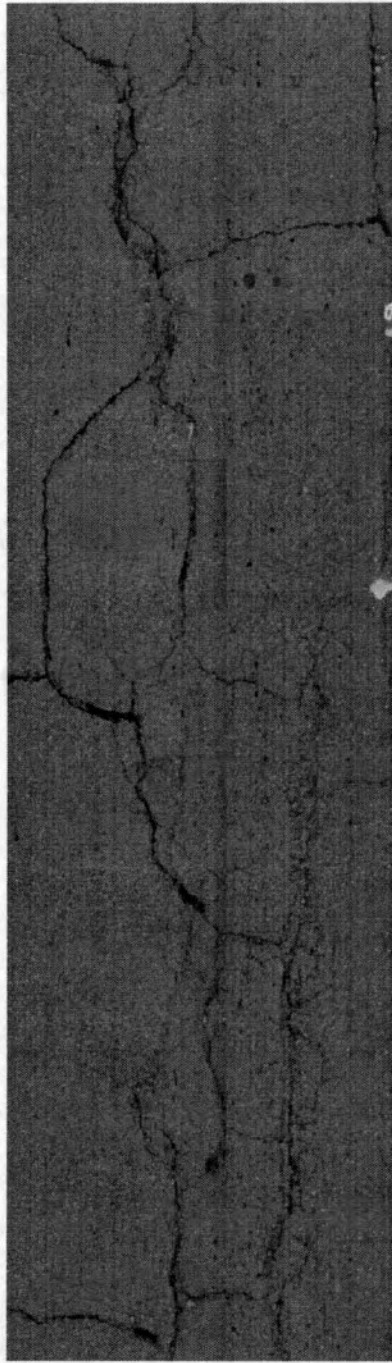


Figure A.4: Training Image 4

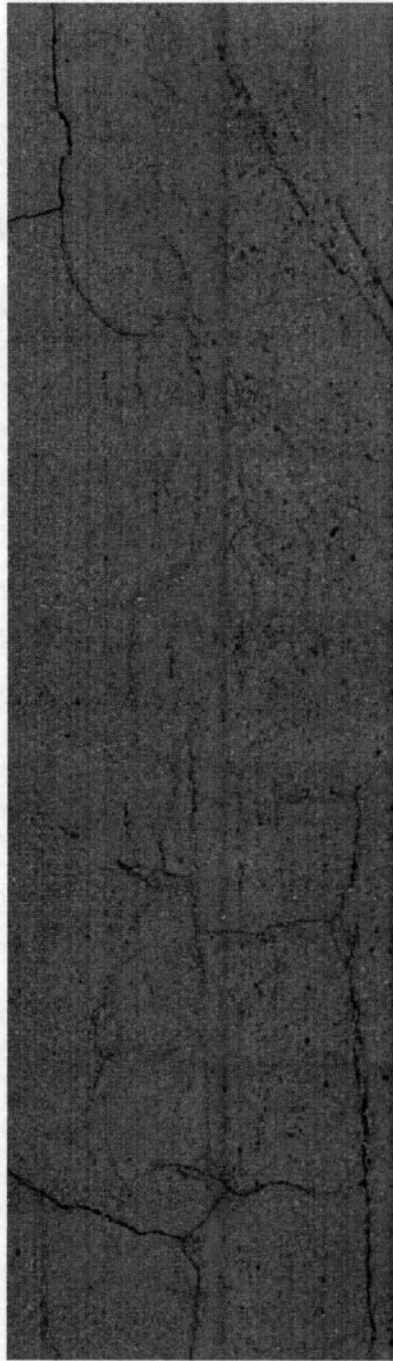


Figure A.5: Training Image 5

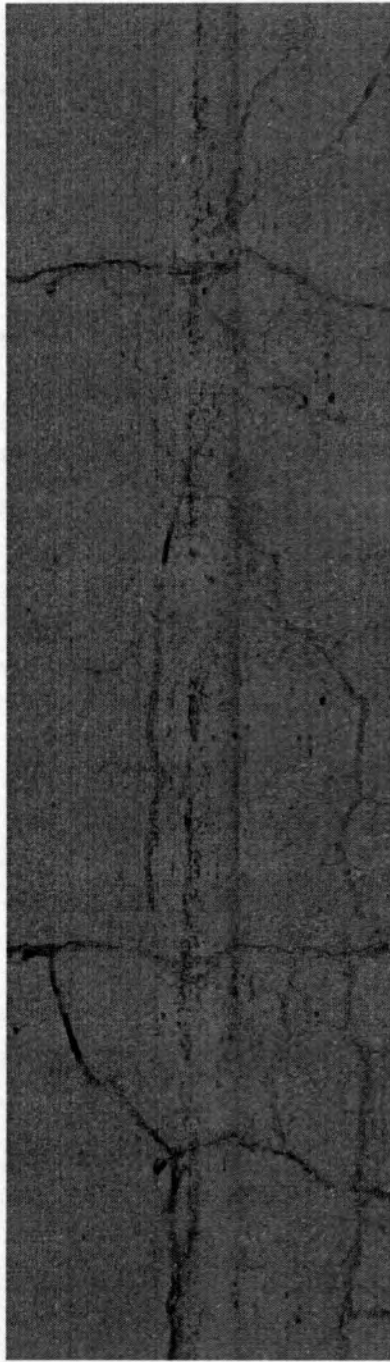


Figure A.6: Testing Image 1



Figure A.7: Testing Image 2



Figure A.8: Testing Image 3



Figure A.9: Testing Image 4

Appendix B

Distress Types and Severities

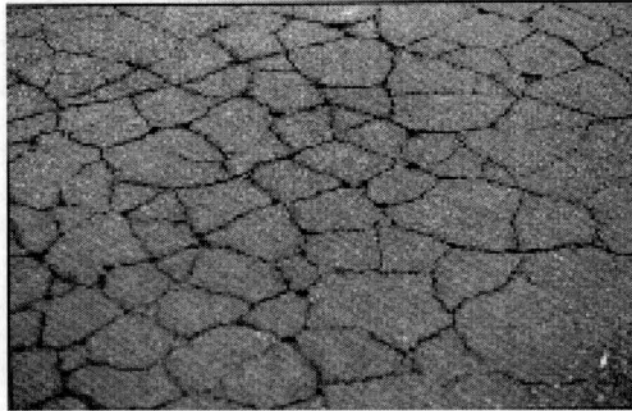


Figure B.1: Fatigue crack with alligator/chicken wire pattern typical in fatigue cracking.



