12-2004

Feature Detection using Linear Structure Modeling and PCA

Kawshif Muhammed Riaz Ahmed

FEATURE DETECTION USING LINEAR STRUCTURE MODELING AND PCA

by

Riaz Ahmed, Kawshif Muhammed

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science in Engineering (Electrical)
Department of Electrical and Computer Engineering

Western Michigan University
Kalamazoo, Michigan
December 2004

ACKNOWLEDGMENTS

I would like to acknowledge my advisors Dr. Ikhlas Abdel-Qader and Dr Osama Abudayyeh for the guidance. I am very much thankful for the support provided by them that helped me complete my research in feature extraction. It was a great privilege for me to learn new ideas in this project that help me complete the thesis.

Secondly I would like to thank my friends and well wishers for the moral support. I also thank the members of my graduate committee, Dr. I. Abdel-Qader, Dr. R. Gejji, Dr. O. Abudayyeh, & Dr. M. Niewiadomska-Bugaj for their support and for reviewing my theis work.

Partial support of this work was provided by the National Science Foundation grant MRI- 0215356. The authors would like also to acknowledge Western Michigan University for its support and contributions to the Information Technology and Image Analysis (ITIA) Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the University of Western Michigan University.

Kawshif Muhammed Riaz Ahmed

FEATURE DETECTION USING LINEAR STRUCTURE MODELING AND PCA

Riaz Ahmed, Kawshif Muhammed, M.S.E

Western Michigan University, 2004

An algorithm to extract features from digital images such as cracks in concrete bridges for automated inspection is developed. The algorithm is based on the use of Principal Component Analysis (PCA). In general, PCA is used to reduce the dimensionality of a data set that consists of a large number of interrelated variables while retaining the variation present in the original data set. This ability of PCA will be used in this project to identify clusters using two training sets of images. Linear structure modeling is used to emphasize the image features (cracks) prior to applying PCA. Whole images processing and block processing are used with different distance measures to optimize the accuracy of the results. Real concrete bridge images with cracks of different sizes and shapes as well as non-cracked images are used. The algorithm development and experimental results are presented.

TABLE OF CONTENTS

Table of Contents–Continued

LIST OF TABLES

LIST OF FIGURES

List of Figures–Continued

CHAPTER 1

INTRODUCTION

1.1 Overview

As we all know that digital image processing methods are basically used for either improving the picture information using image enhancement, autonomous machine perception by training a set of images, transmission of images after using image compression algorithm (1). There are different methodologies that can be applied to digital images for different purposes and also with different objectives. Image acquisition forms the first step in image processing. This step normally involves preprocessing like scaling, adjustments, etc. The second step involves the image enhancement, which is one of the most popular areas of image processing. There are different techniques in image enhancement. Usually image enhancement is done to highlight features of interest in a digital image. Also enhancement is used to increase the contrast of an image to make the image look better. Another methodology in image processing is the image restoration, which is also used to improve the appearance of an image. Usually enhancement is based on human subjective preferences and restoration is objective as it is based on mathematical model or probabilistic model of image degradation. One of the most important areas in digital image processing is the feature recognition.

Feature recognition aims to recognize specific features of interest in a digital image. So that when a new set of images are treated as inputs the same features are extracted and matched with the training set images features. Then it is decided whether

those are similar images or not. Feature detection algorithms have many applications in digital image processing. For example it is has been extensively used in face recognition. In this thesis the feature detection algorithm is implemented to detect cracks in the concrete bridge images. The cracks in a concrete bridge can be of any type. It can be a horizontal crack, vertical crack, oblique etc and it can be found at any part of the image. So the feature detection algorithm had to be built to detect these types of cracks on real concrete bridges. The main problem faced was, if the concrete bridge has some of the features that resembling the cracks can lead to failure of the algorithm. Thus Principal component analysis was used in the feature detection algorithm. As some of the bridge images were misidentified as cracked or vice versa, linear structure modeling was introduced to solve the problem. Linear structure modeling involved the convolution-based masks. There was an improvement in the results. But still there was some problem in identification of cracks as the images had to be re-sized to lower dimensions. To avoid resizing and also to improve the results further block processing was used. In block processing the images were broken down into blocks. The results obtained from all the three analyses will be discussed in this thesis.

1.2 Problem Statement and Goal

The purpose of the thesis was to develop an algorithm using PCA and linear structure detection for feature detection purposes, specifically to detect cracks in concrete bridges. With the use of just the PCA alone was not enough to obtain accurate results So many algorithms were searched (2). And then a line detection method was used to find the linear structures in the bridge images and then the PCA was used to determine

whether the bridge is cracked or not. There are quite a few approaches that have been used for line detection in a digital image. Several different approaches have been described for detection of lines (3), such as directional morphology, simple orientated bins, Hough transform, a directional line operator, curvi-linear structure detection, and directional second order Gaussian derivatives.

There are some methods based on flow fields and also using the information from the Fourier transform. The aim of developing the linear structure detection algorithm is for highlighting any linear structures in the images. After the line strengthened images are obtained. The next step involves filtering, and then any modeling method can be used to identify whether the bridge images have cracks or not. In our case PCA model is used after the linear structure detection, where two sets of training data are used. One set consisted of cracked concrete bridge images, while the other set consisted of non-cracked concrete bridge images. The images are stored in two different locations and the algorithm is applied separately to both the sets of images.

In this thesis a convolution based method is used to find lines in the concrete bridge image. The algorithm rejects some of the unwanted linear structures using a filter and accepts the others based on criteria that have been decided. The algorithm employs PCA to detect cracks in them using a training set that contains enough samples of both kinds of images, that is cracked and non-cracked. So a step ahead was taken to use preprocessing before PCA is used. So the linear structure detection was used in the preprocessing stage. The cracks of different and sizes and orientations can be found in a bridge. The algorithm has to detect these types of cracks. The analysis was further extended to block processing, where the images were broken down into square blocks.

Now the PCA and linear structure modeling was applied to each block in the training set. Whenever a test image is introduced, it is decided whether each block in the test image is cracked or non-cracked. The results obtained using block processing are encouraging.

1.3 Related Work

Much related work on feature detection has been done on face recognition. Some of the approaches that were reviewed are listed as follows and the reason to use PCA has been shown.

1.3.1 Template Matching Using Correlation

The very common technique for template matching in image processing is the use of correlation. This has been applied by Kosugi (4) to detect facial features and in face recognition. Matching Faces by correlation consists of two steps. First an image vector is formed by lithographically ordering the gray scale image of the unknown face. This image vector is then compared to a database of known faces by calculating the correlation between the image vector and each member of the database. The member of the database resulting in the largest correlation is then deemed the closest match. Correlation suffers from high sensitivity to scaling and noise and performs poorly in badly lit conditions.

1.3.2 Self Organizing Maps (SOMs) and Convolutional Neural Networks (CNN) Approach

Using neural networks for feature recognition is another popular approach found in literatures. The SOM was first developed by Kohonen, T et al. (5). This is an unsupervised learning process, which maps the distribution of data without any prior knowledge of its class information.

Self organizing maps belong to a class of neural networks known as Topology Preserving Feature Maps (TPFMs). In TPFMs input pattern, which is highly similar result in excitations in geographically neighboring areas of the network. The use of TPFMs is motivated by Biological observations, particularly existence of tonotopic and retinotopic structures within the cortex of the human brain.

The problem with the SOM is that it arbitrarily divides input space into a set of classes of which the designer has no control or knowledge. The output of the SOM is fed into a convolutional neural network (CNN). CNNs exhibit some degree of translation and rotation invariance and their use is again motivated by biological observations. A CNN consists of a number of three-dimensional neural networks connected in series. Each neural network consists of several planes of neurons, with each neuron only connected to its six immediate neighbors. Each point on the neural network corresponds to a window in the input matrix, with the values of the input matrix in that window being fed to the neuron at that point. Neurons with the same x and y co-ordinates on different planes share a common input window. Furthermore, a constraint is placed on the network that the input weightings for all neurons in the same plane must be equal.

The biggest problem in neural networks is its inability to deal with the high dimensionality.

1.3.3 Eigen Faces Approach

Eigen faces Approach (6) used by Turk and Pentland et al. has shown that the scaling or normalization of facial features according to their relative importance in face recognition is the basic premise behind the eigen faces technique. This method attempts to capture the variation between facial images in an orthogonal basis set of vectors referred to as eigenfaces. The eigenfaces are thus the image vectors, which map the most significant variations between faces. Under the assumption that faces form a simply connected region in image-space (the space containing all possible images, facial and non-facial), we can represent any face as a linear combination of eigenfaces. Each face can thus be represented by weight vector, which contains the proportion of each eigenface needed to construct the face. By comparing the weight vector of the unknown face to the weight vectors of a database of known faces a match can then be determined.

Unlike the correlation-based technique, eigenfaces are robust against noise, poor lighting conditions and partial occlusion. Eigenfaces are relatively insensitive to small variations in scale, rotation and expression. Furthermore, through the use of multi-resolution pyramids and multiple eigenspaces, eigenfaces can be adapted to deal with large changes in scale, in-plane and out-of-plane rotation.

Eigenfaces have shown to produce 96% correct classification under varying lighting conditions. Eigen faces have also been shown to maintain accuracy even with

large-scale database. Finally eigenfaces have been shown to run in real-time speeds in low-end workstations. This means they are suitable for a real-time system.

1.4 Thesis Methodology Overview

In this thesis a feature detection algorithm to extract features from digital images such as cracks in concrete bridges for automated inspection is developed. The algorithm is based on the use of Principal Component Analysis (PCA). In general, PCA is used to reduce the dimensionality of a data set that consists of a large number of interrelated variables while retaining the variation present in the original data set. This ability of PCA will be used in this project to identify clusters using two training sets of images.

A dataset containing 25 cracked and 25 non-cracked bridge images is used to test the algorithm. The first 20 cracked bridge images are trained using the algorithm and the remaining 5 cracked bridge images are used as the test images and also 25 non-cracked bridge images are used as test inputs. In the second case the first 20 non-cracked bridge images are trained and the remaining 5 non-cracked bridge images along with the 25 cracked bridge images are used as the test inputs. In this thesis the Euclidean distance measure was used as the first choice to evaluate the accuracy of the results.

Linear structure modeling is used to emphasize the image features (cracks) prior to applying PCA. Whole images processing and block processing are used with different distance measures to optimize the accuracy of the results. Real concrete bridge images with cracks of different sizes and shapes as well as non-cracked bridge images are used. The algorithm development and experimental results are presented.

CHAPTER 2

FEATURE EXTRACTION AND PCA

2.1 Feature Extraction

A digital image is represented by features that contain the information important for image interpretation. In digital image processing many techniques has been proposed for feature extraction, which will be discussed briefly. Basically feature extraction techniques depend on the type of feature to be extracted. Each technique utilizes different mathematical model. The features can be classified mainly into 3 types namely image edges, lines and homogeneous regions.

Image edges can be extracted using many edge detectors like canny (7), laplacian of Gaussian (8), sobel (1).

Image lines (3) can be extracted using many techniques like Hough transform, directional morphology, simple orientated bins, Hough transform, a directional line operator, curvi-linear structure detection, and directional second order Gaussian derivatives.

Here homogeneous regions are image areas fulfilling a certain similarity criterion, e.g. homogeneity of intensity or color. The goal of segmentation algorithms is to split the image into homogeneous regions that are connected and are covered by non-intersecting edges. These homogenous regions can either be represented by the closed polygons or by enumeration of their member pixels. There are various approaches for region extraction in image processing like thresholding techniques (9). In this process the pixels are divided into a class and the neighboring pixels are merged into the class if they are of same class.

Another technique is the manual extraction where people evaluate the images and the pertinent information is confirmed or identified visually. But this technique is tedious and time consuming for large images.

In this thesis the feature extraction and then detection is performed using principal component analysis (6). In principal component analysis it is possible to work out an optimal co-ordinate system for images. Here an optimal co-ordinate system refers to one along which the variance of the images is maximized. This becomes obvious when we consider the underlying idea of PCA. PCA aims to catch the total variation in the training set images, and to explain this variation by as few variables as possible. This is important because it allows us to explain an observation example a bridge image with crack with fewer variables. This not only decreases the computational complexity of feature recognition, but also scales each variable according to its relative importance in explaining the observation. Components of this optimal basis will be orthogonal and will maximize the variance in the training set images. There is a need to evaluate the variance in the training set images along a given basis. This is precisely what eigen vectors achieve.

2.2 Principal Component Analysis (PCA)

In general PCA is used to reduce the dimensionality of a data set that consists of a large number of interrelated variables, but still retaining the variation present in the data set as much as possible. These goals are attained by transforming principal components that are correlated to a new set of variables that are uncorrelated and also are ordered so that the first few retain most of the variation present in all the original variables. PCA has

many applications in image understanding and pattern recognition that includes pattern matching (10, 11), neural networks (12, 13) speech analysis (14), visual learning (15, 16), and active vision (17). In feature recognition, PCA has been extensively used to identify face features (18).

PCA is a standard decor relation technique and following its application, one derives an orthogonal projection basis, which directly leads to dimensionality reduction, and possibly to feature recognition (10). In our algorithm covariance matrix was used to obtain the eigen vectors and eigen values. The principal components can be found using either covariance matrix or correlation matrix. A covariance matrix is a square and symmetric matrix. Where diagonal elements are variances and the other off-diagonal elements are covariances. Let I be a random vector representing an image, where M is the dimensionality of the image space. The vector is formed by concatenating the rows or the columns of the image, which may be normalized to have a unit norm.

The covariance matrix of I is defined as follows:

$$C = \sum [I - I_{ave}][I - I_{ave}]' \qquad (1)$$

Where C denotes the covariance matrix, The PCA of a random vector I factorize the covariance matrix into the following form:

$$\sum = \alpha A \alpha^T \text{ With } \alpha = [\alpha_1, \alpha_2, \ldots\ldots \alpha_M] \qquad (2)$$

Where α is an orthonormal eigenvector, A is the eigenvalue matrix having eigenvalues λ1, λ2, λM as the diagonal elements in the decreasing order.

The PCA is used specially for the dimensionality reduction purpose due to its property of optimal reconstruction of an image with respect to the minimum Mean Square Error (MSE). That is only few principal components can be used to represent the original image. For example principal components P = [α1, α2, … αE] can be used to construct the random vector I where E < M

$$Y = P^T I \tag{3}$$

The lower dimensional vector Y is nothing but the fingerprint of the random vector I on the vector space. Most Important features of the original random vector can be found in the projection of the vector I. The eigenvectors define a feature space, which drastically reduces the dimensionality of the original space, and feature detection is carried out in the reduced vector space.

2.3 Crack Detection

Bridge monitoring and inspection are expensive, yet essential tasks in maintaining a safe infrastructure. Traditionally, the primary method used to monitor bridges has been visual inspection. During a typical bridge inspection, the various components of a structure are examined at close range by trained inspectors, who evaluate the condition of the components and give them a ranking (20). This ranking is a qualitative evaluation of the current condition based on a set of guidelines and on the inspector's experience. For many situations, this type of evaluation is appropriate and effective. However, due to the subjective nature of this evaluation, rankings of the conditions of similar bridge components can vary widely from inspector to inspector, and from state to state.

Moreover, inspections are not necessarily always performed at the appropriate or critical times. In the past two decades, the need for more advanced methods for bridge inspection to ensure safety and early detection of flaws has been well recognized. Many evaluation methods are designed to operate upon existing bridges without damaging their usability. These methods are known as non-destructive evaluation (NDE) methods.

2.3.1 Types of Cracks in Bridge Decks

Different type of cracks can exist in a bridge deck. These cracks occur due to shrinkage of the concrete or inadequate temperature. The type of crack and the location of crack can determine the condition of a bridge deck. Cracks can be classified into longitudinal cracks, transverse cracks, alligator cracks etc. The longitudinal and transverse cracks occur due to spalling forces, which are caused due to the application of load (21). The alligator cracks occur usually in the bottom of the bridge decks due to high tensile stress and strain. The following figures 1, 2 and 3 show examples of these types of cracks. These images were used in the training set.

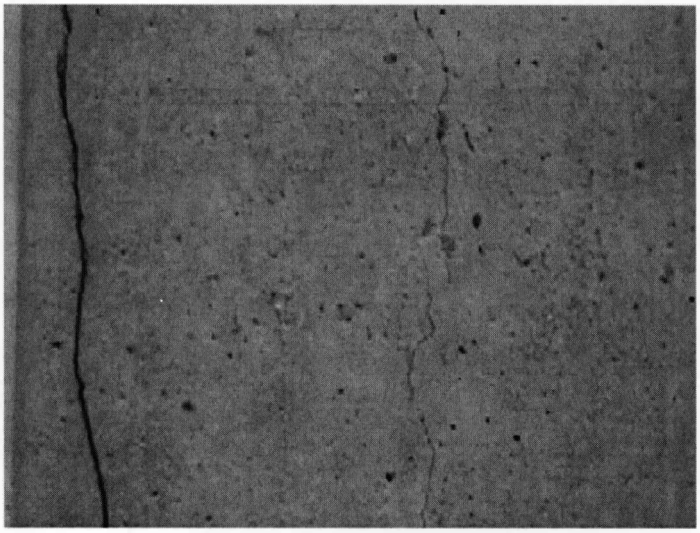


Figure 1: Concrete bridge with transverse crack

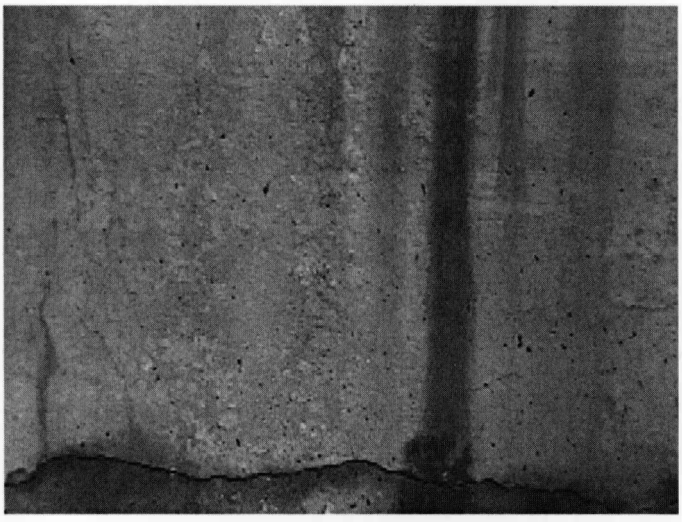


Figure 2: Concrete bridge with longitudinal crack

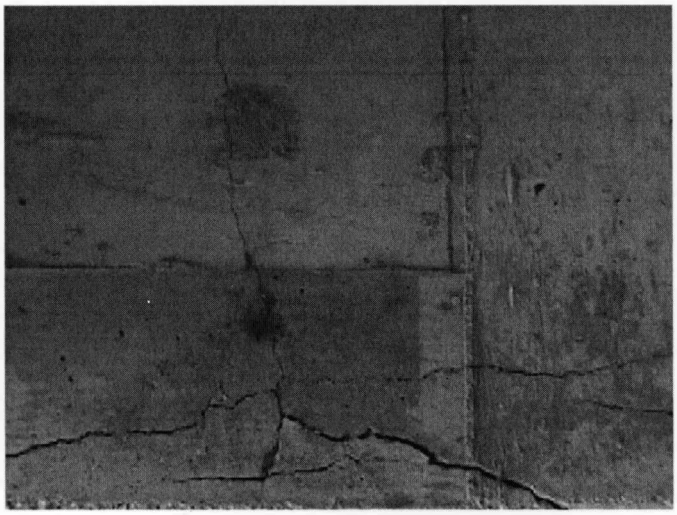


Figure 3: Concrete bridge with alligator crack

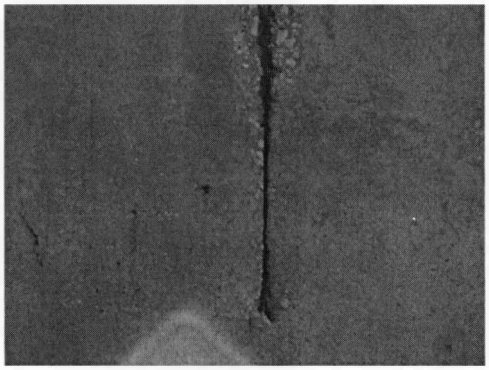
2.3.2 Methods to Detect Cracks on Concrete Bridges

There are many methods to detect cracks on concrete bridges. Inspectors have relied largely on visual inspections to evaluate the condition of bridges. State departments of transportation (DOTs) have employed nondestructive evaluation (NDE) methods to avoid visual inspection (21). But still use of NDE technologies has been limited. NDE methods are used to solve difficult inspection challenges that are beyond the capability of normal visual inspections. There are methods like thermo graphic methods (22) that employ thermo graphic systems are used to detect irregularities in heat transfer that occur due to delaminated material under ambient weather conditions. Also methods like laser technologies are used to measure the bridge deflections to evaluate structural behavior.

