AUTOMATED INCIDENT DETECTION IN SURVEILLANCE SYSTEMS

Murtaza Motiwala Student ID: 05310010
Shupriyo Shafkat Ahmed Student ID: 05210036
Sabrina Ahmed Student ID: 05310005

Department of Computer Science and Engineering

April 2009

BRAC University, Dhaka, Bangladesh

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of
Supervisor
(Tarem Ahmed, Senior Lecturer)

Signature of
Authors
(Murtaza Motiwala)
(Shupriyo Shafkat Ahmed)
(Sabrina Ahmed)

ACKNOWLEDGEMENTS

Special thanks to our supervisor, Mr. Tarem Ahmed for providing us with valuable insight on the subject of anomaly detection and for supporting our work throughout the course of our project, to Mrs. Afroza Sultana for trusting us with her webcam and to Mohammad Rezaul Islam, System Administrator, BRAC University for providing us with security footage on such short notice.

ABSTRACT

The importance of public security cannot be overemphasized in today's world. Surveillance systems consisting of extensive camera networks now have many applications. The London Underground has 9000 cameras, with each staff required to monitor as many as 60 at a time. This presents a significant scope for automation. Processing a sequence of images from a network presents challenges in terms of deciding upon the feature set to be extracted, and the appropriate choice of algorithm. We investigate methods based upon principal component analysis and nearest-neighbor distances to propose a system of automated intruder detection in surveillance systems and suggest new applications in Bangladesh of such systems.

Table of Content

Page

TITLE…… i

DECLARATION…….. ii

ACKNOWLEDGEMENTS…… iii

ABSTRACT…… iv

TABLE OF CONTENT…… v

(1.0) INTRODUCTION…… 1

(2.0) PROJECT METHODOLOGY…… 2

(3.0) THE NEED FOR IMAGE COMPRESSION…… 3

(3.1) THE DISCRETE WAVELET TRANSFORM…….. 5

(3.1.1) THE HAAR WAVELET …….. 7

(3.1.2) PROJET APPLICATION OF THE HAAR TRANSFORM… 11

(3.2) THE FAST FOURIER TRANSFORM…….. 12

(3.2.1) PROJECT APPLICATION OF FFT…….. 12

(4.0) RESIZING USING BILINEAR INTERPOLATION ……. 13

(4.1) PROJECT APPLICATION OF INTERPOLATION…… 14

(5.0) THE PRINCIPAL COMPONENT ANANLYSIS……. 15

(5.1) PROJECT APPLICATION OF PCA……... 15

(5.1.1) THE ONLINE PCA…… 24

(6.0) ONE CLASS NEIGHBOR MACHINE: Kth-NEAREST NEIGHBOR…….. 25
(6.1) PROJECT APPLICATION OF OCNM…… 26

(7.0) ALGORITHMS…….. 30
(7.1) PCA…….. 30
(7.1.1) ONLINE PCA…… 32
(7.2) Kth NEAREST NEIGHBOR …… 35

(8.0) EXPERIMENTAL RESULTS…….. 37

(9.0) CONCLUSION…… 39

(10.0) FUTURE WORK……... 39

LIST OF REFERENCES……. 40

LIST OF TABLES

		Page
(1.1)	Step by step Transformation	9
(2.1)	Mean Subtraction of Data	16
(2.2)	Transformed data (2 eigenvectors)	22
(2.3)	Transformed data (1 eigenvector)	23
(3.1)	Distances of all Neighbors	27
(3.2)	Sorted Nearest Neighbors	28
(4.1)	Algorithm Runtime Comparison	38

LIST OF FIGURES

		Page
Fig(1.1)	(1)Haar (2)Daubechies4 (3)Coiflet1 (4)Symlet2 (5)Meyer (6)Morlet (7)Mexican Hat.	6
Fig(1.2)	Rosa Parks(1955)	8
Fig(1.3)	Magnification of the nose.	8
Fig(1.4)	Compression with varying thresholds.	11
Fig(2.1)	Resizing using interpolation	13
Fig(3.1)	Plot of the mean subtracted data.	16
Fig(3.2)	Plot of the normalized data with the covariance eigenvectors laid on top	20
Fig(3.3)	The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.	22
Fig(3.4)	Online PCA, sliding window	24
Fig(4.1)	Detecting an outlier using nearest neighbor	2
Fig(5.1)	Normal Data	37
Fig(5.2)	Anomalous Data	38

INTRODUCTION

Our motivation for detecting the presence of human beings in images originates from its necessity in **Automated Incident Detection in Surveillance Systems**. These systems are futuristic communication systems that connect an environment to another environment and provide automation for many functionalities. Hence, if a intruder is present in an environment, the status of the him is automatically updated, based upon which several scheduled tasks can be performed. The presence of person can be sensed through optical sensors like cameras. There may be several cameras placed around in the environment in order to sense the presence of a person.

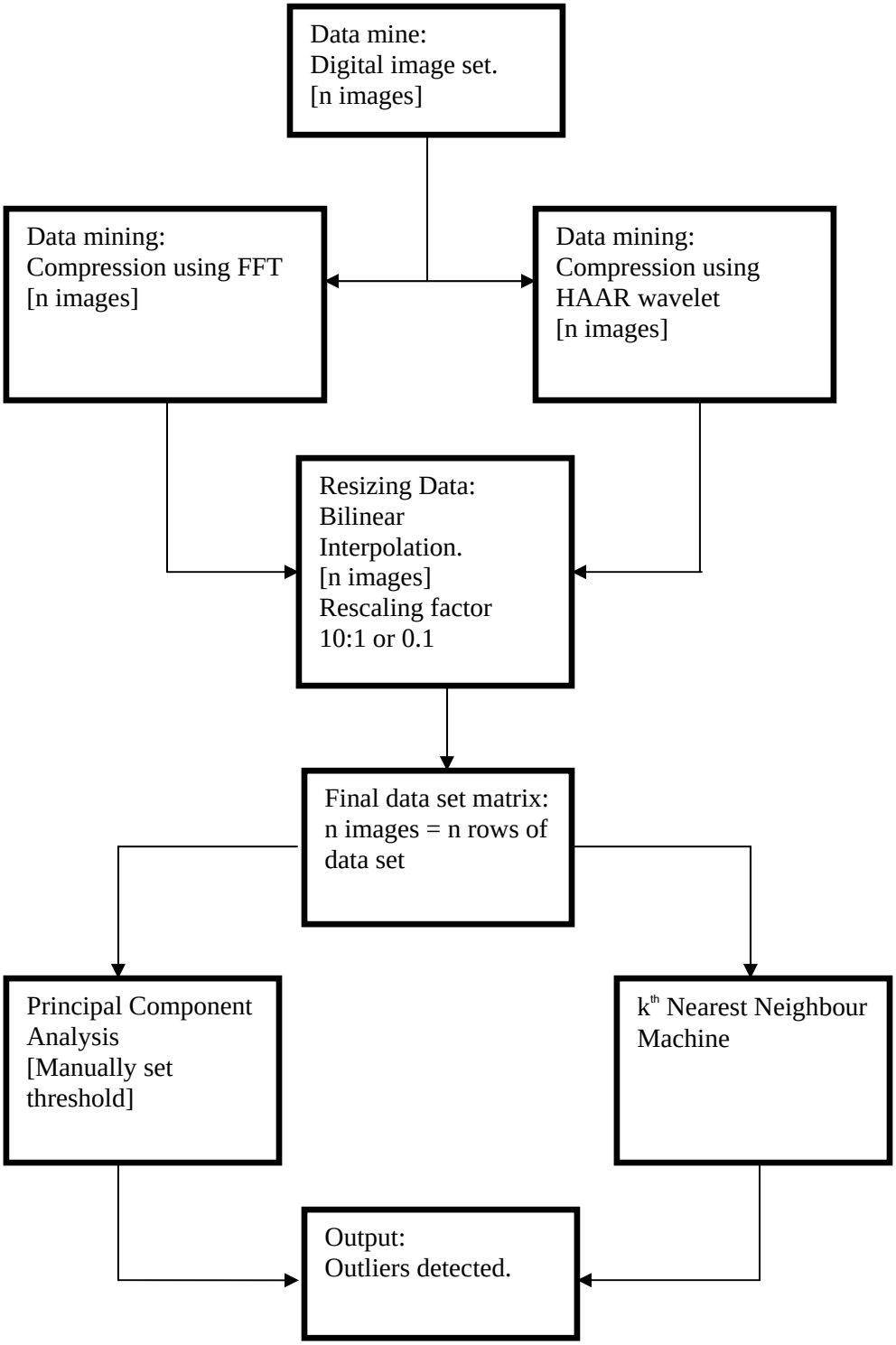
Signals can be represented in two domains:

- Time domain
- Frequency domain

In time domain we get continuous values but in frequency domain we get discrete values. The images that we used can be extracted in either space domain or in the frequency domain by using Discrete Wavelet Transform .

To represent the images our primary target is to transform them to the frequency domain using the Discrete Wavelet transform or the Fast Fourier transform. The space domain could also be used instead of frequency domain, but in frequency domain the difference between the normal image and anomalous image is more pronounced. However, an advantage of using the wavelet is that an image can be observed in both the frequency and the space domain simultaneously. The frequency domain has discrete values. When the image is normal, i.e when there is no intruder, then the wavelet transform result in low discrete values. But whenever an anomaly is present, we observe high frequency value in the result i.e. there is a sudden change in frequency. In this way we can differentiate between a normal image and an anomalous image. Moreover, the wavelet transform results in a significant reduction of redundant data. Since we had images with high dimensions, it was difficult to read all the images in MATLAB. Large dimensions caused run time error in MATLAB. To overcome that problem wavelet transform was used. Subsequent to this transformation, we pass our data through special algorithms that are commonly known as '*classifiers*', usually used as a tool for simple machine learning, to filter out the abnormal information.

(2.0) PROJECT METHODOLOGY



The subsequent sections will explain in detail the theory and the reasons behind choosing each method and process mentioned in the flow chart above.

(3.0) THE NEED FOR IMAGE COMPRESSION

The quality of a digital image is one of the most crucial characteristic taken into account when transforming an image from the analog world to the digital world. Analog images consists of a continuous stream of frequencies over a given area with no discrete pixel boundaries creating the need for vast digital occupancy if they could be transformed to match reality, however this is impossible. The array of photo sensors in a digital camera, in part, prevent a continuous image form taking shape and its capabilities are limited to the spacing between the sensors. However the charges stored in these sensors are still analog and await conversion to digital. Raw digital images are still very large in size and rely on filtering and compression to be practically viable.

The issue of image compression can be explained by three factors:
Spatial redundancy
Spectral redundancy
Temporal redundancy

Uncompressed and unfiltered digital images can occupy spaces as less as 1 MB or more making them unreasonably large for data communication. However, digital technology provides ways for processing large amounts of data following very flexible and effective instructions or ‘algorithms’ so as to reduce data, or in this case, ‘image data reduction’ /‘image compression’.

The term ‘data reduction’ may imply to most that the image quality is also reduced. This is not always true. Loss of data is completely avoided in ‘lossless

compression' through a technique called 'predictive encoding'. Image data is reduced but relevant information is still preserved so that the image can be reproduced without loss at the receiving end.

For this let us assume an example of a gray level image with a sequence of gray levels as follows:
115 116 125 130 142 150 150 130 ……..

An example of predictive encoding is a method where by only the first gray level and only the differences in the following gray levels is preserved.

+115 +1 +9 +5 +12 +8 0 -20…….

This type of encoding reduces the size of the image by 80%. Reducing the digital capacity required by image data or the amount of bits required to be transmitted.

Another method called 'statistical encoding' , used by Samuel Morse over 150 years ago, is also a lossless form of encoding. Statistical encoding and predictive encoding together combine to give a redundancy reduction, which is a reduction of the reiteration of the same bit patterns in the data.

These methods of compression are also reversible, which means that the data can be reconstructed to form the original image without error, provided the algorithms are used at both the compressing and decompressing end.

Now, coming down to the case of 'lossy compression'; it is not always the case that amends made to an image are reversible. An image is reduced to a tolerable level that is objectively accepted. 'Lossy' image processing is a so called 'irrelevancy reduction' where by information is grouped and only what is necessary is specifically chosen while the rest is discarded. For example, in 'transform encoding' which is similar

to the Fourier transform or any other mathematical transform, an image is sorted into areas of gradual spatial variation of brightness and areas of fast variation of brightness. The information on slower changes is communicated ‘losslessly’ as described earlier while the rest is transmitted with very little or no accuracy. This ensures that only areas with significant information is preserved while areas with sudden or irrelevant changes are discarded. Since the information cannot be decompressed to give us the original image, this form of compression is known as ‘lossy’.

The immense requirement for digital capacity and efficient data rates are reduced by ‘lossless’ and ‘lossy’ compression. However, this raises the question of the qualitative factor, this matter is resolved objectively as to what level of qualitative assessment is required. In the next section, we describe how wavelet transforms are used for data compression paying special attention to the Haar wavelet.

(3.1) THE DISCRETE WAVELET TRANSFORM

The Fourier transform can be used to represent signals by a finite or infinite sum of sines and cosines. The drawback of the Fourier expansion is that it is not localized, it has only frequency resolution. Although it might be possible to calculate what frequencies are present it is not possible to tell during what instant of time. Furthermore, they’re calculations are complex and require greater processing power. Solutions were developed to overcome the shortcomings of the Fourier transform which would allow us to view a signal in both the time and frequency domain simultaneously.