Figure B.2: Moderate severity Fatigue cracking.



Figure B.3: High severity Fatigue cracking.

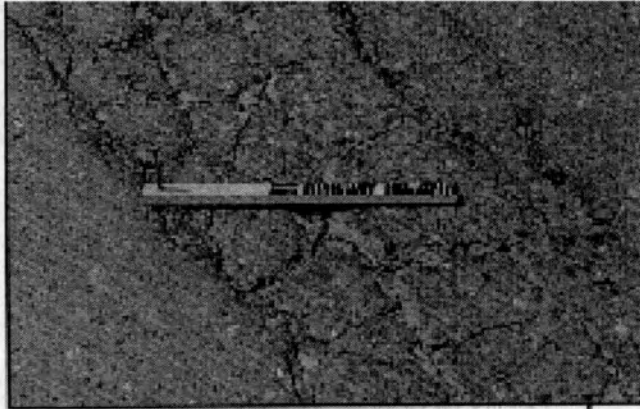


Figure B.4: High severity Fatigue cracking with spalling.

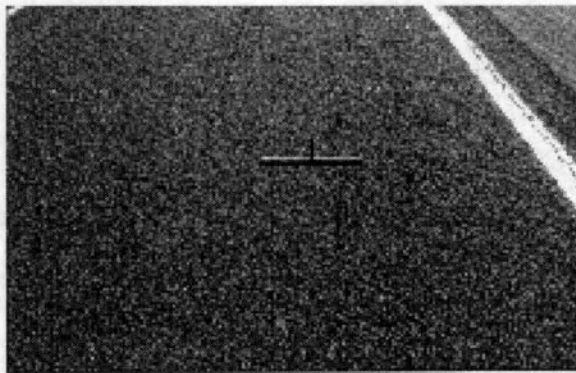


Figure B.5: Moderate severity Block cracking.



Figure B.6: High severity Block cracking.

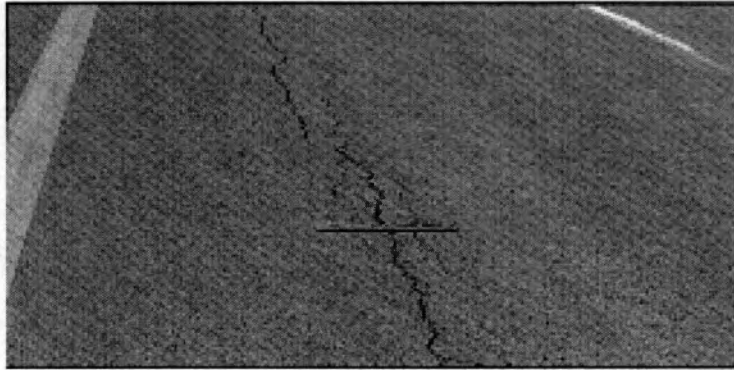


Figure B.7: Moderate severity Longitudinal cracking.