Automatic ultrasound testing (22) is another technique used by the inspectors to detect cracks on pipelines and in aeronautical applications. This technique combines the computer ultrasonic testing methods with computer data acquisition and

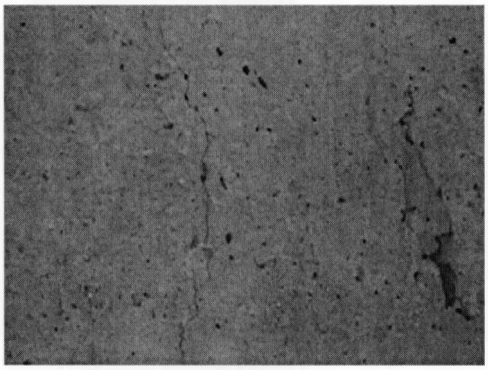
processing. New methods are being found to detect cracks like reactive powder concrete testing.

An effort is being made to find automated techniques to detect cracks on concrete bridges. In this thesis by means of feature detection a method for detecting any type of cracks on concrete bridges is shown. Figures 4 and 5 show some of the concrete bridge images used in the training set.

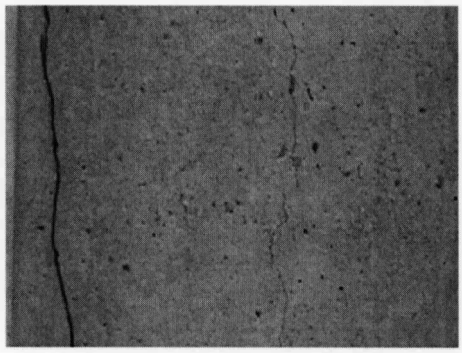


(a) Image 1

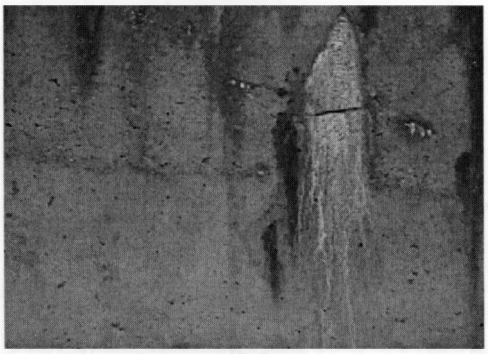
(b) Image 2



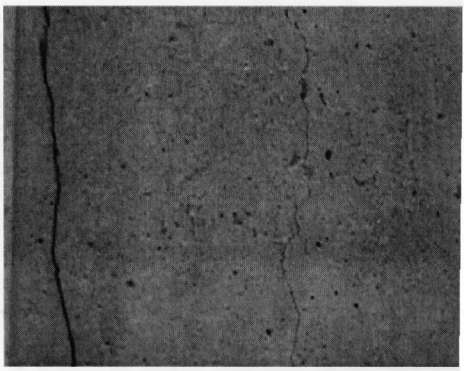
(c) Image 3



(d) Image 4

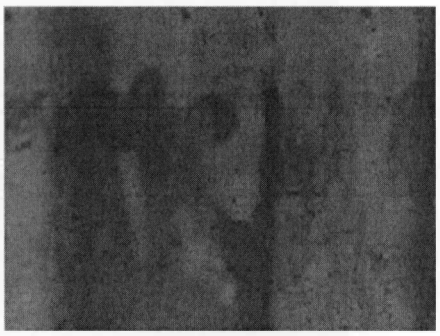


(e) Image 5

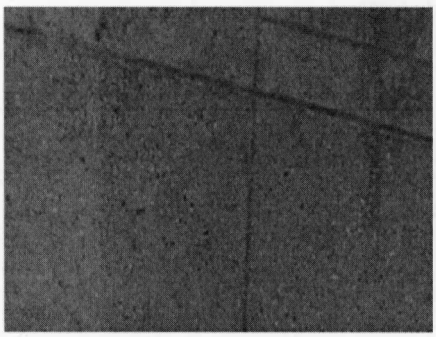


(f) Image 6

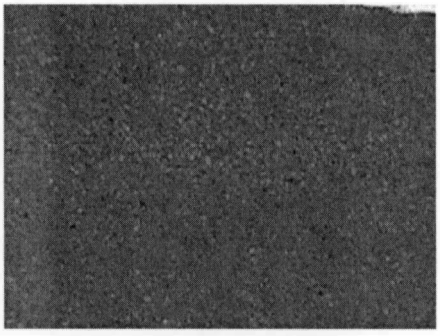
Figure 4: Few of the bridge images with cracks (a-f) used in the training set



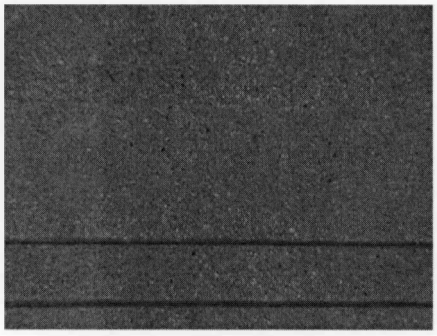
(a) Image 1



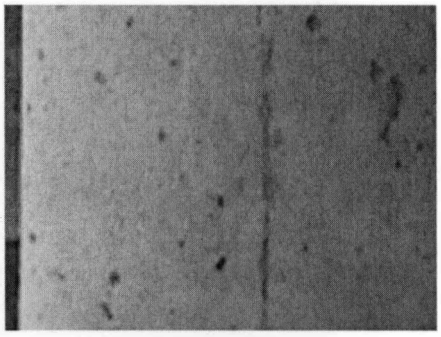
(b) Image 2



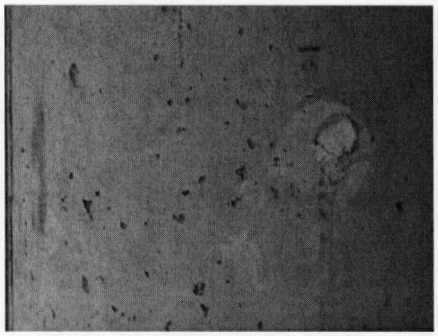
(c) Image 3



(d) Image 4



(e) Image 5



(f) Image 6

Figure 5: Few of the bridge images with no cracks (a-f) used in the training set

CHAPTER 3

METHODOLOGY

3.1 Principal Component Analysis (PCA)

Once bridge images are represented as co-ordinate vectors in a low dimensional space, the task of crack detection itself is quite simple. First, we look at the placement of different vectors representing the bridge images that are cracked. Then using the PCA we find the projections representing the bridge images that are cracked. New bridge images are introduced and are broken down into lower dimensional vectors using similar techniques as used to establish the co-ordinates of bridge images. Now, the crack detection simply becomes a mathematical calculation of distance, to see whether or not a new bridge image falls into the field of cracked bridge images. A similarity measure is adapted in this thesis. Thus, a test image is transformed into the new domain and a new dimensional space M' is obtained and the distance δ_i between the test image projection and the training set projections is computed. If the distance δ_i is small, then the images are greatly correlated and a decision is made on which is the most correlated image from the training set. The following figure 6 illustrates the idea graphically.

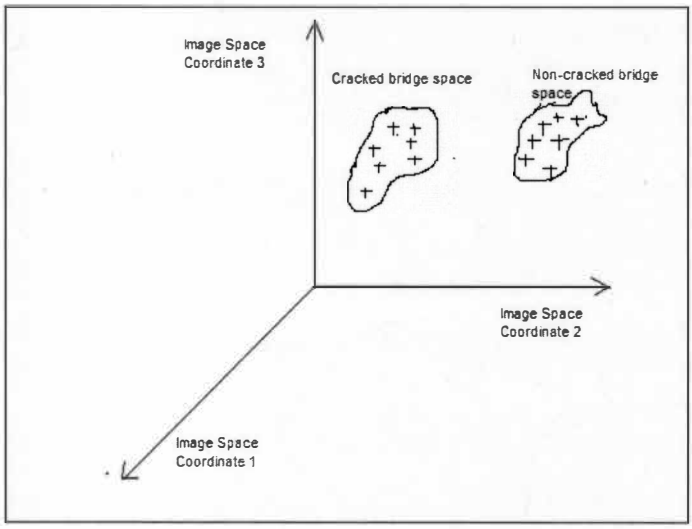


Figure 6: Image space and bridge space representation

Using eigen vectors a new basis for representing images is created where images can be represented as co-ordinates in a much smaller dimensional space. Bridge images with cracks will be located nearer to each other than the bridge images without cracks in this multi-dimensional space because of their similarity. At this level the task of detecting cracks becomes merely a matter of determining the location of the test image vector in a lower dimensional space relative to image vectors of recognized bridge images with cracks.

Since PCA is traditionally used to reduce the dimensionality of a data set in which there are a large number of interrelated variables and at the same time preserving the variation that existed in the original data. In this thesis, PCA is used to identify clusters in an effort to recognize a cracked bridge image from a non-cracked one. PCA basis vectors are computed from a set of training images M. The average of each image in the set is

computed and subtracted from each of the corresponding image in the training set (19).

The covariance matrix C is generated for the training set and the principal components of the covariance matrix shown in equation 4 are computed by solving the equation 6

$$C = 1/M \sum_{i=1}^{M} A_i A_i^T \tag{4}$$

C is the covariance matrix of all the mean normalized vectors in a training set, A_i is the mean normalized column vector of an image in the training set and M is the number of images. In the above equation 4 the mean normalized column vector A_i is obtained using the following equation 5

$$A_i = I_i - I_{ave} \tag{5}$$

Where I_i is an image vector and I_{ave} is the average image.

$$C\alpha_i = \lambda_i \alpha_i \tag{6}$$

Where α_i is the orthonormal eigenvector and λi is the corresponding eigen value. The projections are calculated using equation 7 of the data into the new space that are associated with the largest eigen value is related to the largest variance of the image. The smallest eigen value is associated with the least variance of the data that can be dropped and the reduction in the data dimensionality is established. Euclidean distance as given in the equation 8 is used to for measuring the distance between the Test image projection

and the training set image projection in the new multi-dimensional space.

$$Q_i = \alpha_i^T A_i \tag{7}$$

Q_i is the projection of each mean normalized vector onto the dominant eigen vector, α_i is the i^{th} orthonormal eigenvector, A_i is the mean normalized column vector of each image in the training set.

These projections are arranged to find the lesser dimensional vector space where all training set image projections of the bridge images with cracks are present.

$$\delta_i = \sqrt{\sum_{j=1}^{E} (Q_j - P_j)^2} \tag{8}$$

Where Q_j = Coordinate of the projection of training set image in dimension j, P_j = Coordinate of the projection of test image in dimension j, E = the number of eigen vectors considered for building the training set. δ_i = Euclidean distance between training set image projection and test image projection where i=1, 2, … M. The above model is illustrated graphically in the following figures 7 and 8.

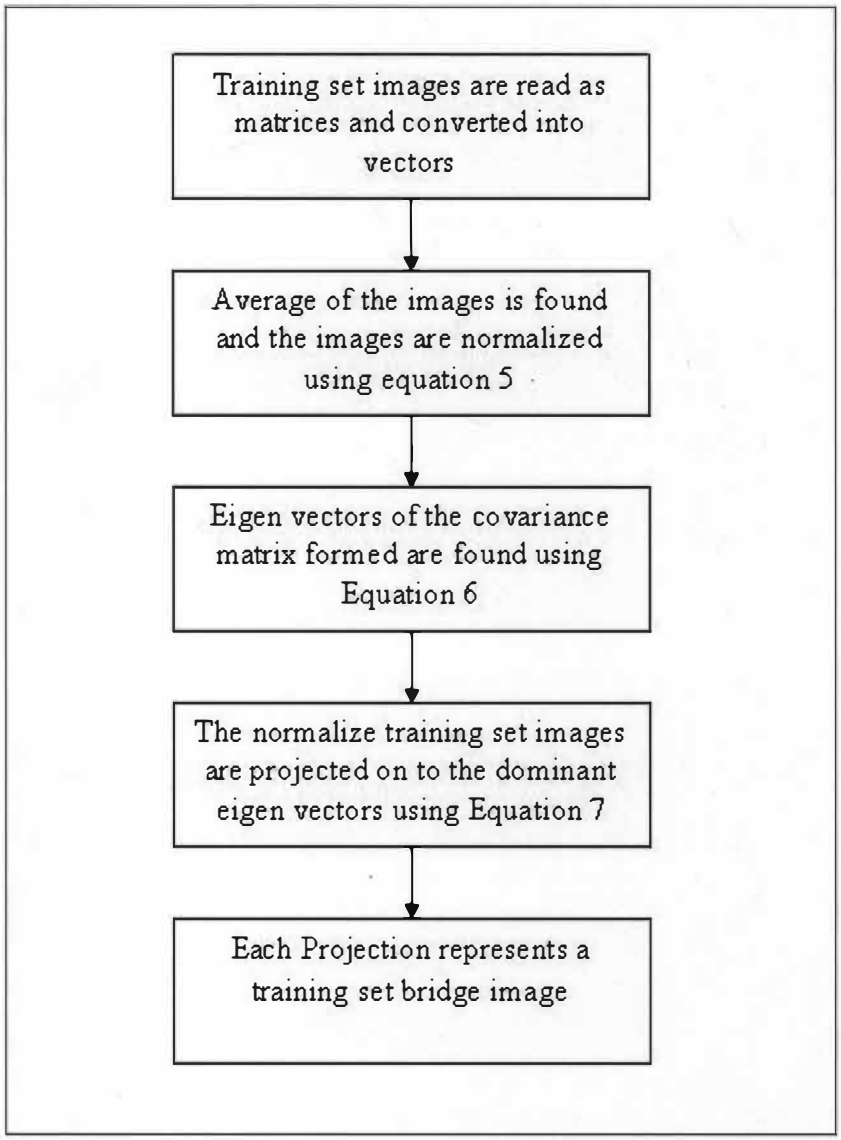


Figure 7: Schematic diagram to illustrate the preprocessing part of the algorithm

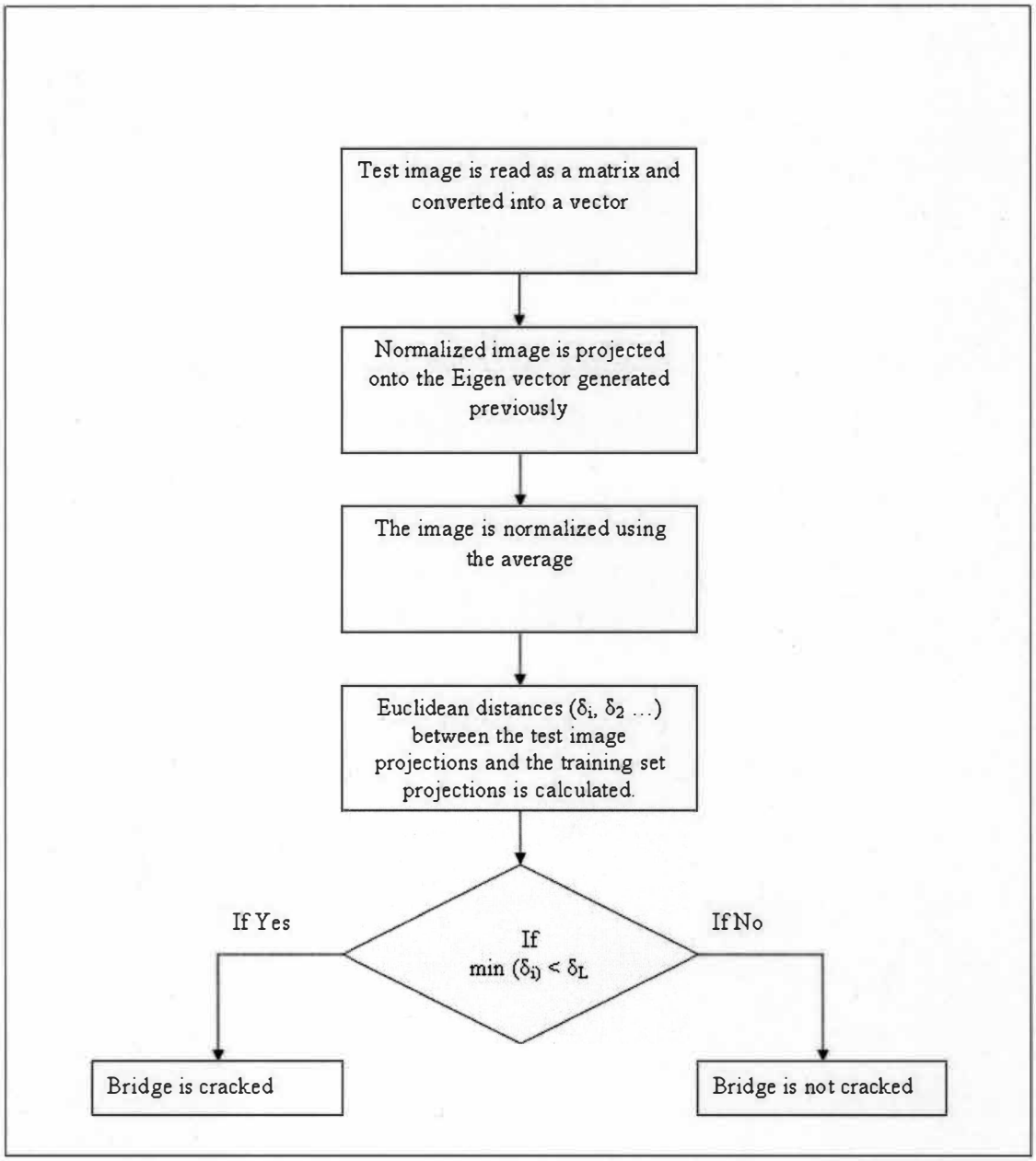


Figure 8: Schematic diagram to illustrate crack detection

Note: δ_L is the threshold for the distance.

3.2 Linear Structure Modeling

There are many approaches that have been used for line detection in an image. Like Hough transform can be used to detect lines in an image. Here in this paper convolution based method is used to find lines in the concrete bridge image then filter the unwanted linear structures using a smoothing filter and then applying principal component analysis (PCA) to detect cracks in them (23).

Several different approaches have been proposed for detection of lines for image analysis. Before using principal component analysis line detection can be used to improve the results for detection of cracks in the concrete bridge images. The importance of linear structure detection is often underestimated. But the need for line detection is as important as the edge detection. Several different approaches have been described for detection of lines (3). There are various methods to detect lines such as directional morphology, curvi-linear structure detection, simple orientated bins, a directional line operator, and directional second order Gaussian derivatives, other methods based on flow fields, and also using the information from the Fourier transform.

In convolution based techniques four 5X5 masks are used namely vertical mask, horizontal mask, 45° and -45°. Then the images are convoluted with the mask using the equation 9.

$$R(n_1, n_2) = \sum_{k1=-2}^{2} \sum_{k2=-2}^{2} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2) \quad (9)$$

Where R (n_1, n2) = Resultant image, a (k_1, k_2) = mask, and b (n_1-k_1, n_2-k_2) = input image. If a vertical line detector mask is used then the longitudinal cracks are highlighted in the resultant image. The vertical line detector mask of size 5X5, which found suitable, is given in (10)

$$a(k_1, k_2) = \begin{bmatrix} -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \end{bmatrix} \quad (10)$$

$$a(k1, k2) = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 4 & 4 & 4 & 4 & 4 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad (11)$$

If a horizontal line detector mask is used then the transverse cracks are highlighted in the resultant image. The horizontal line detector mask of size 5X5 is given in (11). Similarly there are 45° and -45° masks given in (12) and (13) to detection oblique cracks.

$$a(k1,k2)=\begin{bmatrix} -1 & -1 & -1 & -1 & 4 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ 4 & -1 & -1 & -1 & -1 \end{bmatrix} \tag{12}$$

$$a(k1,k2)=\begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix} \tag{13}$$

But the resultant images will contain unwanted lines or the weak lines, which are not cracks. In order to remove these weak lines an averaging filter is used. Then the resultant images are trained by using PCA. Next, the line strength images are either identified as cracked or non-cracked based on the distance threshold determined. The Euclidean distance between the projection of the test image and the projections of all the training set bridge images which are cracked is found and also the Euclidean distance between the projection of the test image and the projections of all the training set bridge images which are not cracked is found. The smallest distance is found to be the closest match for the input test image. And it is found whether the matched image belongs to the cracked training set or in the non-cracked one. Based on this condition the test image is identified as either cracked or non-cracked.