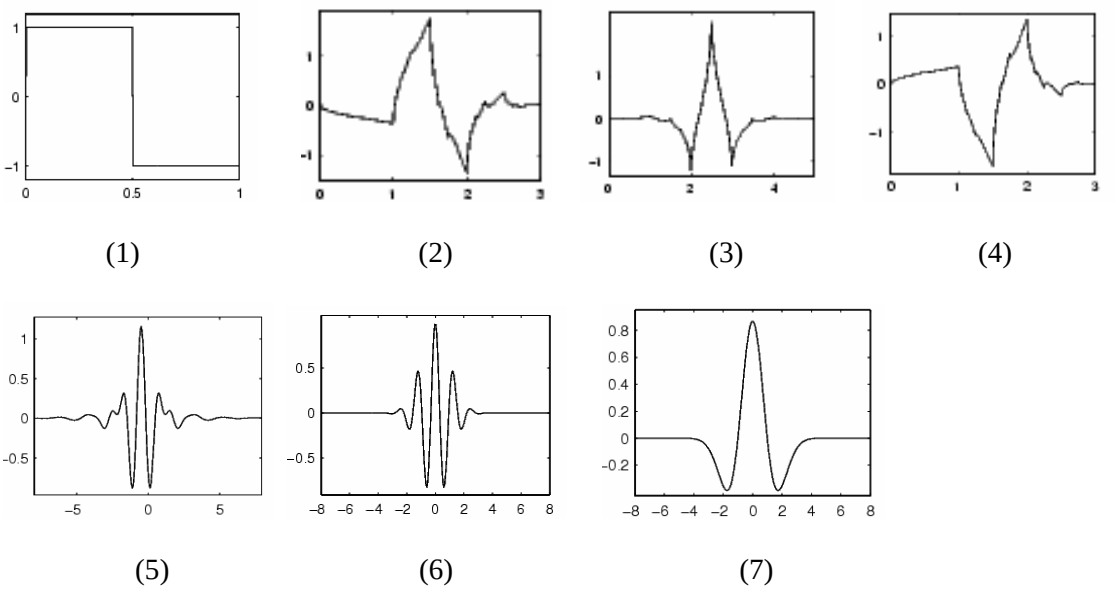
The wavelet transform uses a fully scalable modulated window to cut the signal. The window is shifted along the signal and at every point the spectrum is calculated. This process is repeated a number of times, each time using a smaller window. The end result is a time-scale representation of the signal with multi-resolution which is highly localized, meaning that it has both time and scale resolution.

The wavelets are derived from a single basic wavelet called the mother wavelet, $\Psi(t)$:

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi\left(\frac{t-\tau}{s}\right) \quad (1.1)$$

Where s is the scale factor, τ is the translation factor and $s^{-1/2}$ is the normalization factor used for energy across the different scales.

The following images show the different mother wavelets of the different wavelet families:



(1) (2) (3) (4)

(5) (6) (7)

Fig(1.1) : (1) Haar (2) Daubechies4 (3) Coiflet1 (4) Symlet2 (5) Meyer (6) Morlet (7) Mexican Hat.

Equation (1) is defined as the mother wavelet for the continuous wavelet transform (CWT), its properties make it impossible to be used for practical digital applications because it is highly redundant. To overcome this problem discrete wavelet

transform is used. Discrete wavelets are scaled and translated in small steps, shown in the following representation:

$$\Psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \Psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j} \right) \quad (1.2)$$

Where *j* and *k* are integers and τ_0 is the translation factor which depends on s_0, the dilation step. For computers, $s_0=2$ and $\tau_0=1$ is chosen so that we get a *dyadic* sampling of the time-axis.

(3.1.1) THE HAAR WAVELET

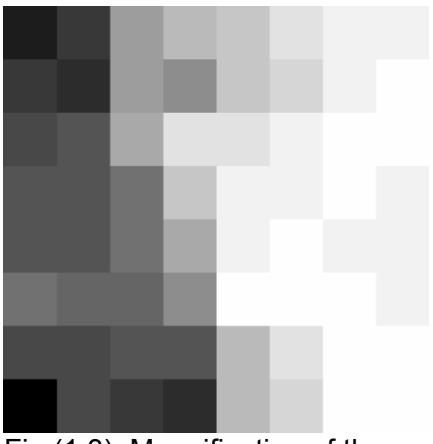
This sub-chapter describes one of the simplest wavelets, one we have preferentially used as a compression technique in our project. The Haar wavelet is a form of predictive encoding, which yields approximations of an image. The relevance of these approximations is that it needs very little space for storage and/or very little computational power to perform any form of editing on it, since there are fewer bits involved.

Throughout this section we will be using an image of Rosa Parks, Fig(1.2), to show what a data set undergoing Haar wavelet transform looks like at different stages.

Fig(1.2): Rosa Parks(1955)

This digital image is represented by a 128●128 matrix with 128^2 = 16,384 elements. The image contains 2^5 = 32 shades of gray and is said to be a 5-bit image. Upon viewing the large array that represents this image it immediately poses the problem of how this data is to be handled.

The Haar wavelet outlines a technique, that functions to transform this data into one that is more easily and efficiently communicated. To make things simpler we consider a magnification of the image around the nose region. This sub-matrix is represented by rows 60 to 67 and columns 105 to 112 of the original matrix.



$$P = \begin{pmatrix} 576 & 704 & 1152 & 1280 & 1344 & 1472 & 1536 & 1536 \\ 704 & 640 & 1156 & 1088 & 1344 & 1408 & 1536 & 1600 \\ 768 & 832 & 1216 & 1472 & 1472 & 1536 & 1600 & 1600 \\ 832 & 832 & 960 & 1344 & 1536 & 1536 & 1600 & 1536 \\ 832 & 832 & 960 & 1216 & 1536 & 1600 & 1536 & 1536 \\ 960 & 896 & 896 & 1088 & 1600 & 1600 & 1600 & 1536 \\ 768 & 768 & 832 & 832 & 1280 & 1472 & 1600 & 1600 \\ 448 & 768 & 704 & 640 & 1280 & 1408 & 1600 & 1600 \end{pmatrix}$$

Fig (1.3): Magnification of the nose.

The first step to transforming this data is to treat each row as a separate component or string. Then *Averaging and Differencing* is performed on each row after which it is performed column-wise on the resulting matrix.

To demonstrate what *Averaging and Differencing* does to a matrix, we consider the first row of the matrix *P*.

Table (1.1): Step by step transformation

576	704	1152	1280	1344	1472	1536	1536
640	1216	1408	1536	**-64**	**-64**	**-64**	**0**
928	1472	**-288**	**-64**	**-64**	**-64**	**-64**	**0**
1200	**-272**	**-288**	**-64**	**-64**	**-64**	**-64**	**0**

The first row is our original data. The first 4 numbers in the second row are the averages of each pair in the first row. The first 2 numbers in the third row are the averages of the 2 pairs in the second row and again similarly the first number in the fourth row is the average of the single pair in the third row. The remaining numbers shown in bold are the deviations from the averages. The last four elements in the second row results from subtracting each average in the second row from the first element of their respective pairs in the first row. These numbers are called detail coefficients. The process is repeated in each subsequent row. The last row is the end result of the transformation and is replaced in the matrix in place of the original row. Once this is completed for all rows the same process is continued with the columns of the matrix. The transformation is then complete and reversible. The final result is a 8●8 matrix that we shall call T.

$$T = \begin{pmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 4 \\ 30 & 36 & -90 & -2 & 8 & -20 & 8 & -4 \\ -50 & -10 & -20 & -24 & 0 & 72 & -16 & -16 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 8 \\ 8 & 8 & -32 & 16 & -48 & -48 & -16 & 16 \\ 20 & 20 & -56 & -16 & -16 & 32 & -16 & -16 \\ -8 & 8 & -48 & 0 & -16 & -16 & -16 & -16 \\ 44 & 36 & 0 & 8 & 80 & -16 & -16 & 0 \end{pmatrix}$$

In the transformed matrix, the regions of little or no variation show up as zero/small elements in the new matrix. The 0 elements in T show up due to the presence of identical adjacent elements in P.