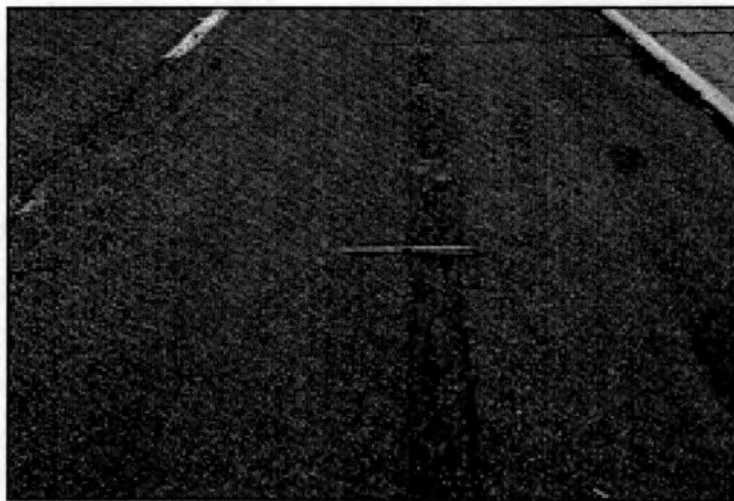


Figure B.8: High severity Longitudinal cracking.

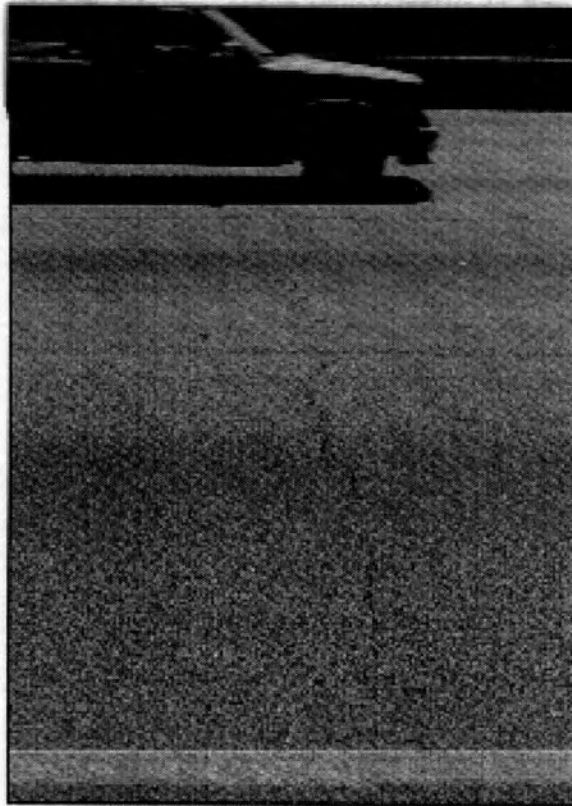


Figure B.9: Low severity Transverse cracking.

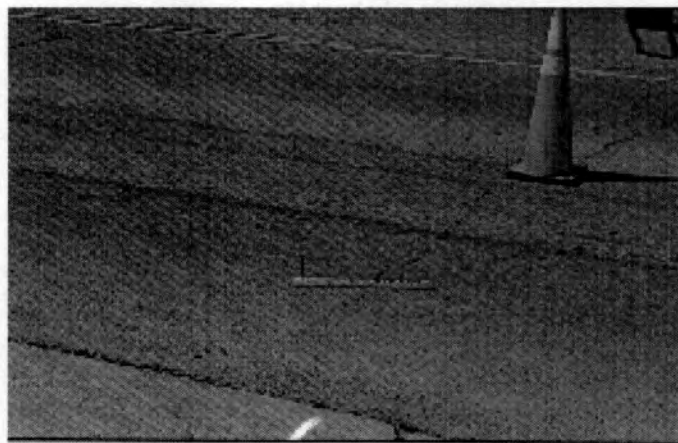


Figure B.10: Moderate severity Transverse cracking.

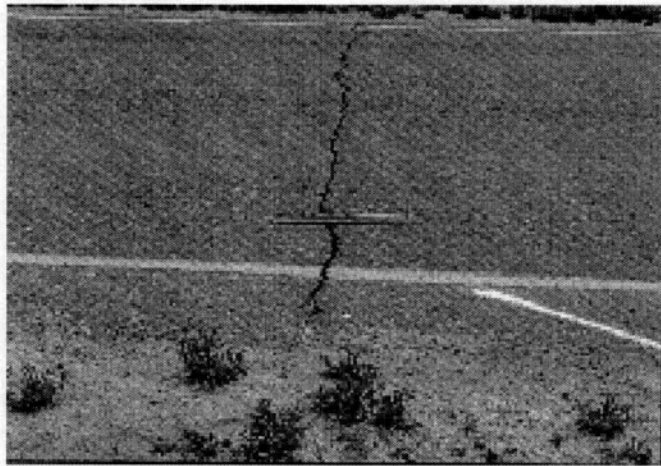


Figure B.11: High severity Transverse cracking.

Vita

Judy Nicole Edwards was born in New York, New York and raised in Suffolk, Virginia. She did her undergraduate studies at Norfolk State University and graduate studies at The University of Tennessee Space Institute. She is currently employed by Hewlett Packard as a Software Design Engineer.