In the principal component analysis (17), the PCA basis vectors are computed from a set of training images M. The average of the image is computed and subtracted from the image. This is repeated for all the images in the training set. The covariance matrix C that is the transformation matrix is generated for the training set.

And as explained previously the principal components of the covariance matrix are computed by solving the equation 6. The model is explained graphically in figure 9, 10 and 11.

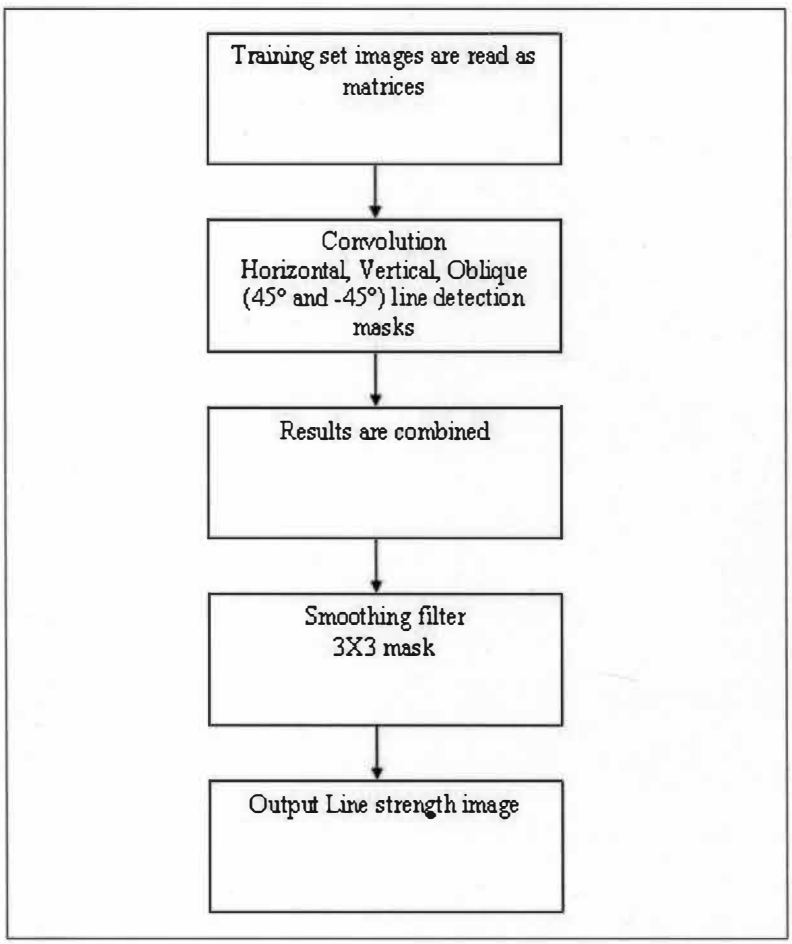


Figure 9: Schematic diagram to illustrate process of detection of cracks in concrete bridges

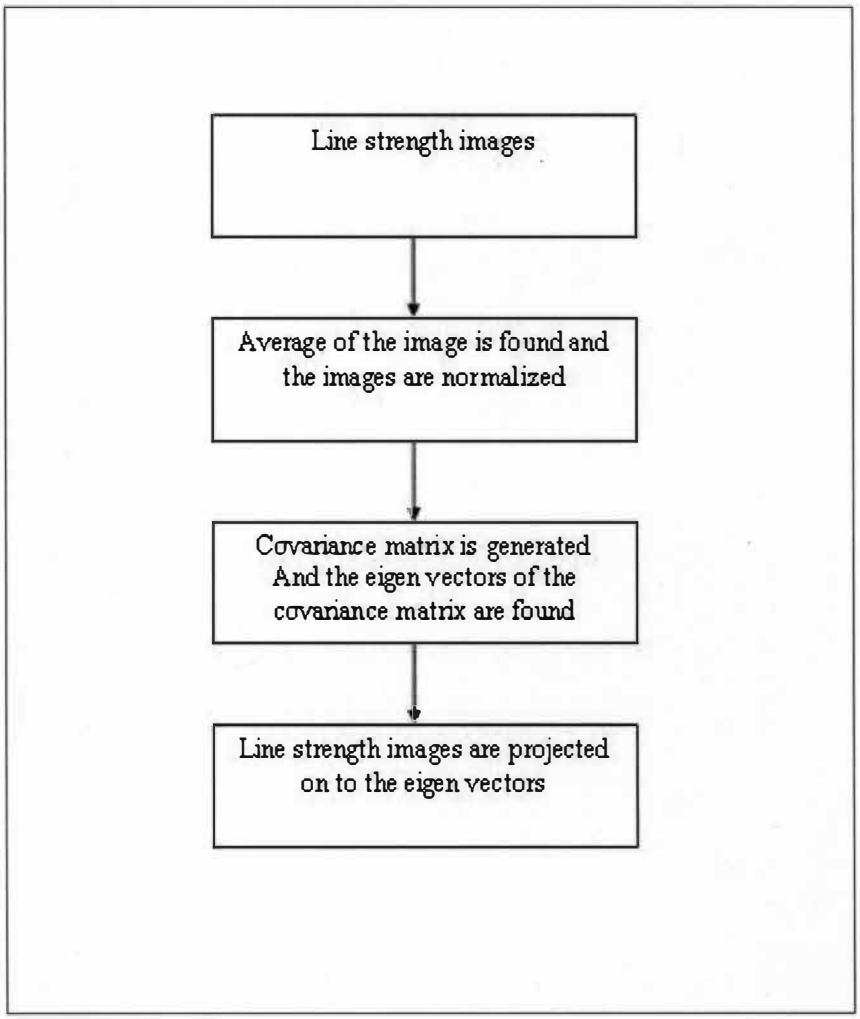


Figure 10: Schematic diagram to illustrate PCA model

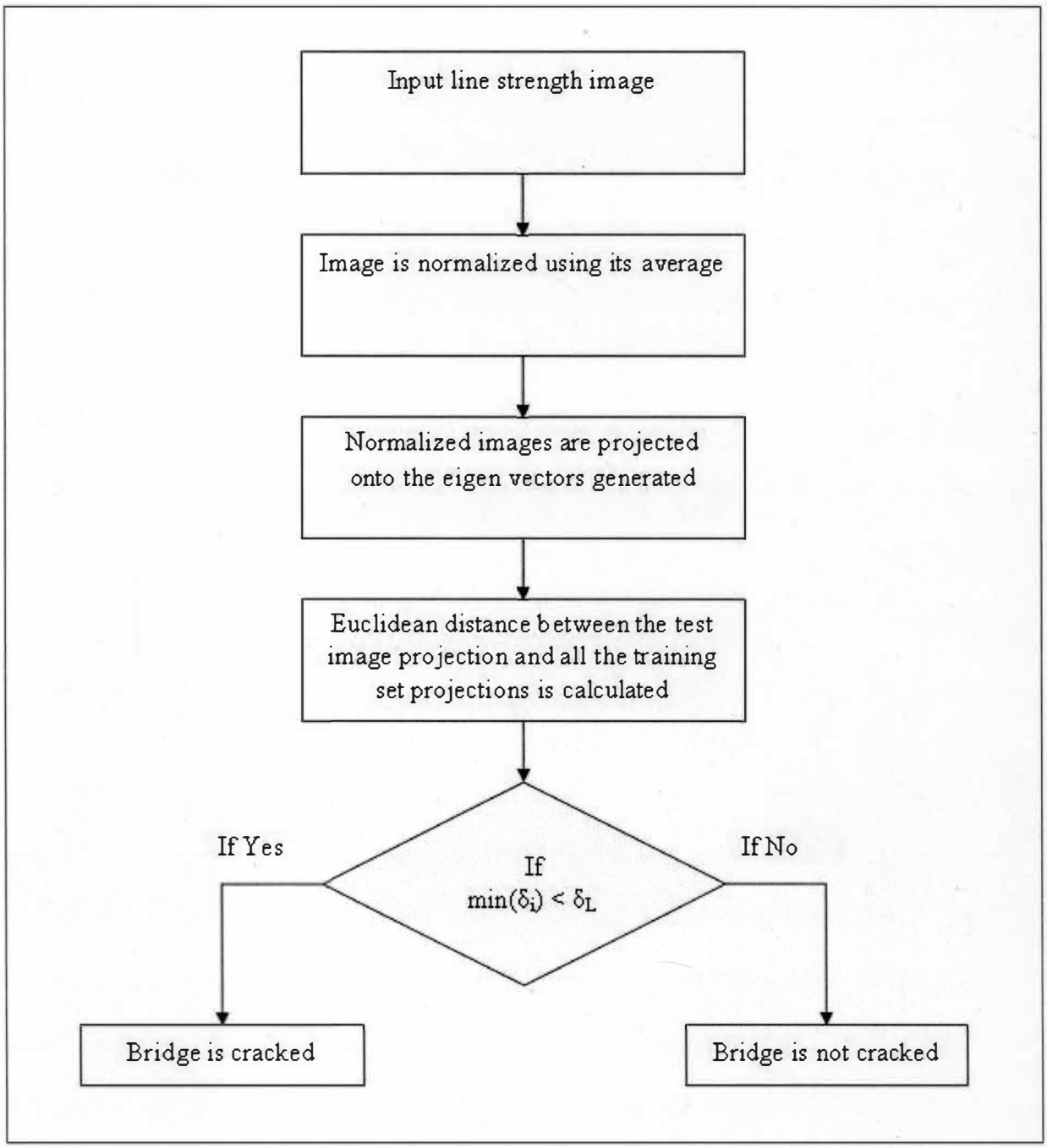


Figure 11: Schematic diagram to illustrate the integrated algorithm

3.3 Block Processing

Here in this analysis we develop an algorithm where we divide each image in the training set into 16 equal square blocks. To accomplish this since the original image is of size 640 X 480 so it had to be resized into a square image of size 480 X 480 and then divided into 16 equal blocks of size 120X120. Initially the algorithm was tried for 5 training set images and then the analysis was extended to 15 training set images. Now after block processing the four-convolution masks horizontal, vertical, Oblique (45° and -45°) were applied to all the blocks separately. All the results of the convolution masks were combined. The resultant matrix contains all the results from the four convolution masks. Since the results obtained consists of unwanted weak linear structures. A 3X3-averaging filter was used to remove most of the weak linear structures and just the strong linear structures of interest were retained in the bridge image. Now the all the resultant blocks are stored again in a single matrix sequentially.

The Principal component analysis is applied to these blocks. Here each image block is converted to a vector and then arranged column wise to form a matrix containing all the block images. Here the average of individual block is calculated and then using the average each corresponding block is normalized using equation 5 and then stored in a large matrix. Using the normalized blocks the covariance matrix is determined using equation 4. Where b_i is the each block in the training set and b_{ave} is the average of each block. Now once the covariance matrix is formed using the normalized blocks the eigen

vectors and corresponding eigen values of the covariance matrix are generated using equation 6.

Once the eigen vectors and the corresponding eigen values of the covariance matrix are found. The dominant eigen vectors have to be chosen according to the largest eigen values. Therefore the eigen values are arranged in the descending order. Half of the eigen vectors are corresponding to the largest eigen values. Now the normalized image blocks have to be projected onto the dominant eigen vectors to obtain the projections of each image block on a lower dimensional vector space as shown in equation 7. So this leads to dimensionality reduction. These projections are stored. Now next part of the algorithm is the detection part. Here the test image is read as a matrix and then resized into a square matrix. Similarly as explained previously the square image are divided into 16 blocks. These blocks are converted to vectors and stored column wise in a single matrix. Now the average of each block is found and then the blocks are normalized. These normalized blocks are projected onto the same dominant eigen vectors generated previously. So the projections of each individual 16 blocks are obtained in the same lower dimensional vector space. So now to determine whether the block is cracked or not. Euclidean distance measure is used to find the distance between projections of the 16 blocks (test image) and all the projections of the blocks in the training set using equation 8. We use the least distance obtained between the blocks as the means to determine whether the block is cracked or not. If the closest match of the block is to a cracked block in the training set then the block is cracked otherwise if the closest match of the block is to a non-cracked block in the training set then the block is not cracked. The whole process of the algorithm is shown schematically in figures 12, 13 and 14.

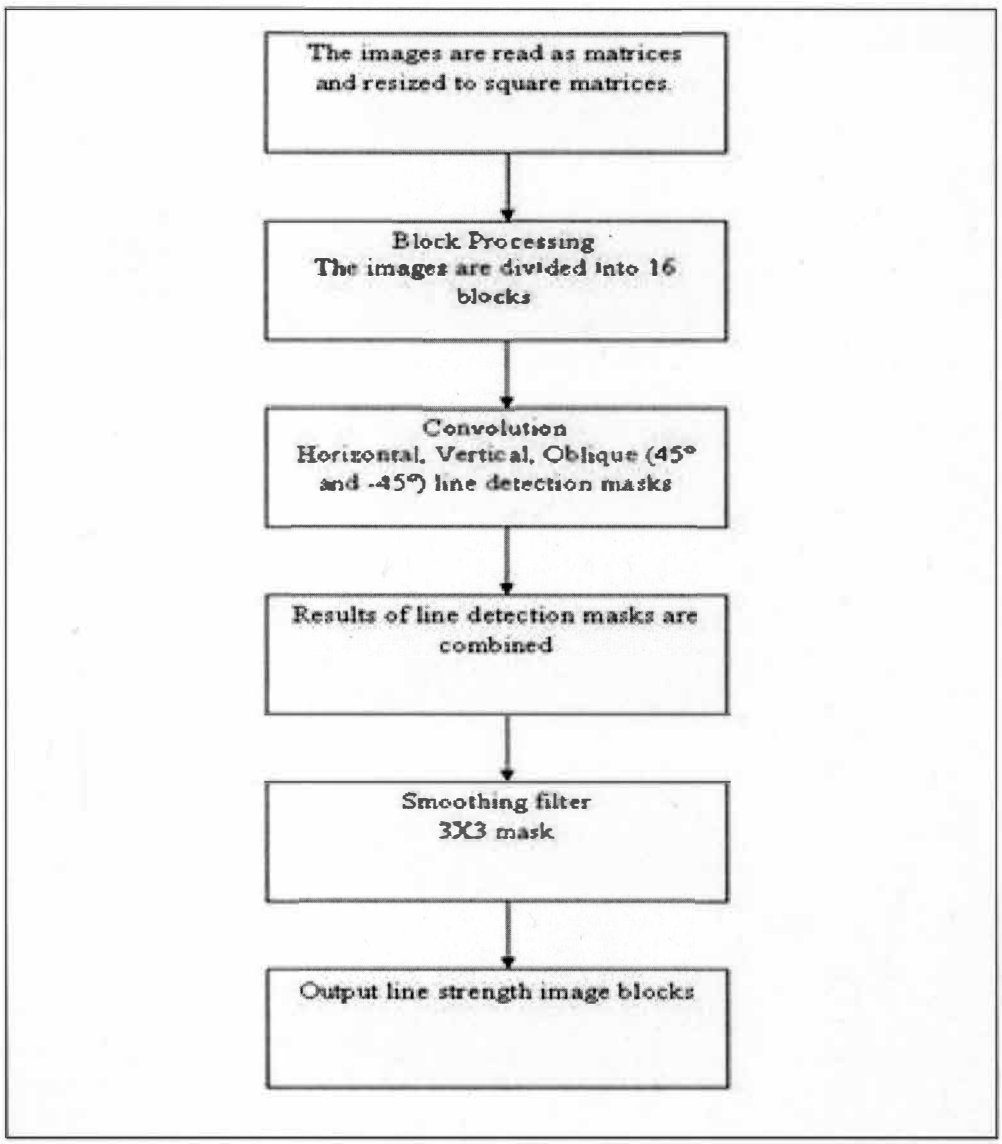


Figure 12: Schematic diagram to illustrate the preprocessing part of the algorithm using block processing