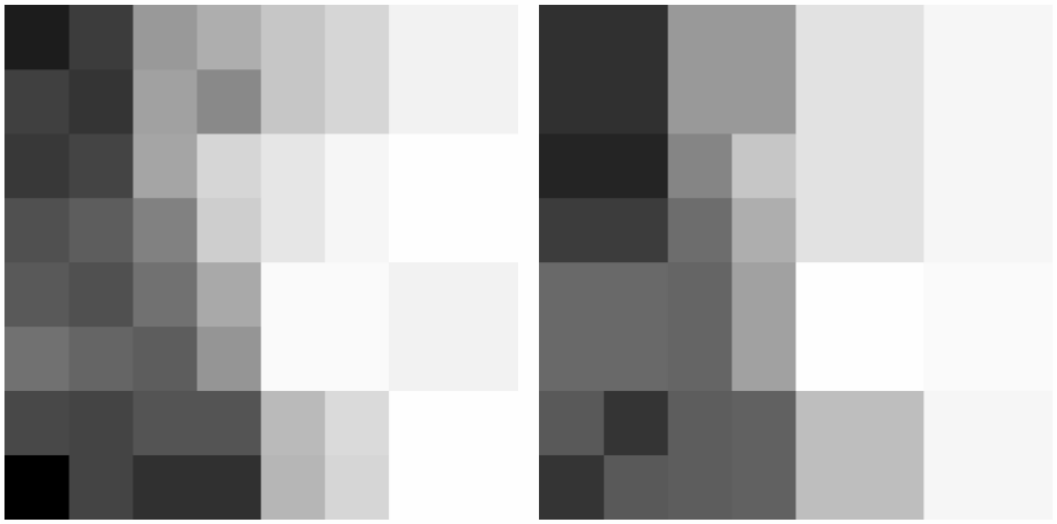
By applying the inverse of this transform, we regain our original data exactly the way it was. As mentioned in previous sections this is a 'lossless' compression. If further compression is required we will have to resort to a loss of image quality. Depending on how much detail loss is acceptable, we pick a non-negative threshold value *e;* any value below this in the transformed matrix T will be set to zero. Assuming *e = 20,* the data set D is obtained.

$$D = \begin{pmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 0 \\ 30 & 36 & -90 & 0 & 0 & 0 & 0 & 0 \\ -50 & 0 & 0 & -24 & 0 & 72 & 0 & 0 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 0 \\ 0 & 0 & -32 & 0 & -48 & -48 & 0 & 0 \\ 0 & 0 & -56 & 0 & 0 & 32 & 0 & 0 \\ 0 & 0 & -48 & 0 & 0 & 0 & 0 & 0 \\ 44 & 36 & 0 & 0 & 80 & 0 & 0 & 0 \end{pmatrix}$$

Upon approximate reconstruction of this data, we get:

$$R = \begin{pmatrix} 582 & 726 & 1146 & 1234 & 1344 & 1424 & 1540 & 1540 \\ 742 & 694 & 1178 & 1074 & 1344 & 1424 & 1540 & 1540 \\ 706 & 754 & 1206 & 1422 & 1492 & 1572 & 1592 & 1592 \\ 818 & 866 & 1030 & 1374 & 1492 & 1572 & 1592 & 1592 \\ 856 & 808 & 956 & 1220 & 1574 & 1590 & 1554 & 1554 \\ 952 & 904 & 860 & 1124 & 1574 & 1590 & 1554 & 1554 \\ 776 & 760 & 826 & 836 & 1294 & 1438 & 1610 & 1610 \\ 456 & 760 & 668 & 676 & 1278 & 1422 & 1594 & 1594 \end{pmatrix}.$$

There is a deviation from the original image, but important attributes of the image are preserved.



e=20 e=50

Fig (1.4) : Compression with varying thresholds.

When e=50, it is obvious that the resulting image deviates even further. If we measure this compression the matrix D has 29 nonzero elements while the matrix T has 60 nonzero elements out of 64, resulting in a 50% or 2:1 compression of the image.

(3.1.2) PROJET APPLICATION OF THE HAAR TRANSFORM

In our project, we have not considered the reconstruction of the transformed images. Our objective was to minimize the amount of data contained within our data set and to obtain a viewable representation of our data set in the frequency domain. Since all images in our set underwent the same process of transformation they remain relatively the same. Despite the fact that all the images don't look like what they use to all the differences and similarities between them have been preserved. This would enable us to process the data using the Principle Component Analysis requiring lower processing power without any deviation in the results in contrast to using the original images themselves.

(3.2) THE FAST FOURIER TRANSFORM

The Fourier Transform's usefulness lies in it's ability to transform signals in the time domain to the frequency domain. Upon translating the signal function, the signal can be dissected to analyze the different sine and cosine signals that make it up. The Fast Fourier Transform or FFT is just a special class of algorithms developed to implement the Discrete Fourier Transform or DFT with lower computational time. The FFT is an approximation of the DFT, it computes the DFT of a given function with considerable reduction in calculations.

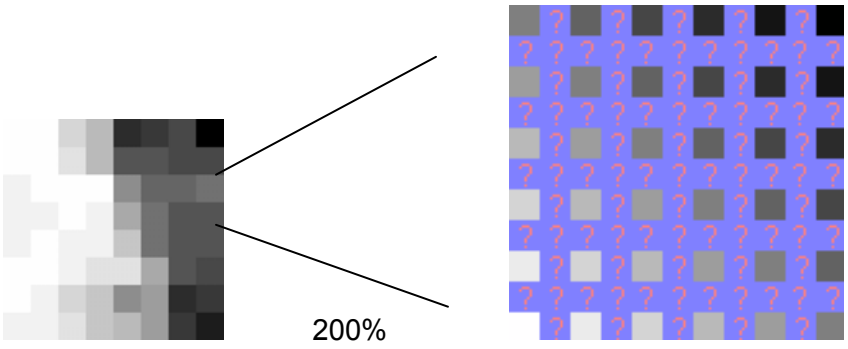
(3.2.1) PROJECT APPLICATION OF FFT

In hopes of overcoming spectral redundancy, we observed that the FFT requires greater computational time and the resulting matrix elements are sometimes large and complex. This prohibited us from obtaining a viewable image after transformation, limiting our abilities of recognizing trends in the transformed images. Although we have not made the FFT our primary choice for compression we have considered it to be a possibility and furthermore to help us index the superiority of the DWT over the FFT.

(4.0) RESIZING USING BILINEAR INTERPOLATION

Every image undergoes interpolation at some point or the other. It is used every time an image is resized or rotated. Interpolation is used to estimate data at unknown points basing its calculations on given known data points. This is similar to a method known as extrapolation only that extrapolation estimates data at points outside the known data points and interpolation provides estimations between or within known data points. For example, if we knew the temperature at 7am and 11am, interpolation would be aimed at determining the temperature at lets say 9am while extrapolation would be aimed at estimating the temperature at say 1pm. Extrapolation is therefore subject to more uncertainty. The results of interpolation however are more accurate since it aims to predict data within an approximate trend.

Every time an image is resized it creates or destroys pixels. Interpolation determines what values are to be placed in these new pixels and its results vary based on what algorithm is used. Since interpolation is only an estimate, image quality is lost every time an image is resized. Image interpolation works in 2 dimensions expanding or reducing the image leaving blank pixels whose parameters are to be determined by interpolation methods, as shown in the figure below.



Fig(2.1): Resizing using interpolation

Bilinear interpolation is one method among several others that considers the nearest 2●2 neighborhood of known pixel values surrounding the unknown pixel. It then

takes an average of the 4 pixels to calculate the final pixel value. This method results in a much smoother image than other techniques.

(4.1) PROJECT APPLICATION OF INTERPOLATION

For our project, we required the matrix of a single image that we obtain after compression to be laid out and form a single row of a new matrix. In this way, *n* images would result in a matrix of *n* rows that we would later feed into our anomaly detection algorithm. However, the large size of these images (> 100●100) would result in a massive matrix. To overcome this problem, we have resorted to using bilinear interpolation to reduce the size of our compressed images by 10 fold (rescaling factor = 0.1). The resulting smaller images were then laid out into the new matrix which was 10 factors smaller. The outliers were detected correctly showing that interpolation had no effect whatsoever on the final output except for lower computational time during the anomaly detection phase.

(5.0) THE PRINCIPAL COMPONENT ANANLYSIS

The first algorithm we have used for incident detection is the Principal Component Analysis, shortly refered to as PCA. It is a popular pattern classifying technique which is applied in many fields ranging from economics to the study of gene pools. PCA is a procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components.

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. PCA can be used for dimensionality reduction in a data set by retaining those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. The goal is to transform a given data set to an alternative data set of smaller dimension.

(5.1) PROJECT APPLICATION OF PCA

Below is a step by step explanation of how the Principal Component Analysis is carried out on a sample data set

i) Organize the data set:

As we mentioned earlier the matrix was reduced in size, since it was difficult for MATLAB to simulate large matrices. Hence after compression we got a matrix X having *t* rows and f coloumns.

Subtract the mean:

Mean subtraction is an integral part of the solution towards finding a principal component. The mean subtracted is the average across each dimension. This produces a data set whose mean is zero.

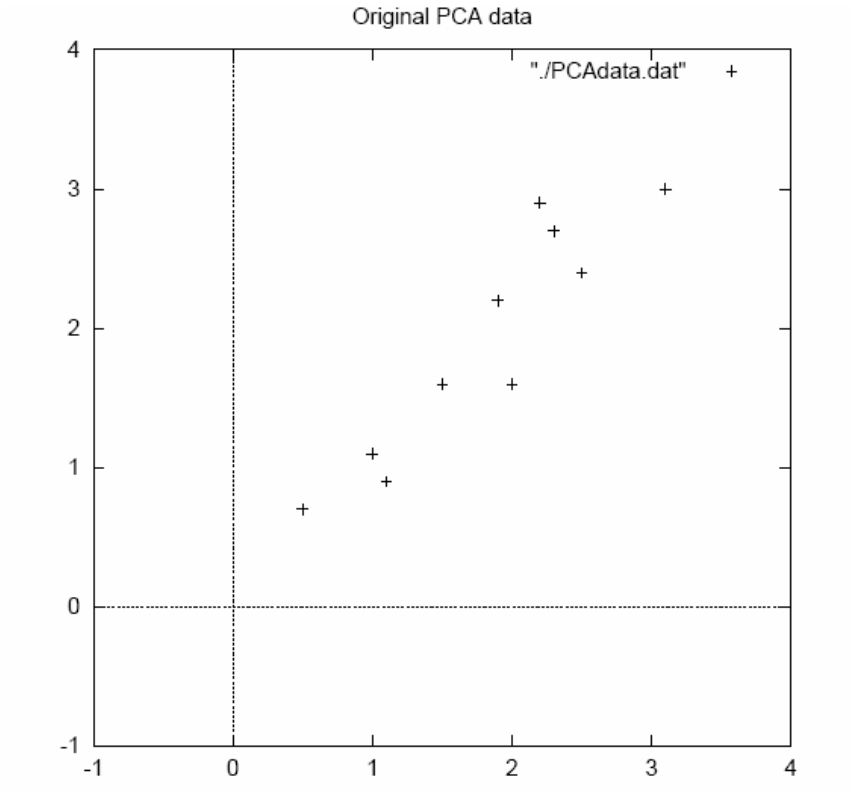
For example if we consider a 2 dimensional data set given below:

Table(2.1) : Mean Subtraction of data

Actual Data	
x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Modified Data	
x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

The modified data are the mean subtractions from the data set on the left.



Fig(3.1): Plot of the mean subtracted data.

In our PCA programme we used the MATLAB function ‘repmat’ to subtract off each dimension across all timesteps.

Before we continue further a review of certain basic statistical parameters are in order.

Variance: Variance is a measure of the spread of data in a data set. In fact it is almost identical to the standard deviation. The formula for variance is :

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)} \qquad (2.1)$$

Standard deviation and variance only operate on 1 dimension, so that we could only calculate the standard deviation for each dimension of the data set independently of the other dimensions. However, it is useful to have a similar measure to find out how much the dimensions vary from the mean with respect to each other. Covariance is such a measure.

Covariance: Covariance is always measured between 2 dimensions. If we calculate the covariance between one dimension and itself, we will get the variance. If we had a 3-dimensional data set (x,y,x)then we could measure the covarience between the x and y dimensions, y and z dimensions and z and x dimensions.
The formula for covariance is :

$$\mathrm{Cov(X,Y)} = \frac{\sum_{i=1}^{n}(Xi - \overline{X})(Yi - \overline{Y})}{(n-1)} \qquad (2.2)$$

ii) Calculate the covariance matrix:

For multi dimension we can find the covariance matrix by taking covariance of any two dimensions each time. If you have an n dimensional data set, then the matrix has n rows and n columns and each entry in the matrix is the result of calculating the covariance

between two separate dimensions. For a 3 dimensional data set we can get the covariance matrix by the following way:

$$C = \begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{pmatrix}$$

Down the main diagonal, we can see that the covariance value is between one of the dimensions and itself. These are the variances for that dimension.

For our example the 2 dimensional data set will have the following covariance matrix:

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

Since the images are multidimensional, a multidimensional covariance matrix is formed. In our program the covariance matrix we calculated is from the mean subtracted value that is from the modified data.

C=X′●X

Here C is the covariance matrix.

iii) Calculate the eigenvectors and eigenvalues of the covariance matrix:

After getting the covariance matrix we need to find the eigenvectors and the eigenvalues. In mathematics, given a linear transformation, an eigenvector of that linear transformation is a nonzero vector which, when that transformation is applied to it, may change in length, but not direction.