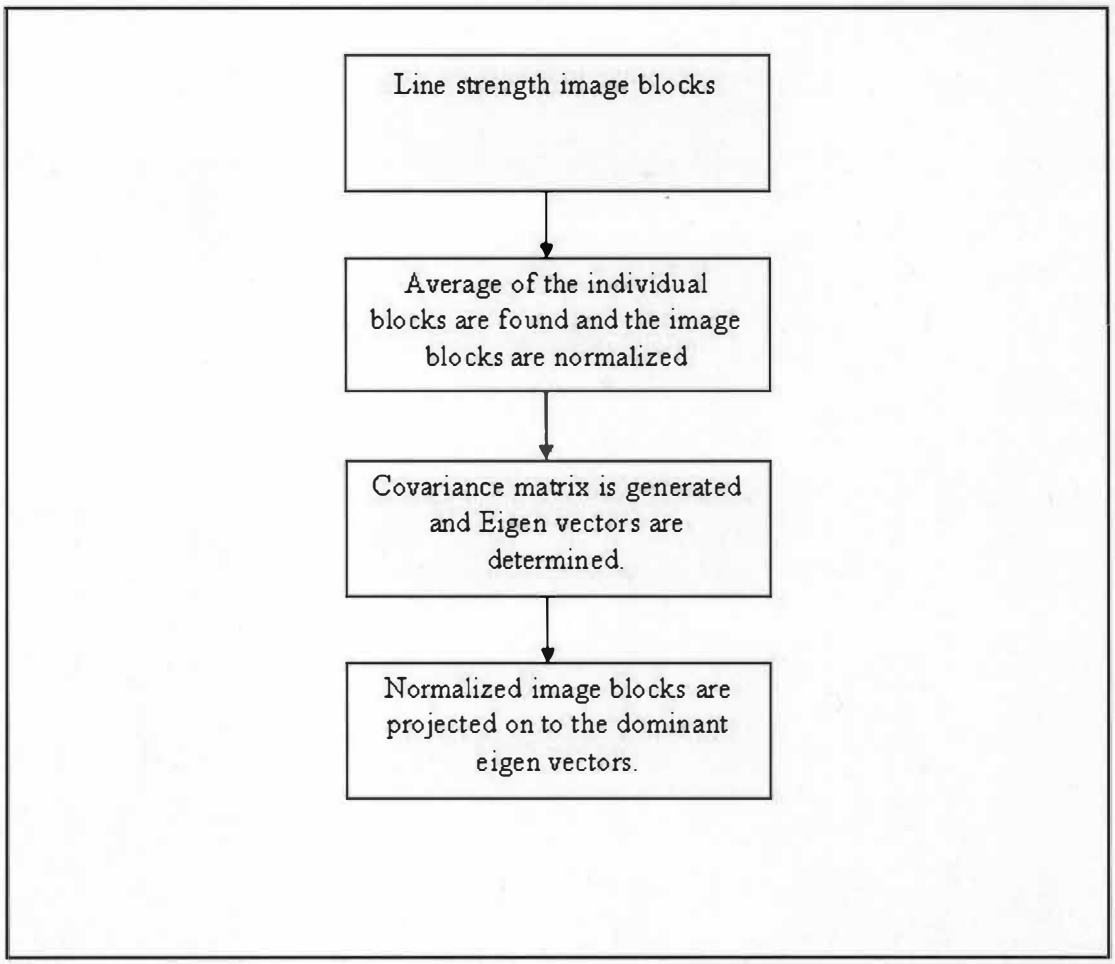


Figure 13: Schematic diagram to illustrate PCA model

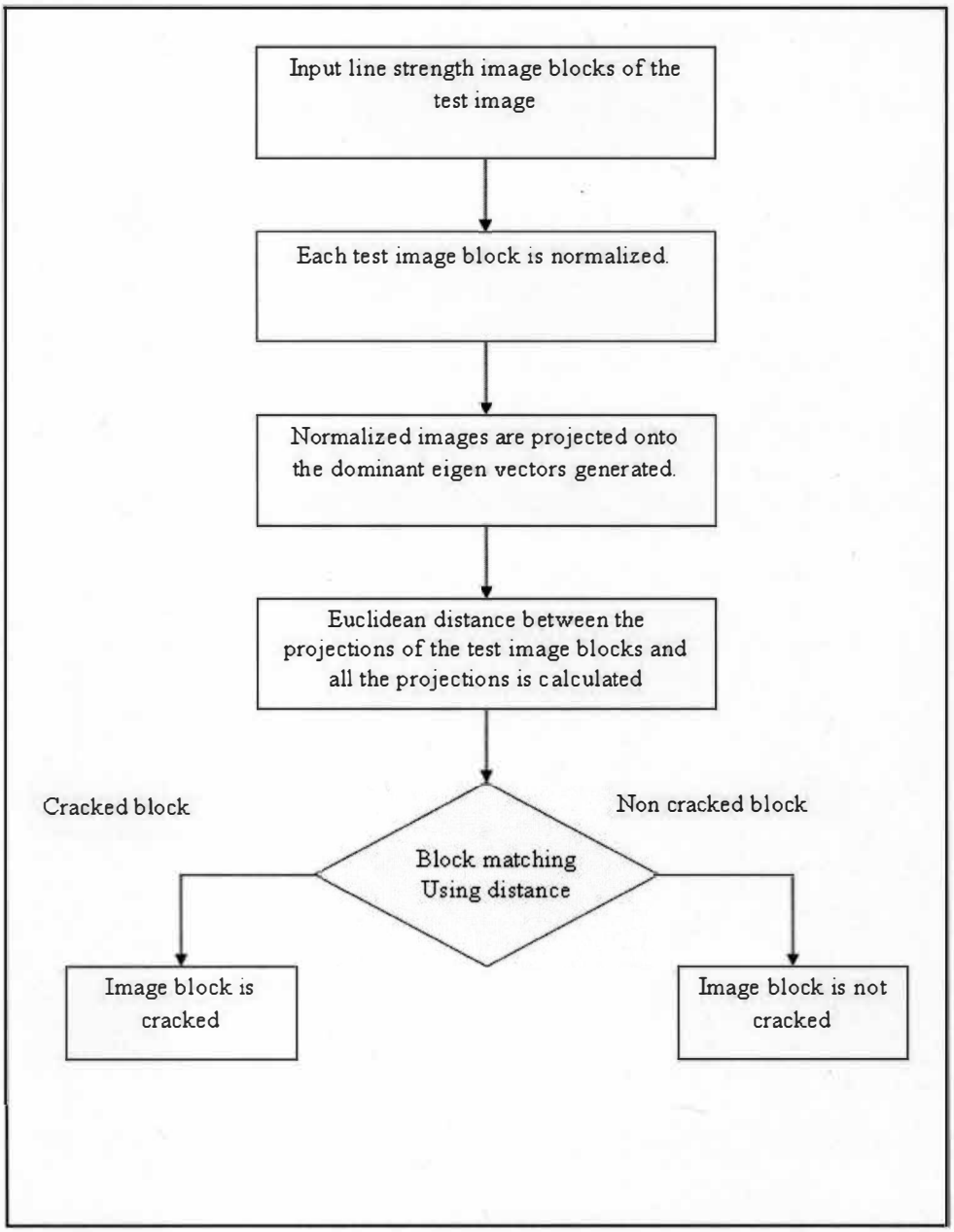


Figure 14: Schematic diagram to illustrate the integrated algorithm

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

4.1 Principal Component Analysis (PCA)

Bridge images containing 20 bridges with crack and 20 bridges without crack are used. To generate the training data, twenty images from the cracked images and twenty images from the crack free image were used. The 5 images cracked and 5 non-cracked images were used as test images. The results are tabulated as shown in tables 1, 2 and 3. The table 1 was generated, which contains the results obtained by training the 20 cracked bridge images and using the 5 cracked bridge images as the test images. The cracks were identified in just one test image out of 5 test images. The table 2 was generated using the results obtained by training the 20 non-cracked bridge images and using 5 cracked bride images as the test images. The results in table 2 indicate that 5 images are misidentified as non-cracked while 20 bridge images were identified as cracked. Of those identified as non-cracked images when they are actually cracked, two were blurred images and two had the bridge showing as part of the image. Others have hairline cracks in turn lead to misidentification of crack in the bridge images. The table 3 shows the results obtained when the non-cracked bridge images were used as the test images and the non-cracked bridge images were trained. The dominant eigen images generated are shown in the figures 15, 16, 17, 18, 19.

Images	Min (δi)	Identified image	Result
1	0	1	Cracked
2	0	2	Cracked
3	0	3	Cracked
4	0	4	Cracked
5	0	5	Cracked
6	0	6	Cracked
7	0	7	Cracked
8	0	8	Cracked
9	0	9	Cracked
10	0	10	Cracked
11	0	11	Cracked
12	0	12	Cracked
13	0	13	Cracked
14	0	14	Cracked
15	0	15	Cracked
16	0	16	Cracked
17	0	17	Cracked
18	0	18	Cracked
19	0	19	Cracked
20	0	20	Cracked
21	3.86	No match	Non-cracked
22	10.39	No match	Non-cracked
23	10.14	No match	Non-cracked
24	2.59	20	Cracked
25	4.88	No match	Non-cracked

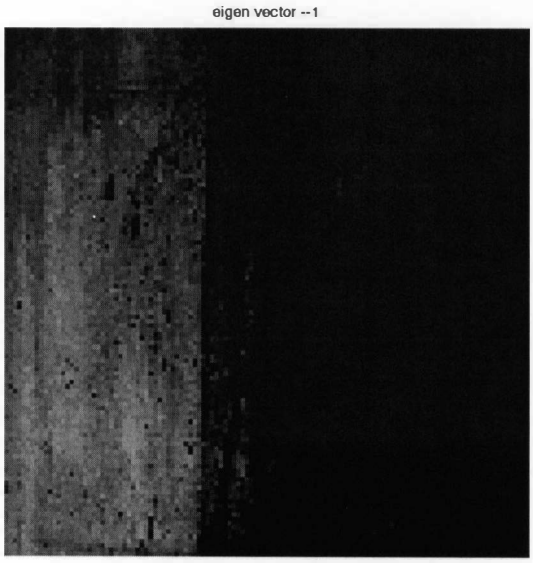
Table 1: Results obtained when cracked bridge image as test data and training set containing cracked bridge images

Test image	Result	Min (δi)
1	Cracked	4.7795
2	Cracked	8.7023
3	Not Cracked	1.1639
4	Cracked	2.2762
5	Not Cracked	0.72118
6	Cracked	2.6253
7	Cracked	7.6871
8	Cracked	11.098
9	Cracked	2.1590
10	Not Cracked	1.1216
11	Cracked	6.9414
12	Cracked	4.1175
13	Cracked	5.8457
14	Cracked	3.4713
15	Cracked	7.9049
16	Cracked	2.9980
17	Cracked	2.0378
18	Cracked	5.1108
19	Cracked	2.8776
20	Cracked	3.3165
21	Not Cracked	0.92939
22	Cracked	10.762
23	Cracked	9.0272
24	Cracked	3.3241
25	Not Cracked	1.1863

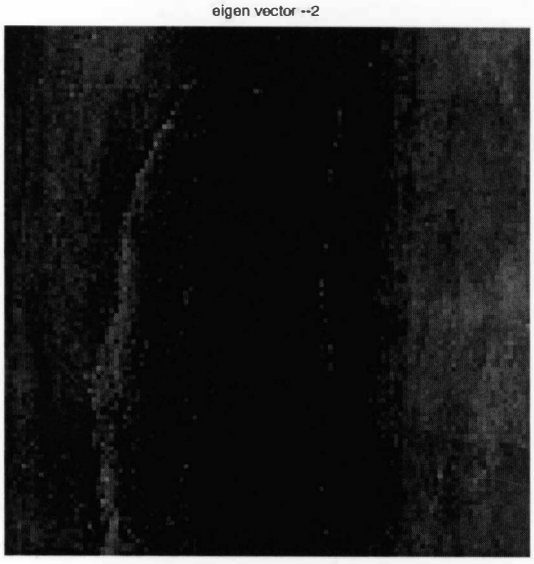
Table 2: Test image with crack and training set containing the non-cracked bridges

Test image	Result	Delta
46	Not cracked	1.3571
47	Not cracked	0.34808
48	Cracked	3.4988
49	Not cracked	0.67077
50	Cracked	3.8076

Table 3: Test image with no crack and training set containing non-cracked bridge images

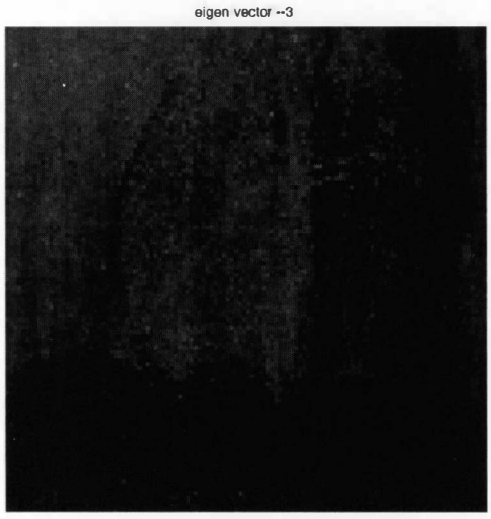


(a) eigen image 1

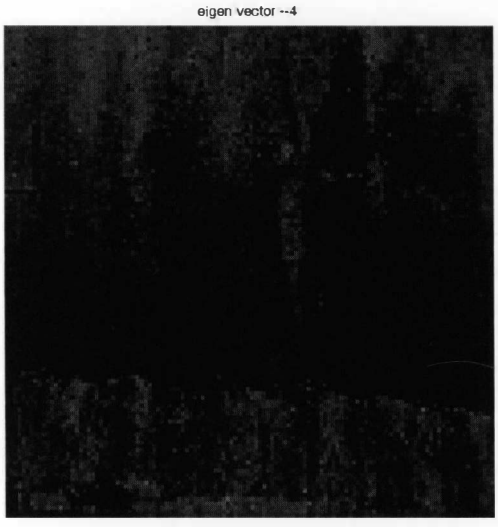


(b) eigen image 2

Figure 15: Dominant eigen images (a) and (b)

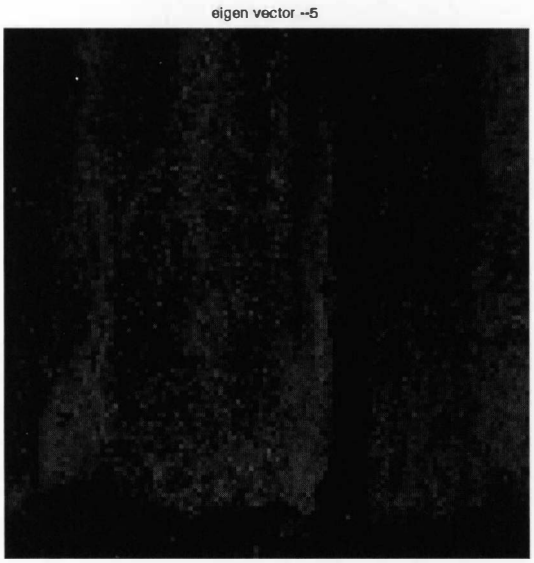


(c) eigen image 3

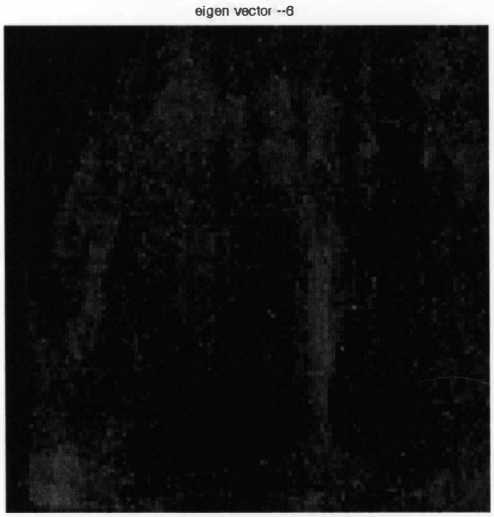


(d) eigen image 4

Figure 16: Dominant eigen images (c) and (d)

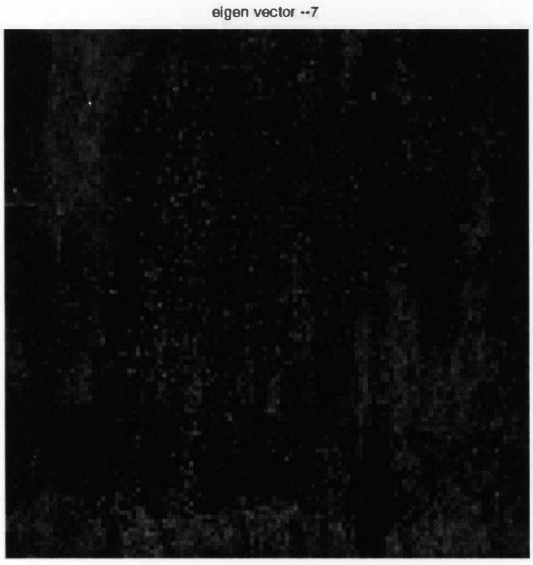


(e) eigen image 5

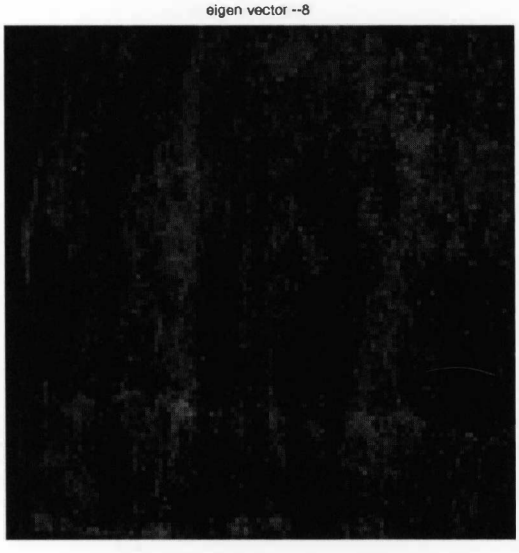


(f) eigen image 6

Figure 17: Dominant eigen images (e) and (f)

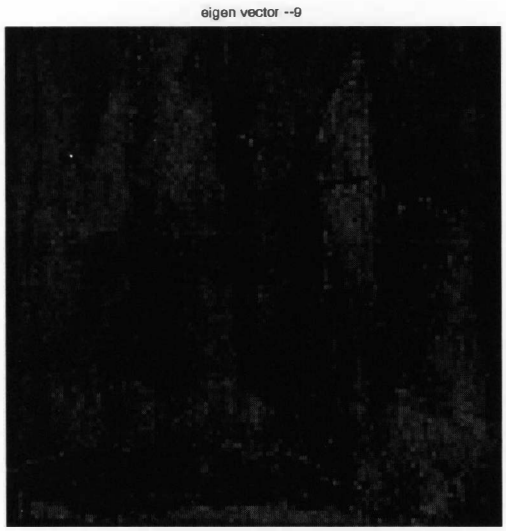


(g) eigen image 7

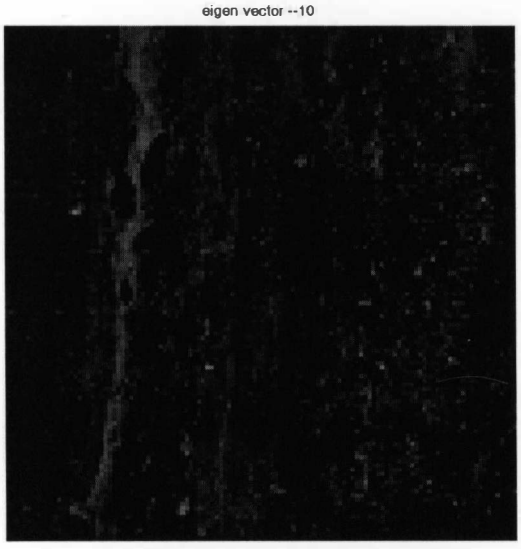


(h) eigen image 8

Figure 18: Dominant eigen images (g) and (h)



(i) eigen image 9



(h) eigen image 10

Figure 19: Dominant eigen images (g) and (h)

4.2 PCA and Linear Structure Modeling

Here again data set containing 25 bridges with cracks and 25 bridges without cracks are used in the analysis. Twenty images from the cracked images and twenty images from the crack free images were used as training images. The other 5 images from both cracked and non-cracked images were used as test images. The algorithm is compared to the crack detection using the PCA algorithm.

Training involved using PCA to create a database of images (image vectors in a new lower dimensional space). So that when the test image is read and converted into image vectors in the same lower dimensional space, which is ready to be matched with any of the images in the database, which are created by PCA in the training stage.

Case 1: 20 cracked bridge images were trained using PCA. They are trained without using the linear structure detection. Then all the images are used including the five cracked bridge image as the test images. The delta value, which is the Euclidean distance (δ_i) between the test image projection and training set images projections in the new lower dimensional space, is calculated. Then each min (δ_i) is plotted against the 25 bridge images. The results are plotted as shown in the figure 8 and also shown previously in the table 1.

Case 2: The 20 cracked bridge images are trained using PCA, but here linear structure detection is used prior to PCA. The 5 cracked bridge images and 25 non-cracked bridge images are used as the test images. Linear structure modeling is applied to all the test images. The results are shown in the Figure 14. One can observe that the Euclidean distance min(δ_i) between projections is zero, if the test image is in the training set. But when the new test image is introduced we see the distances delta with non-zero values as

expected. Observing the Euclidean distances a suitable threshold of 3.1 is chosen to find the closest match in the training set created and to determine that the bridge image is cracked or non-cracked. The threshold is calculated by observing the graphs in the figures 20 and 21.

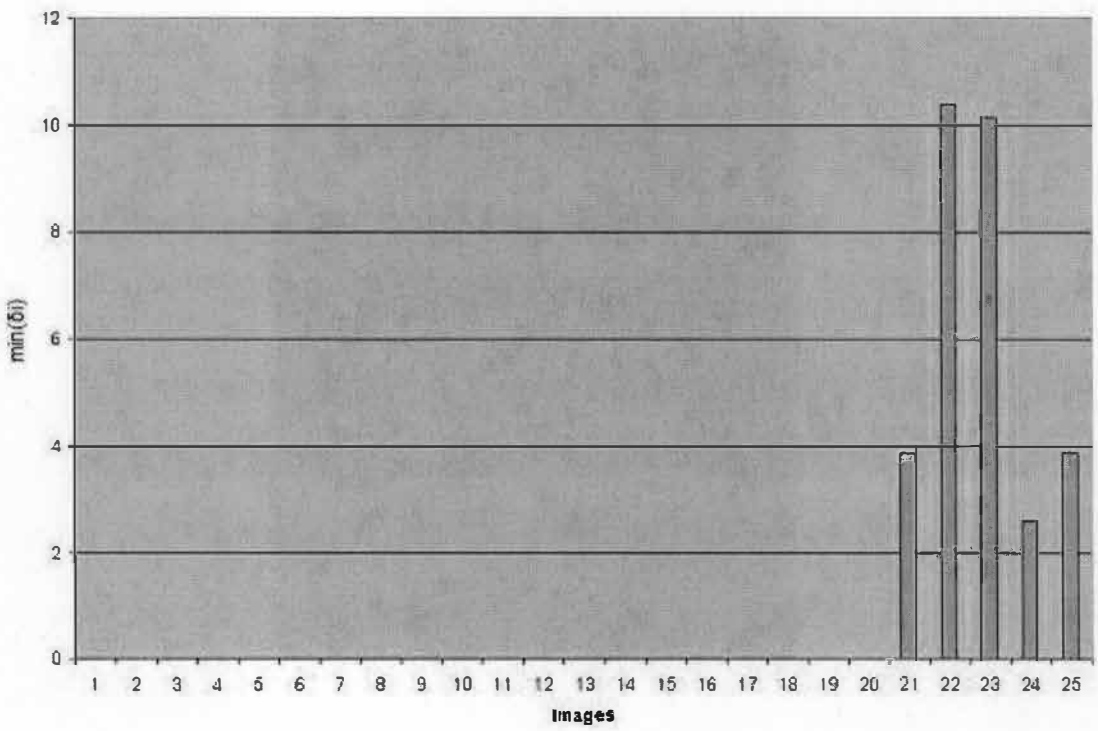


Figure 20: Plot of min (δ_i) values for case 1

Similarly figure 14 show the results for crack detection using the linear structure modeling. Here images 21 and 22 get identified as cracked but images 23, 24 and 25 get misidentified as non-cracked as they lie above the threshold 3.1. This shows that algorithm needs further improvement.