For each eigenvector of a linear transformation, there is a corresponding scalar value called an eigenvalue for that vector, which determines the amount the eigenvector is scaled under the linear transformation. For example, an eigenvalue of +2 means that the eigenvector is doubled in length and points in the same direction. An eigenvalue of

+1 means that the eigenvector is unchanged, while an eigenvalue of −1 means that the eigenvector is reversed in sense. Eigenvectors can only be found for square matrices. And, not every square matrix has eigenvectors. For an *n●n* matrix that does have eigenvectors, there are n of them.Given a *3●3* matrix, there are 3 eigenvectors.

For the example stated above, the eigen values and eigen vectors are:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

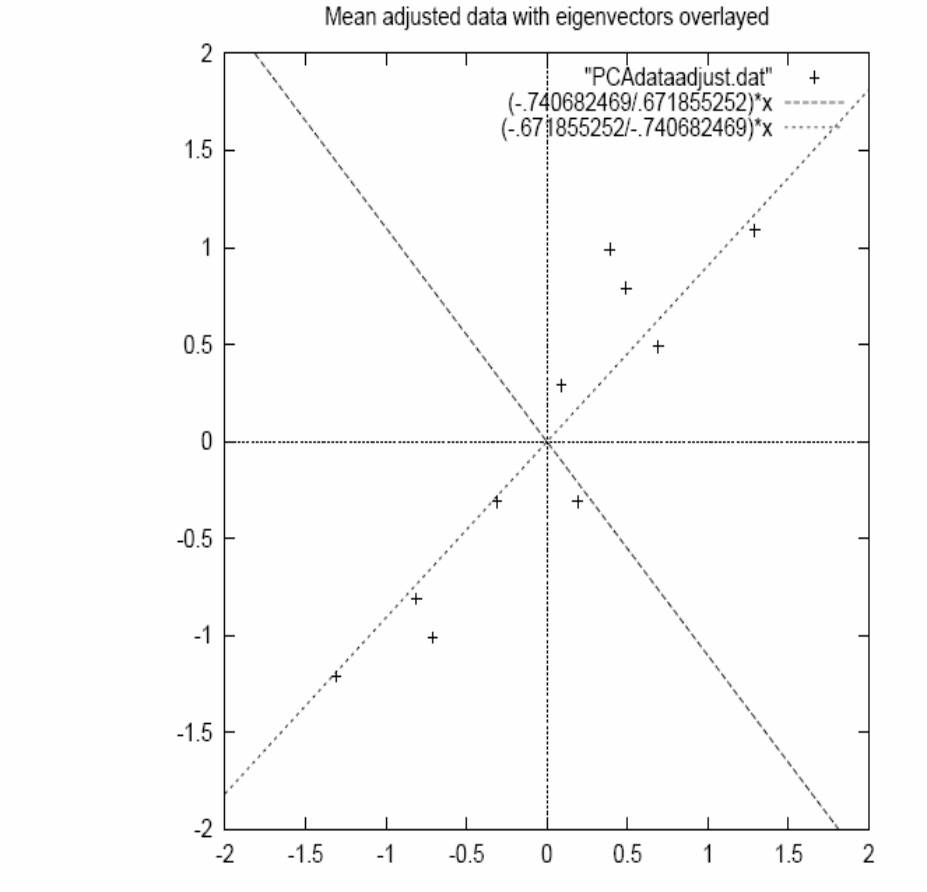
In our program we found the eigenvectors and eigenvalues from the matrix by using ‘eig’ function of MATLAB.

[V D]=eig(c)

Here V is the eigenvectors and D is the eigenvalues.Then we re-arranged the eigenvalus in descending order. This gives you the components in order of significance. Now, if you like, you can decide to ignore the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don’t lose much. If you leave out some components, the final data set will have lesser dimensions than the original. To.The eigenvector with the highest eigenvalue is the principle component of the data set.

iv) Choosing components and forming a feature vector:

The eigenvector with the highest eigenvalue is the principle component of the data set. What needs to be done now is you need to form a feature vector, which is a matrix of vectors. This is constructed by taking the eigenvector that we want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.



Fig(3.2): Plot of the normalized data with the covariance eigenvectors laid on top

In our programme have taken the first three coloumns of the feature vector. Our goal was to choose as less dimensions as possible, but in the same time we have to consider that we should not loose much data. Number of Principal Component allocated to normal subspace for our programme is three and it describes all the data without any error. By this we get our feature vector R.

Given our example set of data, and the fact that we have 2 eigenvectors, we have two choices. We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

v) Deriving the new data set:

This the final step in the PCA, and is also the easiest. Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set.

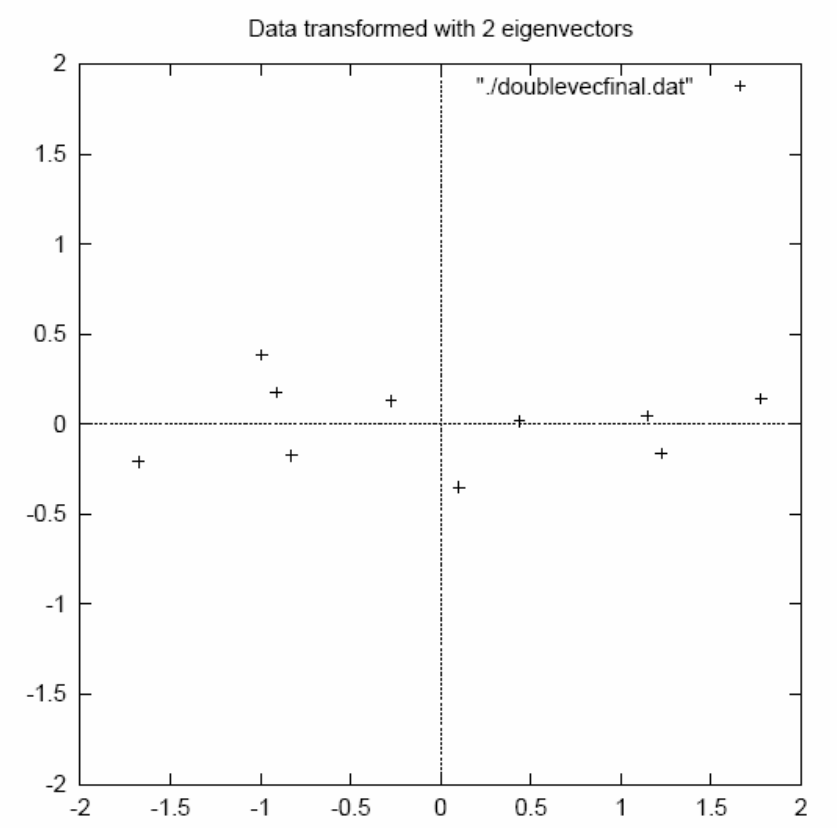
Final data=Row feature vector ●Row feature vector` ● Adjusted data

Where Row feature vector is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and data adjusted is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension.

Table(2.2) : Transformed data (2 eigenvectors)

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287



Fig(3.3): The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Table(2.3) : Transformed data (1 eigenvector)

Transformed Data (Single eigenvector)
x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

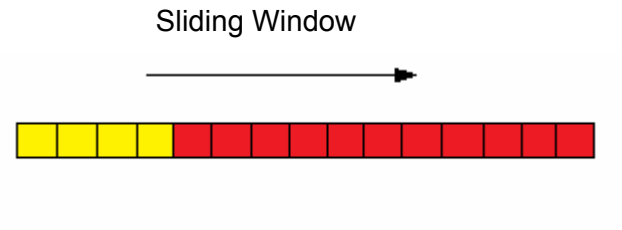
Some other parameters were required to be calculated:

We calculated the projection of our data X on to the feature vector R, that is stated as X_hat.
X_tilde which is the projection of X on the remaining value of the data that we did not use.
X_state which is the magnitude of the projections.
X_residual which is the magnitude of the X_tilde.

Finally we compared the value of X_residual with a manually set threshold value. The threshold was manually set since we were unable to develop a dynamically adaptive algorithm for it. Hence, for every new backdrop of an image the value has to be manually set after observing a test run. When the value of X_residual is greater than the value of the threshold then the image is detected as anomalous.

(5.1.1) THE ONLINE PCA

For the purpose of providing training data to the algorithm to create an initial benchmark we have incorporated into our algorithm a sliding window. This sliding window will buffer in its memory a small number of frames of only the environment in which it is to be place. This buffer will refresh itself after a certain predetermined number of frames follow it. The subsequent images are then mapped back onto the training data This is done so as to incorporate in the algorithm the option for allowing subtle changes in the environment.



Fig(3.4): Online PCA, sliding window.

$$\mathrm{Im}\,ageMatrix = \begin{pmatrix} \mathrm{Im}\,ageVec1 \\ \mathrm{Im}\,ageVec2 \\ . \\ . \\ \mathrm{Im}\,ageVec200 \end{pmatrix}$$

Some Advantages of PCA:

1. It helps to define the pattern of the data easily.
2. It express the data is such a way that it is very easy to find the similarities and differences, since patterns in data can be hard to find in data of high dimension.
3. Reduces the number of dimensions, without much loss of information.

(6.0) ONE CLASS NEIGHBOR MACHINE
Kth-NEAREST NEIGHBOR

The second algorithm we used for our thesis is the “Kth Nearest Neighbor Algorithm. It is an extremely simple and efficient algorithm. It works based on minimum distance from the query instance to the training samples to determine the nearest neighbors. After we gather K nearest neighbors, we simply take the majority of these neighbors to be the prediction of the query instance.

Kth-nearest neighbor is a supervised learning algorithm where the result of new instance query is classified based on majority of K-nearest neighbor category. The purpose of this algorithm is to classify a new object based on attributes and training samples. The classifiers do not use any model to fit and only based on memory. Given a query point, we find K number of objects or (training points) closest to the query point. The classification is using majority vote among the classification of the K objects. Any ties can be broken at random. Kth Nearest neighbor algorithm used neighborhood classification as the prediction value of the new query instance. It is commonly based on the **Euclidean** distance between a test sample and the specified training samples.

In mathematics, the **Euclidean distance** or **Euclidean metric** is the "ordinary" distance between two points that one would measure with a ruler, which can be proven by repeated application of the Pythagorean theorem. By using this formula as distance, Euclidean space becomes a metric space The associated norm is called the **Euclidean norm.** Older literature refers to this metric as **Pythagorean metric**. The technique has been rediscovered numerous times throughout history, as it is a logical extension of the Pythagorean theorem.

N-dimensional distances:

For two N-D points, $P = (p_1, p_2, ..., p_n)$ and $Q = (q_1, q_2, ..., q_n)$, the distance is computed as:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}.$$

It assumes a sample set S comprising T , F dimensional data points, {xt}T t=1. The algorithm requires the choice of a sparsity measure, denoted by g. Example choices of a sparsity measure are the K-th nearest neighbor, Euclidean distance and the average of the first k nearest-neighbor distances. The nearest neighbor algorithm sorts the values of the g measure for the set of points S, and it identifies the point that is different from other points. Those points indicate anomalies.

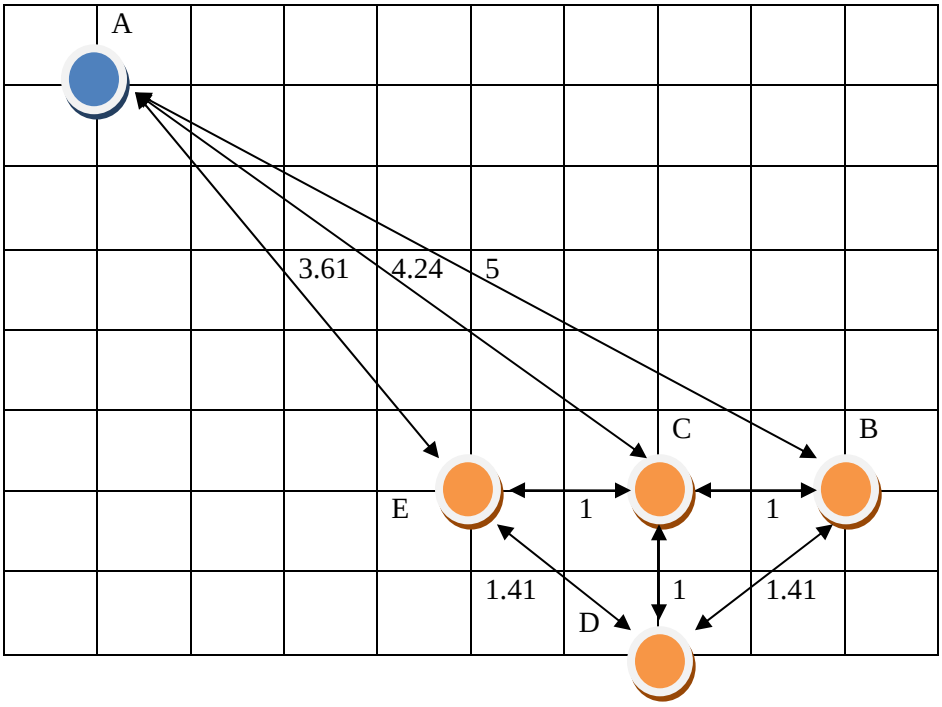
If the K-th nearest neighbor distance function is used as the sparsity measure, the OCNM algorithm involves calculating the distance from every point xt to every other point in the sample set. As each point is F-dimensional and there are T timesteps, the complexity is O(T 2F).

(6.1) PROJECT APPLICATION OF OCNM

Here is step by step on how to compute K-nearest neighbors KNN algorithm:

1. Determine parameter K = number of nearest neighbors.
2. Calculate the distance from every point xt to every other point in the sample set..
3. Average of the first k nearest-neighbor distances are determined.
4. Sort the Average distance and determine nearest neighbors based on the Kth minimum distance.
5. If the distance exceeds a particular threshold then that image is detected as anomalous.

An example is given in 2-D space since there is the luxury of graphical representation:



Fig(4.1) : Detecting an outlier using nearest neighbor

The distance from every point xt to every other point in the sample set and Average of the first k nearest-neighbor distances are tabulated in the following table. The table is done by taking k=2.