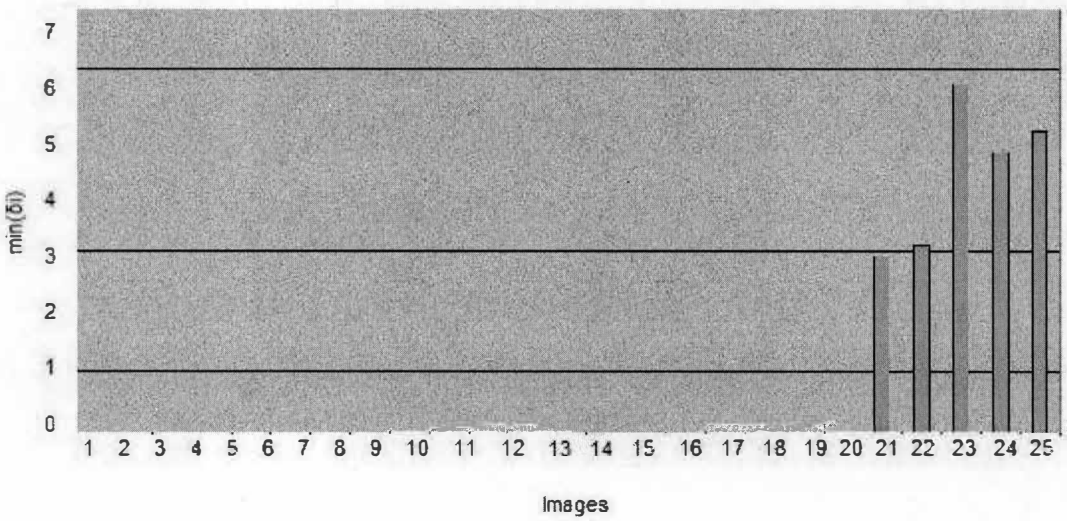


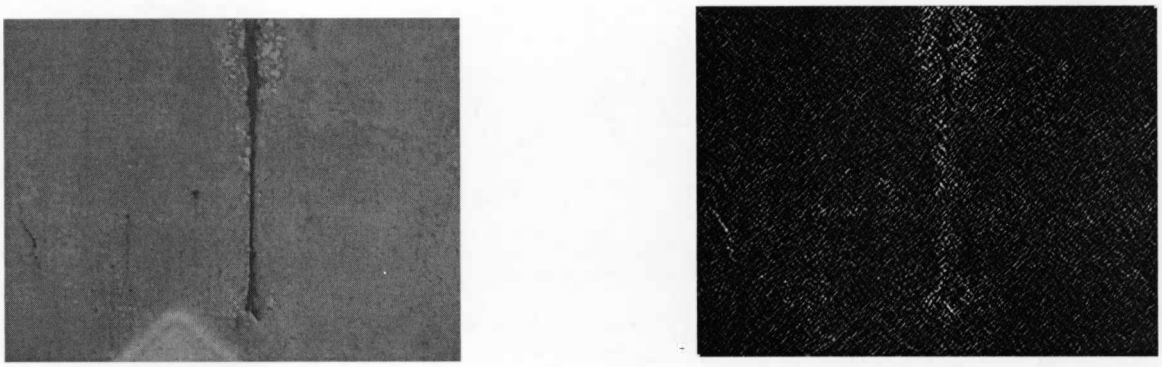
Figure 21: Plot of min (δi) values for case 2

If the Euclidean distance, min (δ_i) lies below the threshold then the input test image is detected as cracked and if the min (δ_i) lies above the threshold then the input test image is detected as non cracked. If we observe the results in figure 14 out of 5 test images 2 get identified as cracked using the algorithm developed. Using 25 non-cracked bridge images as the test images generates the table 4. If we observe the table 4 except for images 26, 32 and 47 all the other bridge images are identified correctly as non-cracked. But if the results are compared with the previous algorithm results we can say that the results have improved if linear structure modeling is used prior to the principal component analysis.

Images	Min (δi)	Identified image	Result
26	3.88	14	Cracked
27	4.08	20	Not cracked
28	4.87	20	Not cracked
29	6.91	1	Not cracked
30	3.13	15	Not cracked
31	5.34	1	Not cracked
32	2.8	20	Cracked
33	3.86	15	Not cracked
34	4.52	20	Not cracked
35	6.14	15	Not cracked
36	4.33	14	Not cracked
37	3.86	15	Not cracked
38	3.52	15	Not cracked
39	5.34	15	Not cracked
40	4.68	20	Not cracked
41	3.4	15	Not cracked
42	4.35	15	Not cracked
43	6.68	12	Not cracked
44	4.73	20	Not cracked
45	5.73	14	Not cracked
46	2.93	14	Not cracked
47	3.03	20	Cracked
48	3.41	20	Not cracked
49	3.37	15	Not cracked
50	3.4	20	Not cracked

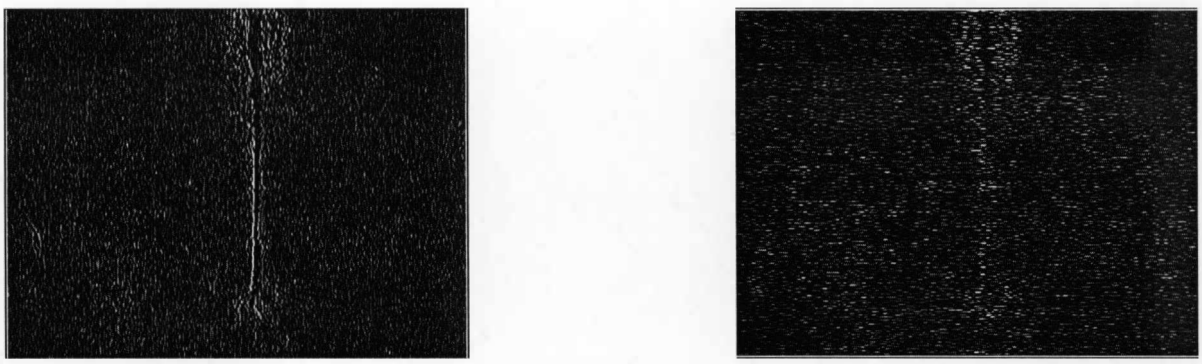
Table 4: Results obtained when non-cracked bridges are used as test data for case 2

The results obtained for linear structure modeling using different line detection masks are shown in figures 22, 23, 24 and 25. Also the 10 dominant eigen images are shown in the figures 26, 27, 28, 29 and 30.



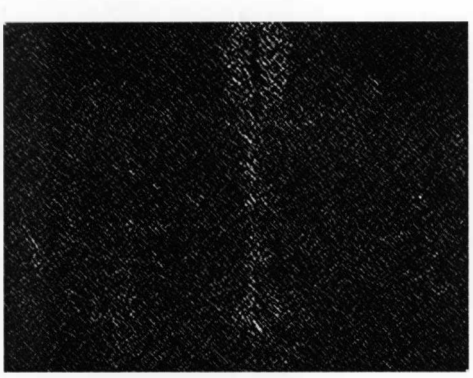
(a) (b)

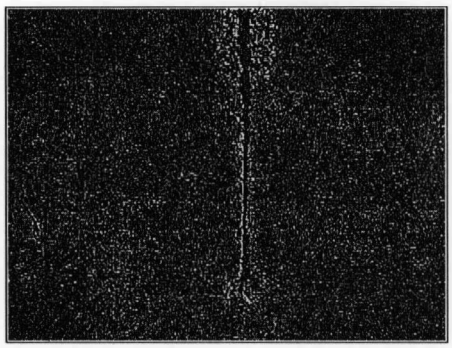
Figure 22: Results obtained for linear structure detection (a) Image 1 in the training set, (b) Horizontal linear detection mask is used



(c) (d)

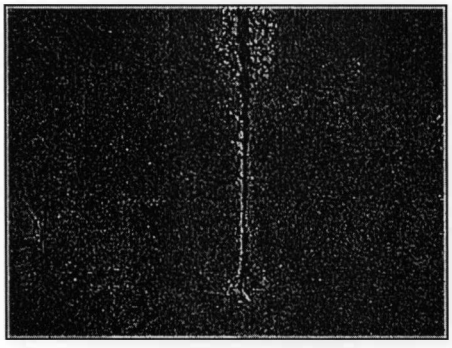
Figure 23: Results obtained for linear structure detection (c) Vertical linear detection mask is used. (d) 45° linear detection mask is used





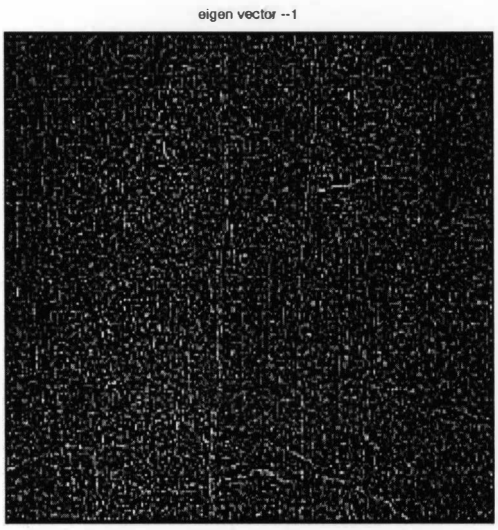
(e) (f)

Figure 24: Results obtained for linear structure detection (e) -45° linear structure detection mask is used, (f) Results b, c, d, e combined

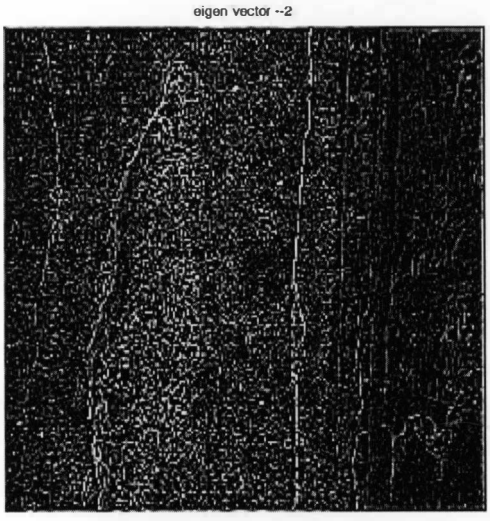


(g)

Figure 25: Results obtained for linear structure detection (g) average filtered image

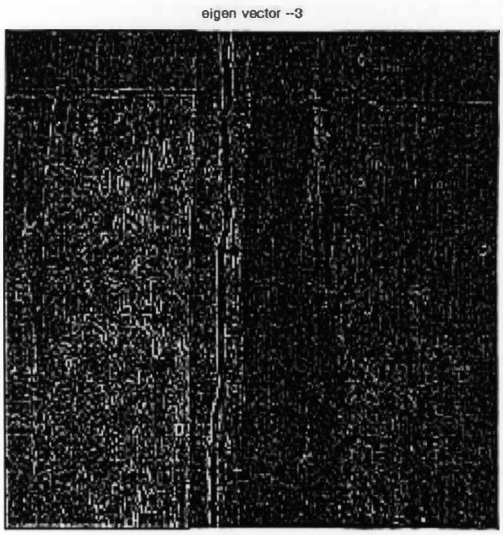


(a) eigen image 1

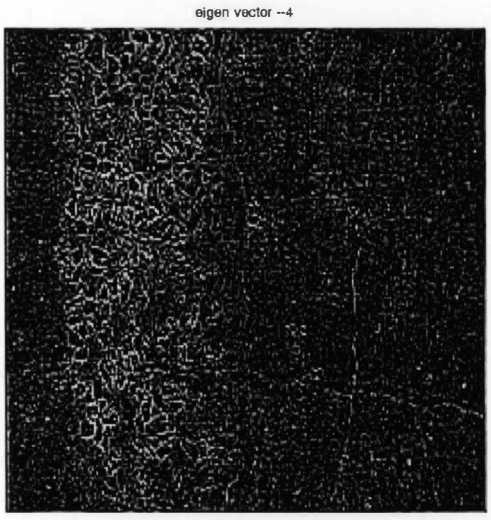


(b) eigen image 2

Figure 26: Dominant eigen images (a) and (b)

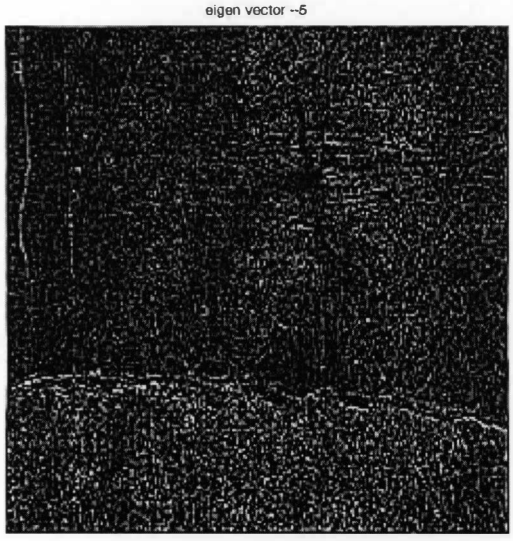


(c) eigen image 3



(d) eigen image 4

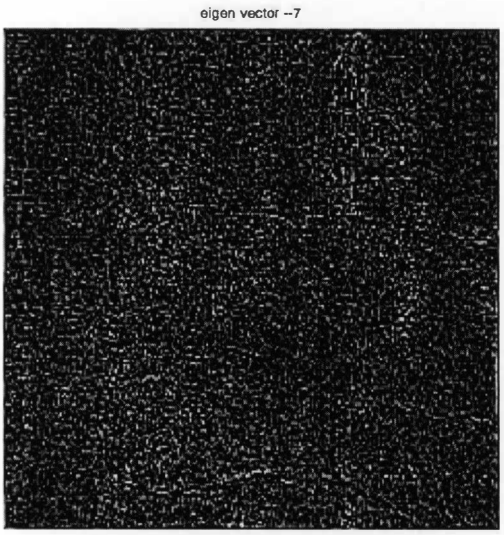
Figure 27: Dominant eigen images (c) and (d)



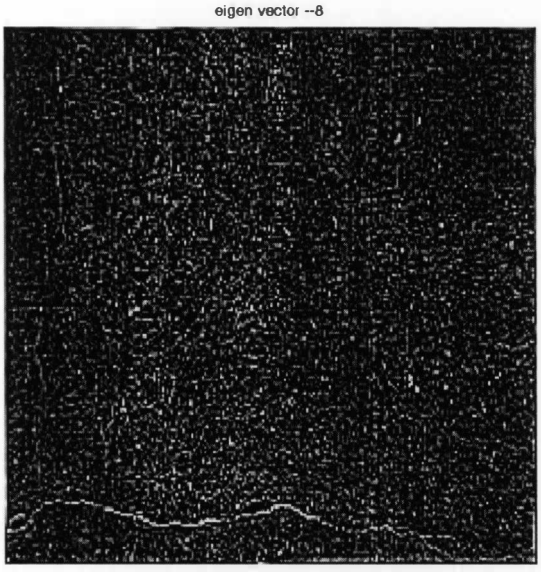
(e) eigen image 5

(f) eigen image 6

Figure 28: Dominant eigen images (e) and (f)



(g) eigen image 7



(h) eigen image 8

Figure 29: Dominant eigen images (g) and (h)

(i) eigen image 9

(j) eigen image 10

Figure 30: Dominant eigen images (i) and (j)

4.3 Distance Measures

In this analysis different distance measures like city block distance, chebyshev distance, minkowski distance and Canberra distance measures are compared with Euclidean distance measure. To compare the distance measures the following analysis was performed. Where 20 cracked bridge images were trained using PCA and linear structure modeling and 5 bridge images with cracks were used as test images. For each test image a bar graph as shown in figures 31, 32, 33, 34 and 35 was plotted to show the comparison. Here the same threshold of 3.1 was used. If the min (delta) is lower than 3.1 the test image is decided as cracked. Otherwise it is decided that the test image is not cracked. From the results it can be seen that chebyshev distance is better compared to other three distance measures namely city block, minkowski and Canberra. But the results obtained are same as that of Euclidean distance measure.

The various distance measures are given by

a) City block distance: This distance measure is the average difference across dimensions. In most of the cases city block resembles the Euclidean distance. The city block distance is given by equation 6.

$$\delta_1(x,y) = \sum_i |x_i - y_i| \tag{6}$$

b) Chebychev distance: This distance measure is used when there is need to differentiate objects based on their dimension in an image. The distance measure is given by equation 7.

$$\delta_2(x,y) = \max_i(|x_i - y_i|) \tag{7}$$

c) Minkowski distance: This distance measure is given by equation 8, $\lambda = 3$ was used in the analysis.

$$\delta_3(x,y) = (\sum_i |x_i - y_i|^{1/\lambda})^{\lambda} \tag{8}$$

d) Canberra distance: This distance measure is given by equation 9.

$$\delta_4(x,y) = \sum_i \frac{|x_i - y_i|}{|x_i + y_i|} \tag{9}$$

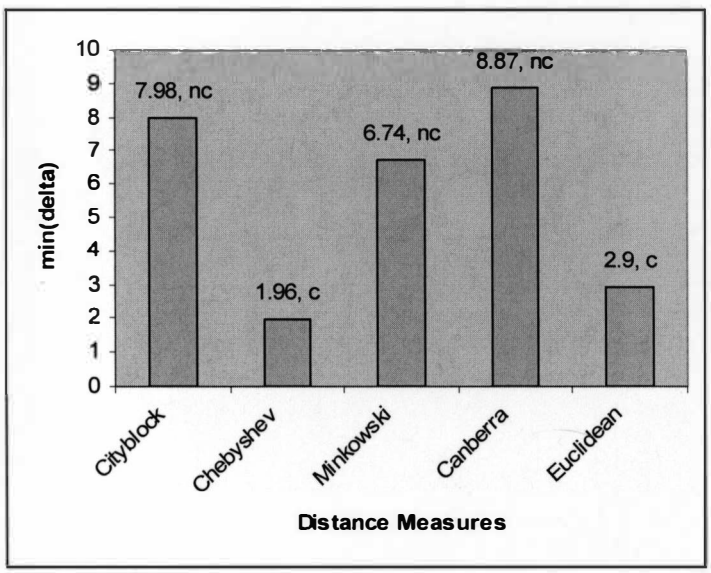


Figure 31: Plot of min (delta) vs distance measures for test image 21, nc= no crack and c=crack

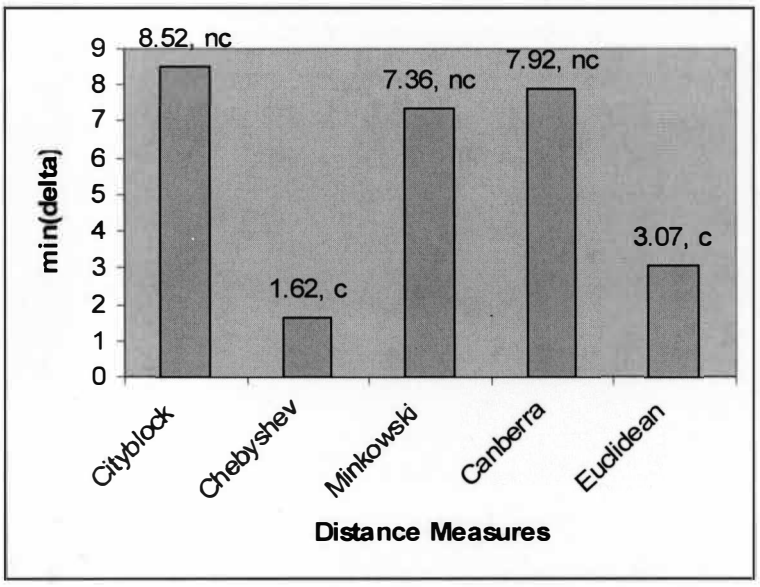


Figure 32: Plot of min (delta) vs distance measures for test image 22, nc= no crack, c=crack

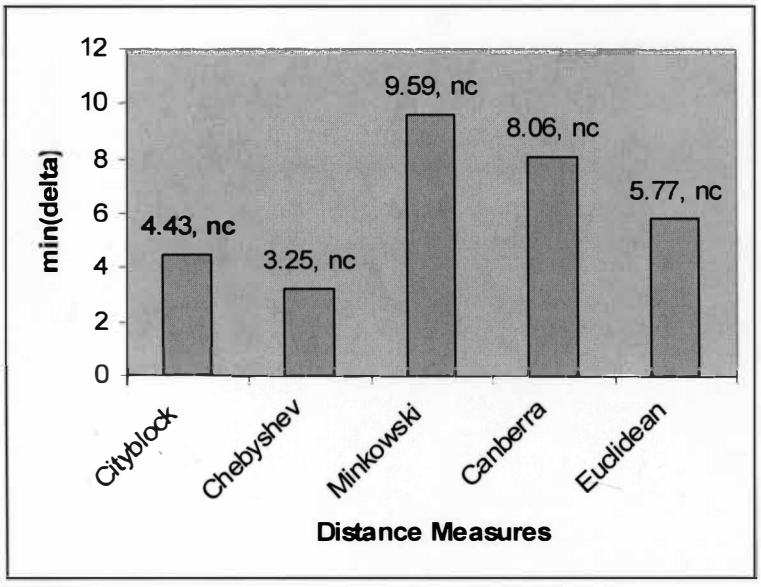


Figure 33: Plot of min (delta) vs distance measures for test image 23, nc= no crack, c= crack

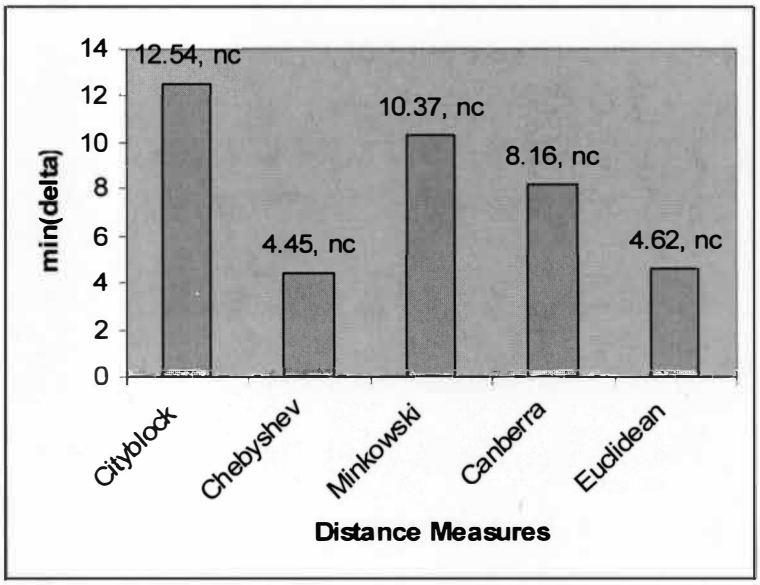


Figure 34: Plot of min (delta) vs distance measures for test image 24, nc=no crack, c = crack

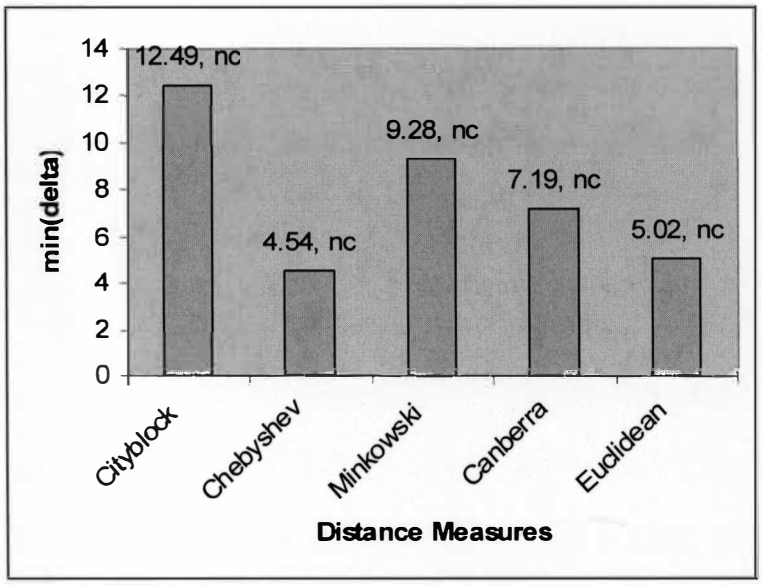


Figure 35: Plot of min (delta) vs distance measures for test image 25, nc=no crack, c=crack

4.4 Block Processing

The results are obtained for three cases. The table 5 is built by the user who identifies the blocks as cracked or non-cracked. So that when the test image is introduced the closest match is found and depending on the matching block being cracked or non-cracked a decision is made. Three cases are considered on the number of images trained.

Case 1: From the data set 5 bridge images with cracks are trained and the remaining 20 bridge images are used as the test images. The table 6 is generated containing the closest match in the training set. Table 7 shows the results. The true identifications are green in color and wrong identifications are red in color. From table 7 it can be seen that indicate inaccuracies. For example in image 6 the crack was not identified in the 5 blocks and also the block 14 was misidentified as cracked. In image 7 the cracks were identified in 4 blocks but 9 blocks were misidentified as non-cracked. The accuracy obtained for true identifications was 55.93%.

Case 2: From the data set 10 bridge images with cracks are trained and the remaining bridge images are used as the test images. The table 8 is generated containing the closest match in the training set. Initially it was thought that the accuracy could be further improved by increasing the training set images. For example in image 11 the crack was identified in just one block, same as that in case 1 but different block. Here the accuracy obtained for true identifications is 45%. There was no improvement in case 2. Instead the accuracy worsened due to the inclusion of 5 more bridge images with crack in the training set. This showed that the result depend on the concrete bridge images with cracks that are trained and not on the number of images.

Case3: From the data set 15 bridge images with cracks are trained and the 5 images were used as the test images. The table 10 is generated containing the closest match in the training set. The results obtained are better than case 1 and case 2. From table 10 it can be seen that in image 16, the block 5 had a part of vertical crack and was correctly identified as cracked. The blocks 9 and 13 were wrongly identified as non-cracked. In image 17, the blocks 7, 10, 13 and 15 were correctly identified as cracked but block 5 was wrongly identified as cracked. In the block 4 of image 18 the horizontal crack was identified, also in blocks 8, 12 and 13 thin cracks were identified but in the blocks 6, 10 and 14 the crack was not identified. In the image 19 block 10 was correctly identified as cracked but some of the blocks like 2, 6 14 the cracks were not identified. In image 20 blocks 5, 9 and 13 were correctly identified as cracked but block 10 was wrongly identified as cracked. The figure 36, 37 and 38 show the closest match for the test image blocks. The results can be improved if the numbers of cracked bridge images are increased in the training set. Accuracy of 60% for true identication was obtained for true identifications. But due to the limitation of matlab the program crashes if we train more than 15 bridge images. This leads to future work were the algorithm can be implemented on real time using c language and the training set can be increased to gain more accuracy. The accuracy obtained in case 3 was the maximum obtained out of the 3 cases.

Image/block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	nc	c	c	nc	c	nc	c	c	c	c	c	nc	nc	nc	c	nc
2	nc	nc	nc	c	c	c	c	c	nc	nc	nc	c	c	c	c	c
3	nc	nc	nc	nc	nc	nc	c	c	nc	nc	nc	nc	nc	nc	nc	nc
4	c	nc	c	nc	c	nc	c	nc	c	nc	c	nc	nc	nc	nc	nc
5	c	nc	nc	c	nc	c	nc	c	nc	c	nc	c	nc	c	nc	nc
6	c	nc	nc	nc	c	nc	nc	c	c	nc	nc	nc	c	nc	nc	nc
7	c	c	nc	c	c	c	c	nc	c	c	c	nc	c	c	c	nc
8	c	c	nc	nc	c	nc	nc	nc	c	c	c	nc	c	nc	nc	nc
9	nc	nc	nc	c	c	c	nc	nc	c	nc	nc	c	c	nc	nc	nc
10	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	c	c	c	c
11	nc	c	nc	nc	nc	c	nc	nc	nc	c	c	c	c	c	c	c
12	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
13	nc	nc	c	nc	nc	nc	c	nc	c	c	c	nc	c	nc	nc	c
14	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	c	c	c	nc
15	nc	nc	nc	nc	c	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc
16	c	c	nc	nc	c	c	nc	nc	c	nc	nc	nc	c	nc	nc	nc
17	c	c	nc	nc	c	nc	nc	nc	c	nc	nc	nc	nc	c	nc	nc
18	c	c	c	c	c	c	c	c	c	nc	c	c	c	nc	c	c
19	nc	c	nc	c	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc
20	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc

Table 5: Table showing the blocks in the training set nc=no crack, c=crack

Image/block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
21	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc
22	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	c	c	c	c
23	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc
24	nc	c	nc	nc	nc	c	nc	nc	c	c	nc	nc	c	nc	nc	nc
25	c	c	nc	c	c	c	nc	c	c	c	nc	c	c	c	nc	c

Table 5—Continued

Table 6: Results obtained for case 1, data showing the closest matching block and image number in the training set respectively

Image/ Block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
6	14,4	12,1	12,1	6,1	13,3	5,5	9,5	4,1	5,5	12,1	16,4	1,1	4,4	5,1	1,1	11,2
7	11,2	1,3	11,2	11,2	11,2	5,2	11,2	1,3	1,3	4,3	5,2	1,1	11,2	5,2	6,2	1,1
8	11,2	7,2	6,2	11,2	11,2	3,2	7,2	1,3	5,2	12,2	11,2	6,1	12,2	12,2	4,3	1,3
9	9,5	8,1	9,2	5,1	3,5	16,1	12,1	12,1	3,5	14,4	8,1	5,1	9,5	4,1	13,1	1,3
10	4,4	9,5	1,1	4,1	5,5	3,5	16,4	9,2	9,3	10,3	4,4	4,4	14,4	14,4	4,1	1,3
11	6,2	11,2	1,3	2,1	11,2	1,1	4,1	12,1	1,3	1,3	2,1	4,4	12,1	9,2	4,4	16,1
12	4,3	6,1	16,1	1,1	1,3	8,1	14,4	4,1	1,1	16,1	1,5	4,1	1,1	16,1	12,1	10,2
13	11,2	1,3	13,1	1,3	1,3	1,3	1,1	11,2	1,1	10,2	11,2	12,2	1,5	14,3	10,3	4,1
14	5,1	3,5	10,3	10,3	5,1	13,3	14,3	14,3	1,3	3,5	14,3	14,3	11,2	5,1	5,5	14,3
15	11,2	4,3	6,2	5,2	11,2	7,2	5,2	11,2	9,1	4,3	12,2	11,2	4,1	13,1	11,2	11,2
16	1,3	5,1	4,4	9,2	9,1	4,1	9,5	12,1	6,1	16,1	5,5	5,5	6,1	4,4	5,5	14,4
17	1,1	5,2	3,2	3,2	1,3	8,2	4,2	3,2	11,2	8,2	8,2	7,2	5,2	12,2	6,2	7,2
18	14,4	13,3	12,1	7,2	9,5	5,5	12,1	8,2	5,5	10,3	12,1	8,2	9,3	14,3	9,5	12,2
19	14,3	16,4	5,2	10,2	14,3	16,4	11,2	8,1	14,3	9,3	11,2	8,1	14,3	14,3	11,2	4,1
20	4,1	12,1	9,5	12,1	8,1	1,5	14,4	14,4	8,1	1,5	5,5	5,5	8,1	9,5	5,5	5,5
21	10,1	4,4	7,5	14,4	4,1	4,4	5,5	5,5	2,1	4,4	5,5	5,5	6,1	12,1	12,1	1,5
22	9,5	16,1	2,1	11,2	9,3	9,3	9,5	4,1	14,3	14,3	9,3	1,5	1,1	1,3	11,2	11,2
23	1,3	12,2	11,2	2,1	11,2	12,2	11,2	2,1	11,2	12,2	11,2	13,1	4,3	7,2	11,2	13,1
24	8,1	16,1	14,4	12,1	16,4	16,1	1,5	5,5	1,3	2,1	14,4	5,5	1,1	4,1	9,5	16,3
25	14,4	5,5	3,5	3,5	3,5	13,3	14,3	3,5	5,5	10,3	14,3	5,5	5,5	14,3	14,3	5,5

Table 7: Results showing blocks as cracked or non-cracked for case 1, nc=no crack, c=crack

Image/Block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
6	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc
7	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	c	nc	nc	c	c	nc
8	nc	c	c	nc	nc	nc	c	nc	c	nc	nc	nc	c	c	nc	nc
9	nc	c	nc	c	nc	nc	nc	nc	nc	nc	c	c	nc	nc	nc	nc
10	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
11	c	nc	nc	c	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc
12	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	c	nc	nc	nc	nc	nc
13	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	c	c	nc	nc	nc
14	c	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc
15	nc	nc	c	c	nc	c	c	nc	c	nc	c	nc	nc	nc	nc	nc
16	nc	c	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
17	nc	c	nc	nc	nc	c	c	nc	nc	c	c	c	c	c	c	c
18	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c
19	nc	nc	c	nc	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	nc
20	nc	nc	nc	nc	c	nc	nc	nc	c	c	nc	nc	c	nc	nc	nc
21	c	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	c
22	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc
23	nc	c	nc	c	nc	c	nc	c	nc	c	nc	nc	nc	c	nc	nc
24	c	nc	nc	nc	nc	nc	c	nc	nc	c	nc	nc	nc	nc	nc	nc
25	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc

Note: c, nc = correct identification and c, nc = misidentification

Table 8: Results obtained for case 2, data showing the closest matching block and image number in the training set respectively

Image/ Block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
11	15,7	7,7	8,7	2,1	4,7	16,7	4,1	12,1	1,3	16,9	2,1	4,4	3,6	9,2	1,10	16,1
12	7,7	10, 1	16,1	15,6	16,9	8,1	6,4	4,6	15,6	16,1	6,4	4,1	12,7	16,1	10,6	6,1
13	3,7	15,6	13,1	16,10	8,7	8,7	12,7	16,6	15,6	10,2	3,7	13,8	7,6	14,3	10,3	4,1
14	5,1	3,5	10,3	10,3	5,1	13,3	14,3	14,3	8,7	3,5	14,3	14,3	16,6	5,1	5,10	14,3
15	16,6	7,7	15,7	4,3	3,7	7,2	11,7	16,6	9,1	4,7	13,8	7,7	4,1	13,1	4,7	16,6
16	15,6	5,1	1,10	9,2	9,1	4,10	9,5	12,1	6,1	16,1	5,5	5,10	4,6	13,5	5,5	14,4
17	12,7	9,8	6,8	3,2	16,9	8,2	7,2	6,8	11,2	8,2	6,8	7,2	9,8	13,8	6,2	13,8
18	16,3	9,3	12,1	7,2	1,5	5,5	3,6	8,2	5,10	10,3	3,6	8,2	13,3	14,3	7,6	13,8
19	14,3	16,4	5,2	10,2	14,3	16,4	4,7	16,4	14,3	5,6	4,7	8,1	14,3	14,3	4,7	4,6
20	4,1	12,1	9,5	10,6	8,1	1,5	14,4	14,4	8,1	1,5	6,6	6,6	8,1	10,6	6,6	6,6
21	4,6	1,10	5,10	14,4	4,6	1,10	6,6	5,5	2,1	1,10	5,5	5,5	6,1	12,1	10,6	1,5
22	9,5	16,1	2,1	16,6	9,3	3,5	9,5	4,1	14,3	14,3	10,3	7,6	1,1	16,10	11,2	4,7
23	16,10	12,2	16,6	2,1	3,7	12,2	16,6	5,1	16,6	12,2	11,2	13,1	4,3	7,2	16,6	13,1
24	8,1	16,1	14,4	12,1	16,4	16,1	1,5	6,6	15,6	2,1	14,4	6,6	1,1	4,10	9,5	16,3
25	14,4	7,5	3,5	3,5	5,10	9,3	14,3	3,5	5,10	10,3	14,3	5,5	6,6	14,3	14,3	6,6

Table 9: Results showing blocks as cracked or non-cracked for case 2, nc=no crack, c=crack

Image/block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
11	c	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
12	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
13	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
14	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
15	nc	c	c	nc	nc	c	c	nc	c	nc	nc	c	nc	nc	nc	nc
16	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
17	nc	c	nc	nc	nc	c	c	nc	nc	c	nc	c	c	nc	c	nc
18	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	nc
19	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
20	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
21	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
22	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc
23	c	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	nc	nc
24	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
25	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc

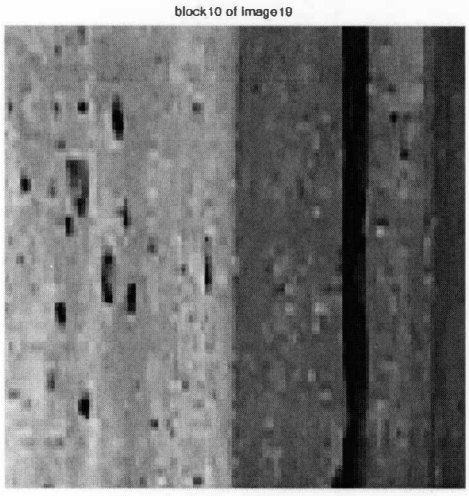
Note: c, nc = correct identification and c, nc = misidentification

Table 10: Results obtained for case 3, data showing the closest matching block and image number in the training set respectively

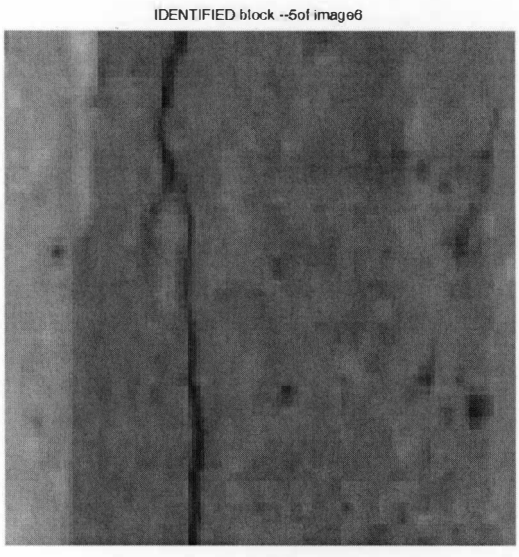
Image/ Block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16	15,6	1,14	4,4	9,2	9,1	4,10	9,5	15,12	6,1	14,12	5,5	5,10	8,12	4,4	5,5	14,4
17	4,12	4,15	6,8	3,2	16,9	8,2	4,2	6,8	15,15	8,2	6,8	6,15	9,8	11,15	6,2	6,15
18	16,3	13,3	15,12	7,2	7,6	5,5	3,6	8,2	5,10	4,14	3,6	8,2	13,3	8,14	7,6	13,8
19	16,14	16,4	7,15	10,2	12,14	16,4	5,11	16,4	12,14	5,6	5,11	13,15	12,14	11,14	5,11	4,6
20	4,1	12,1	9,5	15,12	8,1	11,12	14,4	14,4	8,1	1,5	6,6	6,6	8,1	15,12	5,5	6,6
21	4,6	1,10	2,14	14,4	8,12	1,10	6,6	5,5	2,1	1,10	5,5	5,5	6,1	12,1	15,12	1,5
22	9,5	16,1	4,11	16,6	6,14	2,14	2,4	4,1	14,3	14,3	6,14	11,12	13,12	9,14	8,15	4,7
23	5,13	12,2	5,15	4,11	3,7	1,11	16,15	5,14	8,15	12,13	5,11	13,1	2,15	7,2	16,15	13,1
24	8,1	14,12	14,4	15,12	16,4	16,1	1,5	6,6	4,12	1,14	14,4	6,6	1,1	13,15	9,5	16,3
25	14,4	5,10	2,14	2,14	2,14	2,14	4,14	2,14	15,14	10,3	11,14	5,5	5,10	11,14	12,14	6,6

Table 11: Results showing blocks as cracked or non-cracked for case 3, nc =no crack, c =crack

Image/block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
17	nc	nc	nc	nc	nc	c	c	nc	nc	c	nc	nc	c	nc	c	nc
18	nc	nc	nc	c	nc	nc	nc	c	nc	nc	nc	c	c	nc	nc	c
19	nc	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc
20	nc	nc	nc	nc	c	nc	nc	nc	c	c	nc	nc	c	nc	nc	nc
21	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	c
22	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	c	nc	nc
23	nc	c	c	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc
24	c	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc	nc	nc
25	nc	nc	nc	nc	nc	nc	nc	nc	c	nc	nc	nc	nc	nc	nc	nc

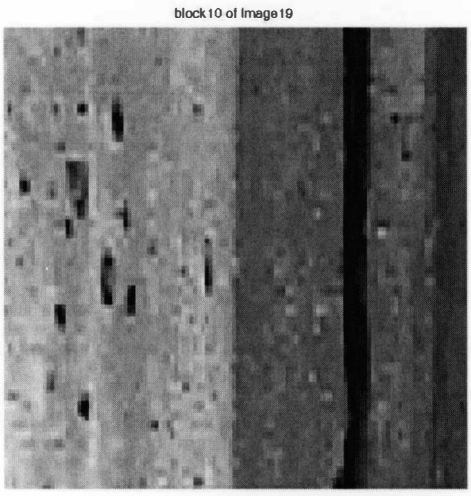


(a) Block 10 of test image 19

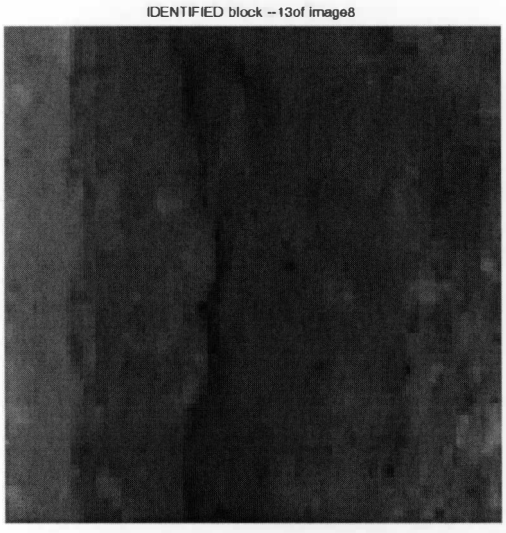


(b) Block 5 of training set image 6
(closest match)

Figure 36: Figure showing the test image blocks (a) and the corresponding closest match in the training set images (b)

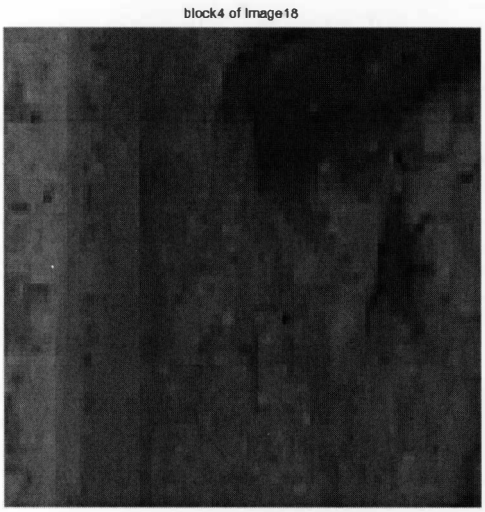


(c) Block 16 of test image 18

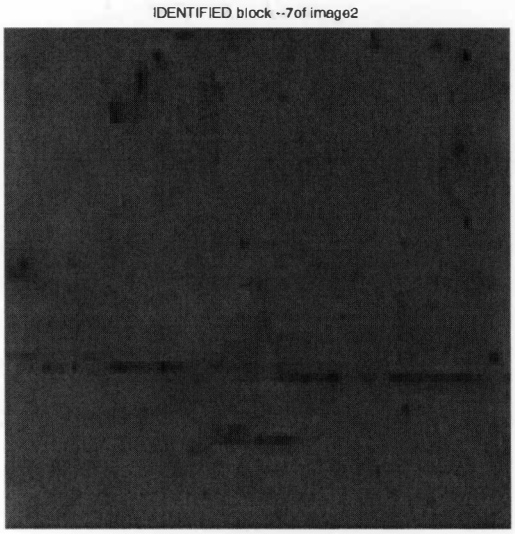


(d) Block 13 of training set image 8
(closest match)

Figure 37: Figure showing the test image blocks (c) and the corresponding closest match in the training set images (d)



(e) Block 4 of test image 18



(f) Block 7 of training set image 2
(closest match)

Figure 38: Figure showing the test image blocks (e) and the corresponding closest match in the training set images (f)

CHAPTER 5

CONCLUSIONS

5.1 Summary of Work

This thesis presents an algorithm to automate bridge maintenance by automating identification of cracks in concrete bridge decks. In this thesis initially the analysis of crack detection was performed using PCA alone. Then the analysis was extended further where linear structure modeling was used prior to the PCA model. The results required further improvement. Thus block processing was used were 3 cases were investigated. In block processing both the linear structure modeling and PCA was used. Results are reviewed in the next section and necessary recommendations are given.

5.2 Conclusions and Recommendation

To begin with the first analysis were just the PCA alone was used. The results indicated in table 1 show that when the 20 cracked bridge images were trained and the remaining 5 cracked bridge images were used as the test images just one image was identified as cracked. The accuracy was very less. Table 2 showed the results when the 20 non-cracked bridge images were trained and the 25 cracked bridge images were used as the test images. The results obtained were satisfactory where 21 images were correctly identified as cracked. The accuracy obtained was 84% from table 2. This showed the use of PCA model encouraging and more work had to be done to improve the table 1 results.

From table 3 it was shown that 3 out of 5 non-cracked bridge images were identified correctly as non-cracked when the same 20 non-cracked bridge images were trained and the remaining 5 non-cracked bridge images were used as the test images. So the accuracy obtained was 60%. The same threshold was maintained throughout the analysis.

To enhance results, linear structures were identified prior to PCA. The table 4 showed that the results improved significantly and the accuracy obtained was 88%. Here 20 non-cracked bridge images were trained and the 25 cracked bridge images were used as the test images. Figure 14 also showed that 2 out of 5 cracked bridge images were identified correctly as cracked when 20 cracked bridge images were trained. This showed improvement over the case 1.

Finally block processing was used. Here the analysis was performed using just the cracked bridge images for the training set as well as the test images. Here 3 cases were considered to test the algorithm. The results obtained for 3 cases showed that accuracy increased as the number of cracked bridge images increase in the training set. The maximum accuracy of 60% was obtained in case 3. Some of the failures occurred due to some of the cracks were larger in width. The cracks were split into two different blocks caused the algorithm to misidentify the cracks as non-cracked. This shows that the results depend highly on the quality of concrete bridge images being trained.

5.3 Future Work

As a future direction linear structure modeling can be modified to improve the accuracy much more. Use of adaptive threshold can be recommended in future work instead of using a constant threshold as used in this work for the first and second analysis. In the third analysis namely block processing since the results tend to improve in case 3. Due to the limitation in MATLAB the program crashes if more than 15 images are trained. This leads to future work were the algorithm can be implemented using c language were the training set can be increased to gain more accuracy.

APPENDIX A

MATLAB CODE FOR PCA

```
% Program1: Training
% Step 1: Creation of training set

M=input('Enter the number of images ')
for i=1:M
    dirname='c:\crack';
    fullpath=strcat(dirname,'\',num2str(i),'.bmp');
    imindex=I; %To number the images
    gim{imindex}=imread(fullpath);
    gime=gim{imindex};
    figure,imshow(gime)
    title(strcat('Image – ',num2str(i)));
    dd{imindex}=im2double(gim{imindex});
end

nRows = 50;
nCols = 50;

% Step 2: All the database images are read as matrices

for mh=1:1:M
    f1=li5{mh};
    qimage{mh}=imresize(dd,[nRows nCols]); % Resize all images to same size of 50*50
end

% Step 3: Create column stack vectors by concatenating rows.

For m = 1:M % csImage is a 1 x (nRows*nCols) row vector containing all elements of a
                % picture
    pimage=qimage{m}; %read as a matrix (that is image{imIndex}) in this case.
    For row = 1:nRows;
        csImage(1, (row-1)*nCols+1 : row*nCols) = pimage(row, :);
    end
    csImages(m, :) = csImage;
end

csImages=csImages';   % now csImages(nRows*nCols) X M
```

```
% Step 4: Calculation of covariance matrix


% Clear some memory
clear pimage image csImage; % as image is no longer required since csImages has been
                            %  found
clear V D
pack;
Psi = sum(csImages)/(nRows*nCols);
 % dimensions 1xM, where M= no of bridge images trained
 % with all rows added up and divided by the number of pixels in each image
 % to yield average of each image in the training set arranged in a row
for m = 1:nRows*nCols
   PhiMat(m, :) = csImages(m, :)  – Psi; % dimensions (nRows*nCols)XM
end

C=PhiMat'*PhiMat ;  dimensions M*M . C is the covariance matrix


% Step 5: To find the dominant eigen vectors

if M>10
   E=10;                    % No. of Eigen vectors to be taken
else
   if M<=10 & M>1
   E=M-1;
else
   E=1;
end
end
OPTIONS.disp = 0;
SIGMA = 'LM';            % Consider only the largest valued eigen vectors.
[V, D] = eigs(C, E, SIGMA, OPTIONS);  % Gets hung over here even for E=1 and not
                                      %10
clear C

for t=1:E
   eg( :,t)=PhiMat*V( :,t) ;          % eg has dimensions (nRows*nCols) * E
end
pack; % Step 6: To display the dominant eigen images
eg=eg';
for kl=1:E;
```

```
for row = 1:nRows;
    mimage(row, :)=eg(kl, (row-1)*nCols+1 : row*nCols);
    % To break up 1*(nRows*nCols) eigen vector as  nRows*nCols image
    end
egimage{kl}=mimage;
figure,imshow(mimage);
title(strcat('eigen vector – ',num2str(kl)));
end
eg=eg';

% Step 7: Calculation of projections of each cracked bridge onto the dominant eigen
% vectors
clear tm pq;
for tm=1:M
    for pq=1:E
    w(tm,pq)=eg(:,pq)'*PhiMat(:,tm);
    end
end
save kaw Psi M eg E nRows nCols w


% Program 2: Detection of cracks
load kaw;

% Step 1:  Read the test image

dirName='c:\crack\';
deltalimit=3.1; % threshold
b=input('Enter the bridge no  ')
FullPath = strcat(dirName, num2str(b),'.bmp');
c1 = imread(FullPath);
d1=size(c1);
figure,imshow(c1);
title(strcat(' test image --',num2str(b)));
[m1,n1]=size(d1);
e1=m1*n1;              % e1=3 meaning a color picture, e1=2 means black & white
    if e1==3
    f1=rgb2gray(c1);      % If e1=3 then change color to black and white
    else                  f1=c1;              % If e1=2 do not make any changes
    end


dd=im2double(f1);
pause;
```

```
kimage=imresize(dd,[nRows nCols]); % Resize all images to same size of 50*50
kimage = double(kimage);                % conversion to double is required to
% to perform mathematical operations in matlab

% Step 2 – Create column stack vector by concatenation

pimage=kimage;

for row = 1:nRows;
   csImage(1, (row-1)*nCols+1 : row*nCols) = pimage(row, :);
end

csImages = csImage';   % 2500X1

csImages = double(csImages);

Psi1=sum(csImages)/(nRows*nCols);   % average of the input image


% Step 3: The image is normalized and projected onto the eigen vectors

clear pimage kimage; % as image is no longer required since csImages has been found

for ih=1:1:nRows*nCols
   PhiMat1(ih,1)= csImages(ih,1) - Psi1;   % dimensions (nRows*nCols)*1
end
clear csImages;


%Step 4: To display the normalized bridge

PhiMat1=PhiMat1';

for row = 1:nRows;
    image1(row, :)=PhiMat1(1, (row-1)*nCols+1 : row*nCols);
end
egimage1=image1;            % egimage now contains the reconstructed bridge.
figure,imshow(egimage1);


title('NORMALISED BRIDGE')

 % Step 5: normalized bridge is projected onto the eigen vectors
```

```
PhiMat1=PhiMat1';

for pq=1:E                % no of dominant eigen vectors considered.
    w1(pq)=eg(:,pq)'*PhiMat1;
end

dist=input('Enter the distance to be used')

% different distance measures are compared
% dist=1 for city block
if dist==1
   for h=1:1:M
   w2(h)=sum(abs(w(h,:)-w1));
end
end

% dist=2 for chebyshev

if dist==2
    for h=1:1:M

 w2(h)=max(abs(w(h,:)-w1));
    end
end
% dist=3 for canberra
 if dist==3
    for h=1:1:M
      w2(h)=sum(abs(w(h,:)-w1)./abs(w(h,:)+w1));
    end
end
% dist=4 for minkowski
if dist==4
r=input('Enter the value of r metric')
    for h=1:1:M
    w2(h)=(sum(abs(w(h,:)-w1).^(1/r)))^r;
end
end
% dist=5 for euclidean
if dist==5
for h=1:1:M
    w2(h)=sqrt(sum((w(h,:)-w1).^2));
end
end
```

```
[delta,PIN]=min(w2)

 % Step 6: To display the reconstructed bridge

T=0;
for jk=1:E
    T=T+w1(1,jk)*eg(:,jk);
end

for uu=1:1:nRows*nCols
JK(uu,1)=T(uu,1)+Psi1;
end
JK=JK';
for row = 1:nRows;
    image1(row, :)=JK(1, (row-1)*nCols+1 : row*nCols);
end
egimage1=image1;            % egimage now contains the reconstructed bridge.
figure,imshow(egimage1);
title('Reconstruction')


% Step 7: To identify bridge image is cracked or non-cracked

else

if  delta<=deltalimit
    'Input is a Bridge'
    ' THE GIVEN CONCRETE BRIDGE HAS BEEN CRACKED '

 bb=strcat(dirName,num2str(PIN),'.bmp');
        figure,imshow(bb)
        title(strcat('IDENTIFIED BRIDGE --',num2str(PIN)));

end
if  delta>deltalimit
     'Recognition Error'

'THE GIVEN CONCRETE BRIDGE HAS NOT BEEN CRAKCED'
end
end
end
clear;
close all
```

APPENDIX B

MATLAB CODE FOR PCA WITH LINEAR STRUCTURE MODELING

```
% Training

% Program1: Training
% Step 1: Creation of training set

M=input('Enter the number of images ')
for i=1:M
    dirname='c:\crack';
    fullpath=strcat(dirname,'\',num2str(i),'.bmp');
    imindex=I; %To number the images
    gim{imindex}=imread(fullpath);
    gime=gim{imindex};
    figure,imshow(gime)
    title(strcat('Image – ',num2str(i)));
    dd{imindex}=im2double(gim{imindex});
end

Step 2: Linear structure modeling
for k=1:M
   imindex=k;
mas=[-1 -1 -1 -1 -1;-1 -1 -1 -1 -1;4 4 4 4 4;-1 -1 -1 -1 -1;-1 -1 -1 -1 -1];
lil1{imindex}=conv2(dd{imindex},mas);
gim22=lil1{imindex};
gim33=im2uint8(gim22);
figure,imshow(gim33)
title(strcat('line strength image (Horizontal line detector) for image -- ',num2str(k)));
end

pause;
close all
end

for k=1:M
    imindex=k;


    mas1=[-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1];
    lil2{imindex}=conv2(dd{imindex},mas1);
```

```
    gim2=lil2{imindex};
    gim2=im2uint8(gim2);
figure,imshow(gim2)
title(strcat('line strength image (vertical line detector) for image -- ',num2str(k)));
    figure,imhist(gim2)
    title(strcat('histogram of the line strength image -- ',num2str(k)));
    end
pause;
close all
end

for l=1:M
    imindex=l;
    mas2=[-1 -1 -1 -1 4;-1 -1 -1 4 -1; -1-1 4 -1 -1;-1 4 -1 -1 -1;4 -1 -1 -1 -1];
    lil3{imindex}=conv2(dd{imindex},mas2);
    gim3=lil3{imindex};
    gim3=im2uint8(gim3);
    figure,imshow(gim3)
end
pause;
close all
end

for m=1:M
    imindex=m;
    mas3=[4 -1 -1 -1 -1;-1 4 -1 -1 -1; -1 -1 4 -1 -1;-1 -1 -1 4 -1;-1 -1 -1 -1 4];
    lil4{imindex}=conv2(dd{imindex},mas3);
    gim4=lil4{imindex};
    gim4=im2uint8(gim4);
    figure,imshow(gim4)
end
pause;
close all
end

for ev=1:1:M

tot{ev}=lil1{ev} + lil2{ev}+ lil3{ev} + lil4{ev};
    tot1=tot{ev};
    tot2=im2uint8(tot1);
    figure,imshow(tot2)
end

for jk1=1:1:M
 mask1=[1 1 1 ;1 1 1 ;1 1 1 ]/9;
 li55=conv2(tot{jk1},mask1);
```

```
li5{jk1}=li55;
li55=im2uint8(li55);
figure,imshow(li55)
title('Filtered image')
end
pause;
close all
clear tot2 tot1 lil1 lil2 lil3 lil4;
nRows = 50;
nCols = 50;

% Step 3: All the database images are read as matrices

 for mh=1:1:M
    f1=li5{mh};
    qimage{mh}=imresize(f1,[nRows nCols]); % Resize all images to same size of 5
                                                    %50*50
end

% Step 4: Create column stack vectors by concatenating rows.

For m = 1:M % csImage is a 1 x (nRows*nCols) row vector containing all elements of a
              % picture
    pimage=qimage{m}; %read as a matrix (that is image{imIndex}) in this case.
    For row = 1:nRows;
        csImage(1, (row-1)*nCols+1 : row*nCols) = pimage(row, :);
    end
    csImages(m, :) = csImage;
end


csImages=csImages';   % now csImages(nRows*nCols) X M% Step 5: Calculation of
covariance matrix

% Clear some memory
clear pimage image csImage; % as image is no longer required since csImages has been
                                  %  found
clear V D
pack;
Psi = sum(csImages)/(nRows*nCols);
 % dimensions 1xM, where M= no of bridge images trained
 % with all rows added up and divided by the number of pixels in each image
% to yield average of each image in the training set arranged in a row
```

```
for m = 1:nRows*nCols
   PhiMat(m, :) = csImages(m, :)  – Psi; % dimensions (nRows*nCols)XM
end

C=PhiMat'*PhiMat ;  dimensions M*M . C is the covariance matrix

% Step 6: To find the dominant eigen vectors

if M>10
   E=10;                        % No. of Eigen vectors to be taken
else
   if M<=10 & M>1
   E=M-1;
else
   E=1;
end
end
OPTIONS.disp = 0;
SIGMA = 'LM';              % Consider only the largest valued eigen vectors.
[V, D] = eigs(C, E, SIGMA, OPTIONS);   % Gets hung over here even for E=1 and not
                                        %10
clear C

for t=1:E
  eg( :,t)=PhiMat*V( :,t) ;          % eg has dimensions (nRows*nCols) * E
end


pack;

 % Step 7: To display the dominant eigen images

eg=eg';
for kl=1:E;
   for row = 1:nRows;
   mimage(row, :)=eg(kl, (row-1)*nCols+1 : row*nCols);
   % To break up 1*(nRows*nCols) eigen vector as  nRows*nCols image
   end
egimage{kl}=mimage;
figure,imshow(mimage);
title(strcat('eigen vector – ',num2str(kl)));
end
eg=eg';
```

```
% Step 8: Calculation of projections of each cracked bridge onto the dominant eigen
% vectors
clear tm pq;
for tm=1:M
    for pq=1:E
    w(tm,pq)=eg(:,pq)'*PhiMat(:,tm);
    end
end
save kaw Psi M eg E nRows nCols w

% Program 2: Detection of cracks

load kaw;

% Step 1:  Read the test image

dirName='c:\crack\';
deltalimit=3.1; % threshold
b=input('Enter the bridge no  ')
FullPath = strcat(dirName, num2str(b),'.bmp');
c1 = imread(FullPath);
d1=size(c1);
figure,imshow(c1);
title(strcat(' test image --',num2str(b)));


[m1,n1]=size(d1);
e1=m1*n1;              % e1=3 meaning a color picture, e1=2 means black & white
    if e1==3
    f1=rgb2gray(c1);      % If e1=3 then change color to black and white
    else
    f1=c1;              % If e1=2 do not make any changes


 end
dd=im2double(f1);

Step 2: Linear structure modeling
mas=[-1 -1 -1 -1 -1;-1 -1 -1 -1 -1;4 4 4 4 4;-1 -1 -1 -1 -1;-1 -1 -1 -1 -1];
lil1=conv2(dd,mas);
gim22=lil1;
gim33=im2uint8(gim22);
figure,imshow(gim33)
```

```
pause;
close all

mas1=[-1 -1 4 -1 -1; -1-1 4 -1 -1; -1-1 4 -1 -1; -1-1 4 -1 -1; -1-1 4 -1 -1];
lil2=conv2(dd,mas1);
gim2=lil2;
gim2=im2uint8(gim2);
figure,imshow(gim2)
figure,imhist(gim2)

pause;

   mas2=[-1 -1 -1 -1 4;-1 -1 -1 4 -1; -1-1 4 -1 -1; -14 -1 -1 -1;4 -1 -1 -1 -1];
lil3=conv2(dd,mas2);
gim3=lil3;
gim3=im2uint8(gim3);
figure,imshow(gim3)
pause;
mas3=[4 -1 -1 -1 -1;-1 4 -1 -1 -1;-1 -1 4 -1 -1;-1 -1 -1 4 -1; -1-1 -1 -1 4];
lil4=conv2(dd,mas3);
gim4=lil4;
gim4=im2uint8(gim4);


figure,imshow(gim4)

pause;

    tot=lil1 + lil2+ lil3 + lil4;
    tot1=tot;
    tot2=im2uint8(tot1);
    figure,imshow(tot2)
    mask1=[1 1 1 ;1 1 1 ;1 1 1 ]/9;
    li55=conv2(tot,mask1);
    li555=im2uint8(li55);
    figure,imshow(li555)
    title('Filtered image')
    pause;
    kimage=imresize(dd,[nRows nCols]); % Resize all images to same size of 50*50
    kimage = double(kimage);                  % conversion to double is required to
                                              % to perform mathematical operations in matlab

% Step 3: Create column stack vector by concatenation
```

```
pimage=kimage;

for row = 1:nRows;
    csImage(1, (row-1)*nCols+1 : row*nCols) = pimage(row, :);
end

csImages = csImage';   % 2500X1
csImages = double(csImages);
Psi1=sum(csImages)/(nRows*nCols);   % average of the input image

% Step 4: The image is normalized and projected onto the eigen vectors

clear pimage kimage; % as image is no longer required since csImages has been found

for ih=1:1:nRows*nCols
    PhiMat1(ih,1)= csImages(ih,1) - Psi1;   % dimensions (nRows*nCols)*1
end
clear csImages;



%Step 5: To display the normalized bridge
PhiMat1=PhiMat1';
for row = 1:nRows;
    image1(row, :)=PhiMat1(1, (row-1)*nCols+1 : row*nCols);
end
egimage1=image1;            % egimage now contains the reconstructed bridge.
figure,imshow(egimage1);
title('NORMALISED BRIDGE')

% Step 6: normalized bridge is projected onto the eigen vectors

PhiMat1=PhiMat1';
for pq=1:E                  % no of dominant eigen vectors considered.
    w1(pq)=eg(:,pq)'*PhiMat1;
 end

dist=input('Enter the distance to be used')


% different distance measure are compared

% dist=1 for city block
```

```
if dist==1
    for h=1:1:M
    w2(h)=sum(abs(w(h,:)-w1));
end
end

% dist=2 for chebyshev

if dist==2
    for h=1:1:M
    w2(h)=max(abs(w(h,:)-w1));
    end
end
% dis=3 for canberra
 if dist==3
    for h=1:1:M
      w2(h)=sum(abs(w(h,:)-w1)./abs(w(h,:)+w1));
 end


end

% dist=4 for minkowski
if dist==4
    r=input('Enter the value of r metric')
    for h=1:1:M
    w2(h)=(sum(abs(w(h,:)-w1).^(1/r)))^r;
end
end
% dist=5 for euclidean
if dist==5
    for h=1:1:M
    w2(h)=sqrt(sum((w(h,:)-w1).^2));
end
end
[delta,PIN]=min(w2)
 % Step 7: To display the reconstructed bridge
T=0;
for jk=1:E
    T=T+w1(1,jk)*eg(:,jk);
 end
for uu=1:1:nRows*nCols
JK(uu,1)=T(uu,1)+Psi1;
end
```

```
JK=JK';
for row = 1:nRows;
    image1(row, :)=JK(1, (row-1)*nCols+1 : row*nCols);
end
egimage1=image1;            % egimage now contains the reconstructed bridge.
figure,imshow(egimage1);
title('Reconstruction')

% Step 8: To identify bridge image is cracked or non-cracked
else
if  delta<=deltalimit
    'Input is a Bridge'
' THE GIVEN CONCRETE BRIDGE HAS BEEN CRACKED '

        bb=strcat(dirName,num2str(PIN),'.bmp');
        figure,imshow(bb)


title(strcat('IDENTIFIED BRIDGE --',num2str(PIN)));
   end


  if  delta>deltalimit
     'Recognition Error'
'THE GIVEN CONCRETE BRIDGE HAS NOT BEEN CRAKCED'
end
end
end
clear;
close all
```

APPENDIX C

MATLAB CODE USING BLOCK PROCESSING

```
% Training after dividing training set images into blocks

% Step 1: Reading the database images

M=input('Enter the number of images ')
for i=1:M
    fullpath=strcat(num2str(i),'.bmp');
    imindex=i; %To number the images
    gim{imindex}=imread(fullpath);
    gime=gim{imindex};
    gimr=imresize(gime,[480,480]);
    figure,imshow(gime)
    title(strcat('Image -- ',num2str(i)));
    gimm{imindex}=gimr
    [x,y]=size(gimm{1})
    dd{imindex}=im2double(gimm{imindex});
end
pause;
close all;

% Step 2: Dividing the images into 6 blocks of 120X120 each
c=1;
for imindex=1:1:M

    for ii=1:120:x
        for kk=1:120:y
            g=dd{imindex};
            ddd{c}=g(ii:ii+119,kk:kk+119);
            d=ddd{c};
            da{c}=im2double(ddd{c});
            [x1,y1]=size(d);
            c=c+1;
            figure,imshow(d)
            title(strcat('Image -- ',num2str(imindex)));
        endend
end
c=c-1;
%clear da;
```

```
pause;
close all;

for k=1:c
    imindex=k;
    mas=[-1 -1 -1 -1 -1;-1 -1 -1 -1 -1;4 4 4 4 4;-1 -1 -1 -1 -1;-1 -1 -1 -1 -1];
    lil1{imindex}=conv2(da{imindex},mas);
    gim22=lil1{imindex};
    gim33=im2uint8(gim22);
    figure,imshow(gim33)
    title(strcat('line strength image (Horizontal line detector) for image -- ',num2str(k)));
end
pause;
close all

for k=1:c
    imindex=k;
    mas1=[-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1];
    lil2{imindex}=conv2(da{imindex},mas1);
    gim2=lil2{imindex};
    gim2=im2uint8(gim2);
    figure,imshow(gim2)
    title(strcat('line strength image (vertical line detector) for image -- ',num2str(k)));
    figure,imhist(gim2)
    title(strcat('histogram of the line strength image -- ',num2str(k)));
end

close all
for l=1:c
   imindex=l;
 mas2=[-1 -1 -1 -1 4;-1 -1 -1 4 -1;-1 -1 4 -1 -1;-1 4 -1 -1 -1;4 -1 -1 -1 -1];
lil3{imindex}=conv2(da{imindex},mas2);
gim3=lil3{imindex};
gim3=im2uint8(gim3);
figure,imshow(gim3) end
pause;
close all

for m=1:c
   imindex=m;
mas3=[4 -1 -1 -1 -1;-1 4 -1 -1 -1;-1 -1 4 -1 -1;-1 -1 -1 4 -1;-1 -1 -1 -1 4];


lil4{imindex}=conv2(da{imindex},mas3);
```

```
gim4=lil4{imindex};
gim4=im2uint8(gim4);
figure,imshow(gim4)
end
pause;
close all


for ev=1:1:c
   tot{ev}=lil1{ev} + lil2{ev}+ lil3{ev} + lil4{ev};
   tot1=tot{ev};
   tot2=im2uint8(tot1);
   figure,imshow(tot2)
end
clear lil1 lil2 lil3 lil4;
for jk1=1:1:c
mask1=[1 1 1;1 1 1 ;1 1 1]/9;

li5{jk1}=conv2(tot{jk1},mask1);

li55=li5{jk1}
[x2,y2]=size(li55);
li55=im2uint8(li55);
figure,imshow(li55)
%title('Filtered image')
end
pause;
close all

save info c x2 y2 li5
save blocks ddd

% PCA model

% Step 1: Create column stack vectors by concatenating rows

load info;
load blocks;

for m = 1:c
  % csImage is 1 by (nRows*nCols) row vector containing all elements of a picture
  % read as a matrix (that is image{imIndex}) in this case.
  pimage=li5{m};
for row = 1:x2;
```

```
csImage(1, (row-1)*y2+1 : row*y2) = pimage(row, :);
end
csImages(m,:) = csImage;
end
csImages=csImages';  % now csImages(nRows*nCols)XM
clear pimage image csImage; % as image is no longer required since csImages has been
%found
clear V D
pack;

Psi = sum(csImages)/(x2*y2);   % dimensions 1xM, where M= no of bridge images
                               % trained
                               % with all rows added up and divided by the number of
                               %  pixels in %each image  to yield average
                               % of each image in the training set arranged in a row

for m = 1:x2*y2
  PhiMat(m,:) = csImages(m,:) - Psi;   % dimensions (nRows*nCols)XM
end

C=PhiMat'*PhiMat;     % dimensions M*M . C is used to compute the eigen vectors of
                      % the covariance matrix, which is nothing but the eigen
                      % bridges

 Step 2: To find the dominant eigen vectors

if c>10
  E=(M*16)/2 ;     % Half of Eigen vectors considered
else
if c<=10 & c>1
    E=M-1;
    else
    E=1;


end
end
OPTIONS.disp = 0;
SIGMA = 'LM';                  % Consider only the largest valued eigen vectors.
[V, D] = eigs(C, E, SIGMA, OPTIONS); % Gets hung over here even for E=1 and not 10
clear C

for t=1:E
    eg(:,t)=PhiMat*V(:,t);          % eg has dimensions (nRows*nCols) * E
```

```
end
pack;
eg=eg';
for kl=1:E;
for row = 1:x2;
    mimage(row, :)=eg(kl, (row-1)*y2+1 : row*y2);
    % To break up 1*(nRows*nCols) eigen vector as  nRows*nCols image
end
egimage{kl}=mimage;      % egimage now contains the eigen vectors
%kiimage=imresize(mimage,[480,640])
figure,imshow(mimage);
title(strcat('eigen vector -- ',num2str(kl)));
end
eg=eg';

% Step 3: Projections are calculated
clear tm pq;
for tm=1:c
    for pq=1:E
    w(tm,pq)=eg(:,pq)'*PhiMat(:,tm);
    end
end

clear dd

save kaw Psi c eg E x1 y1 w

% detection part

load blocks
  load kaw;


% Step 1: Initializations

dirName='c:\crack\';
b=input('Enter the bridge no  ')
FullPath = strcat(dirName, num2str(b),'.bmp');
c1 = imread(FullPath);
c2=imresize(c1,[480,480])


figure,imshow(c1);
title(strcat('Test Image --',num2str(b)));
```

```
[m1,n1]=size(c2)
e1=m1*n1;              % e1=3 means its a color picture,e1=2 means black & white

   if e1==3
   f1=rgb2gray(c2);      % If e1=3 then change color to black and white
   else
   f1=c2;              % If e1=2 donot make any changes
   end

[x3,y3]=size(f1)
dd=im2double(f1);
cc=1;

   for ii=1:120:x3
   for kk=1:120:y3
   g=dd;
   ddd{cc}=g(ii:ii+119,kk:kk+119);
   d=ddd{cc};
   da{cc}=im2double(ddd{cc});

   cc=cc+1;
   figure,imshow(d)
   title(strcat('Image -- ',num2str(cc-1)));

  end
  end
  cc=cc-1;

for k=1:cc
   imindex=k;

   mas=[-1 -1 -1 -1 -1;-1 -1 -1 -1 -1;4 4 4 4 4;-1 -1 -1 -1 -1;-1 -1 -1 -1 -1];
   lil1{imindex}=conv2(da{imindex},mas);
   gim22=lil1{imindex};
   gim33=im2uint8(gim22);
   figure,imshow(gim33)
   title(strcat('line strength image (Horizontal line detector) for image -- ',num2str(k)));
end
close all


for k=1:cc
   imindex=k;
   mas1=[-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1;-1 -1 4 -1 -1];
```

```
lil2{imindex}=conv2(da{imindex},mas1);
gim2=lil2{imindex};
gim2=im2uint8(gim2);
figure,imshow(gim2)
title(strcat('line strength image (vertical line detector) for image -- ',num2str(k)));
figure,imhist(gim2)
title(strcat('histogram of the line strength image -- ',num2str(k)));
end

close all

for l=1:cc
    imindex=l;
    mas2=[-1 -1 -1 -1 4;-1 -1 -1 4 -1;-1 -1 4 -1 -1;-1 4 -1 -1 -1;4 -1 -1 -1 -1];
    lil3{imindex}=conv2(da{imindex},mas2);
    gim3=lil3{imindex};
    gim3=im2uint8(gim3);
    figure,imshow(gim3)
end
close all

for m=1:cc
    imindex=m;
    mas3=[4 -1 -1 -1 -1;-1 4 -1 -1 -1;-1 -1 4 -1 -1;-1 -1 -1 4 -1;-1 -1 -1 -1 4];


lil4{imindex}=conv2(da{imindex},mas3);
    gim4=lil4{imindex};
    gim4=im2uint8(gim4);
    figure,imshow(gim4)
end

close all
for ev=1:1:cc
    tot{ev}=lil1{ev} + lil2{ev}+ lil3{ev} + lil4{ev};
    tot1=tot{ev};
    tot2=im2uint8(tot1);
    figure,imshow(tot2)
end


for jk1=1:1:cc
mask1=[1 1 1;1 1 1 ;1 1 1]/9;
 li5{jk1}=conv2(tot{jk1},mask1);
```

```
li55=li5{jk1}
[x1,y1]=size(li55);
li55=im2uint8(li55);
figure,imshow(li55)
%title('Filtered image')
end
close all

% Step 3: Create column stack vectors by concatenating rows

clear pimage;
clear csImages csImage;
for kk=1:cc
pimage=li5{kk};

    for row = 1:x1;
    csImage(1, (row-1)*y1+1 : row*y1) = pimage(row, :);
    end
    csImages(kk,:) = csImage;
    end

    csImages = csImages';


 Psi1=sum(csImages)/(x1*y1);   % 1 X 16 average of the input image

% Step 4 -  The projections are calculated
clear pimage kimage;       % as image is no longer required since csImages has been
                           % found
clear PhiMat1;

for ih=1:1:x1*y1
PhiMat1(ih,:)= csImages(ih,:) - Psi1;   % dimensions (3600)*16
end
clear csImages;

Step 5:  To display the normalized image


PhiMat1=PhiMat1';       % dimension
for kyy=1:cc
  PhiMat2=PhiMat1(kyy,:);
for row = 1:x1;
  image1(row, :)=PhiMat2(1, (row-1)*y1+1 : row*y1);
```

```
end
egimage1= image1;            % egimage now contains the reconstructed
bridgefigure,imshow(egimage1);
title(strcat('NORMALISED BRIDGE ',num2str(kyy)))
end

% Step 5: Projections are calculated
for ihk=1:1:cc
   PhiMat11=PhiMat1(ihk,:);
PhiMat11=PhiMat11';

for pq=1:E                  % no of dominant eigen vectors considered.
    w1(pq)=eg(:,pq)'*PhiMat11;
end
w12{ihk}=w1;
end
pause;
w22=[];
   for ihk=1:cc
   w11=w12{ihk};

    for h=1:1:c
   w2(h)=sqrt(sum((w(h,:)-w11).^2));
end
w22=[w22,w2];
end
[p,q]=size(w22)
clear h;
ine=M*16;
inee=ine-1;
for h=1:ine:q
[delta,PIN]=min(w22(h:h+inee))
    PP=PIN/16;
im=ceil(PP);
im1=floor(PP);
%bb=strcat(dirName,num2str(PIN),'.bmp');
if mod(PIN,16)==0
   im1=im1-1;
end
d=ddd{PIN};
ppb=PIN-16*im1;
figure,imshow(d)
title(strcat('IDENTIFIED block --',num2str(ppb), 'of image', num2str(im )));
end
```

REFERENCES

[1] Gonzalez R.C, Woods R.E "Digital Image Processing, Prentice Hall, 2nd edition, 2002

[2] Abdel-Qader I, Ahmed K, Abudayyeh O, Miller D, "Feature Recognition using PCA and Cluster analysis," Proceedings of the IEEE Information and Technology Conference, Indianapolis, Indiana, June 2003.

[3] Zwiggelaar R, Parr T.C and Taylor C.J, "Finding orientated line patterns in digital mammographic images," Proceedings of the 7th British Machine Vision Conference, pp. 715-724, 1996

[4] Kosugi Y, Sase M, Kuwatani H, Kinoshita N, Momose T, Nishikawa J, Watanabe T, "Neural network mapping for nonlinear stereotactic normalization of brain MR images," Journal of computer assisted tomography, volume 17, pp. 455-460, 1993

[5] Kohonen T, "Self-Organizing and associative memory," Springer-Verlag, 2nd edition, 1988

[6] Turk M & Pentland, "Eigenfaces for Recognition," journal of cognitive neuroscience, VOL. 3, No 1, pp 71-86, 1991

[7] Canny J, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679-698, 1986

[8] Nalwa V. and Binford T. O, "On Detecting Edges," IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):699-714, 1986.

[9] Gonzalez R. and Wintz P, "Digital Image Processing," Addison-Wesley, Reading, Massachusetts, 1977.

[10]Boyle R.D "Scaling additional contributions to principal component analysis," Pattern Recognition, Volume 31, Pages 2047-2053, 1998

[11] Nayar S.K and Poggio T., "Early Visual Learning,"Oxford University Press, NY, 1996.

[12] Chen C.Y and Lee R.C.T "A near pattern-matching scheme based upon principal components analysis," Pattern Recognition Letters, Volume 16, Pages 339-345, 1995.

[13] Karhunen J. and Joutsensalo J. "Generalizations of principal component analysis, optimization problems, and neural networks," Neural Networks 8, Pages 549-562, 1995.

[14] Murasae H. and Nayar S.K "Learning and recognition of 3-D objects from brightness image, "Proc. AAAI Fall symp. Machine Learning in Computer Vision, Pages 25-29, 1993.

[15] Pinkowski B "Principal component analysis of speech spectrogram images," Pattern Recognition, Volume 30, Pages 777-787, 1997.

[16] Rodtook S., Rangsanseri Y. "Adaptive thresholding of document images based on laplacian sign, "International Conference on information technology 2001, Pages 501-505, 2001.

[17] Bleake A. and Isard M. "Active Contours, "Springer, NY, 1998.

[18] Turk M.A and Pentland A.P "Face recognition using eigenfaces," Proc. Conf. Computer Vision Pattern Recognition, Pages 586-591, 1991.

[19] Joliffe I.T, "Principal Component Analysis," Springer-Verlag, 2nd edition, 2002.

[20] Abdel-Qader I, Abudayyeh O, Kelly M, "Analysis of Edge Detection Techniques for Crack Identification in Bridges," Journal of Computing in Civil engineering, accepted, 2003.

[21] Brinckerhoff P, Silano L.G, "Bridge inspection and rehabilitation A practical guide,"Wiley-Interscience, 1st edition, 1992

[22] Washer G.A, "Improving bridge inspections," Public roads, Volume 67, No 2, 2003

[23] Abdel-Qader I, Ahmed K, Abudayyeh O, Linear Structure Modeling and PCA Algorithm for Bridge Crack Detection," Proceedings of the IEEE Information and Technology Conference, Milwaukee, Wisconsin, Aug 2004