Table(3.1): Distances of all Neighbors

	point A	point B	point C	point D	point E	Average of the 1st K nearest neighbor
point A		5	4.24	5	3.61	3.93
point B	5		1	1.41	2	1.21
point C	4.24	1		1	1	2.61
point D	5	1.41	1		1.41	1.21
point E	3.61	2	1	1.41		1.21

Sort the Average distance and based on the K-th minimum distance the anomalous point is determined:

Table(3.2): Sorted Nearest Neighbors

	The sorted distance of Average of the 1st K nearest neighbor(k=2)
point A	3.93
point C	2.61
point B	1.21
point D	1.21
point E	1.21

Thus it can be found that the point having the maximum Euclidean distance (POINT A) is an anomaly from all the points. The rest of the points having the same Euclidean distance is assumed to lie in the same cluster.

Since images are multidimensional so the Euclidean distance is found by N-Dimensional method which is stated above. Since our data matrix has multiple dimensions so it is difficult to show the calculations of the Kth Nearest Neighbor with our data.

In our program we determined the number of nearest neighbor, K=5.Then the Euclidean distance from every point xt to every other point in the sample set was calculated, which is the g_list. Then the values were sorted in ascending order. When the value exceeds the threshold value then the image is shown as anomalous. The threshold is a user defined value.

Advantages of using k nearest neighbor algorithm:

1. Robust to noisy training data.
2. Effective if the training data is large.
3. Because the process is transparent, it is easy to implement and debug
4. In situations where an explanation of the output of the classifier is useful, k-NN can be very effective if an analysis of the neighbours is useful as explanation
5. There are some noise reduction techniques that work only for k-NN that can be effective in improving the accuracy of the classifier.

Disadvantages of using k nearest neighbor algorithm

1. Because all the work is done at run-time, k-NN can have poor run-time performance if the training set is large.
2. k-NN is very sensitive to irrelevant or redundant features because all features contribute to the similarity and thus to the classification. This can be ameliorated by careful feature selection or feature weighting.
3. On very difficult classification tasks, k-NN may be outperformed by more exotic techniques such as Support Vector Machines or Neural Networks.

(7.0) ALGORITHMS

Below are 3 different algorithms used for our project, any one can be used at a given instance based on what classifier is to be used.

(7.1) Anomaly detection using Principal Component analysis:

```
clc
clear all
jpgFiles = dir('C:\MATLAB7\work\*.jpeg');
for k = 1:length(jpgFiles)
  filename = jpgFiles(k).name;
  a= double(imread(filename));
  a1=dwt2(a,'haar');
%    a1 = fft2(a);
%   a2=imresize(abs(a1),0.1,'bil');

a2=imresize(a1,0.1,'bil');

[m n]=size(a2);
a2_expand=[];
for r=1:m
a2_expand=[a2_expand a2(r,:)];
end
X(k,:)=[a2_expand];
end

[T f] = size(X);

%X = X_P'; %Transpose X, if data matrix is in transposed form
```

```
X = X - repmat( mean(X,1) , size(X,1) , 1 ); %Then subtract off each dimension across all timesteps.

C = X'*X;
[V D] = eig(C); %Columns of V are the e-vectors
d = diag(D);
V = fliplr(V);
d = flipud(d);
D = diag(d);

normSquare = sum((X*V).^2);
var = normSquare/sum(normSquare); %Percentage of variances
u = X*V;
u = u ./ repmat( sqrt(normSquare) , size(X,1) , 1 );

r=3; %Number of Principal Components allocated to normal subspace
R = V(:,1:r);
X_hat = R*R'*X'; %Projections
X_tilde = ( eye(size(R,1)) - R*R' ) * X';
X_state=sum(X_hat.^2,1);
X_residual=sum(X_tilde.^2,1);
z=1;
for i=1:length(jpgFiles)
   if X_residual(i)>4*10^4
 %   if X_state(i)>0.5*10^5
      figure(i)
      imshow(jpgFiles(i).name)
            if z==1

%         [y,Fs] = wavread('AVSEQ01');
%           wavplay(y,Fs);
      end
      z=2;
```

```
   end
end


figure(i+1)
subplot(3,1,1);stem(var(1:5)); title('Variance');
subplot(3,1,2);stem(X_state); title('X_state');
subplot(3,1,3);stem(X_residual); title('X_residual');
```

(7.1.1) Anomaly detection using the online PCA:

```
clc
clear all

jpgFiles = dir('C:\MATLAB7\work\*.jpg');
for k = 1:length(jpgFiles)
 filename = jpgFiles(k).name;
 a= imread(filename);
 a1=dwt2(a,'haar');
 %a1 = fft2(a);
 % a2=imresize(abs(a1),0.1,'bil');
 a2=imresize(a1,0.1,'bil');

[m n]=size(a2);
a2_expand=[];
for r=1:m
a2_expand=[a2_expand a2(r,:)];
end
P(k,:)=[a2_expand];
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Training Period
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X = P(1:20,:);
[T f] = size(X);
[T f] = size(X);
[T f] = size(X);

%X = X'; %Transpose X
X = X - repmat( mean(X,1) , size(X,1) , 1 ); %Then subtract of each dimension across 864 timesteps.

C = X'*X;
[V D] = eig(C); %columns of V are the e-vectors
d = diag(D);
V = fliplr(V);
d = flipud(d);
D = diag(d);


normSquare = sum((X*V).^2);
var = normSquare/sum(normSquare); %Percentage of variances
u = X*V;
u = u ./ repmat( sqrt(normSquare) , size(X,1) , 1 );

r=4;
R = V(:,1:r);
X_hat = R*R'*X'; %Projections
```

```
X_tilde = ( eye(size(R,1)) - R*R' ) * X';
X_state=sum(X_hat'.^2,1);
X_residual=sum(X_tilde.^2,1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run for future timesteps
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X = P(20:59,:);
[T f] = size(X);
%X = X'; %Transpose X
X = X - repmat( mean(X,1) , size(X,1) , 1 ); %Then subtract of each dimension across all timesteps.

X_hat = R*R'*X'; %Projections
X_tilde = ( eye(size(R,1)) - R*R' ) * X';
X_state=sum(X_hat'.^2,1);
X_residual=sum(X_tilde.^2,1);


% for i=1:(length(jpgFiles)-20)
%     if X_residual(i)>4*10^4
%         figure(i)
%         imshow(jpgFiles(i+19).name)
%     end
% end


figure(i+1)
subplot(3,1,1);stem(var(1:5)); title('Variance');
```

```
subplot(3,1,2);stem(X_state); title('X_state');
subplot(3,1,3);stem(X_residual); title('X_residual');
```

(7.2) Anomaly detection using Kth nearest neighbor:

```
clc
clear all

jpgFiles = dir('C:\MATLAB7\work\*.jpeg');
for k = 1:length(jpgFiles)
  filename = jpgFiles(k).name;
  a= double(imread(filename));
  a1=dwt2(a,'haar');
%   a1 = fft2(a);
%   a2=imresize(abs(a1),0.1,'bil');

a2=imresize(a1,0.1,'bil');

[m n]=size(a2);
a2_expand=[];
for r=1:m
a2_expand=[a2_expand a2(r,:)];
end
X(k,:)=[a2_expand];
end
% size(X)
% imshow(jpgFiles(8).name)
nu=2; k=3;
X = X./repmat(sqrt(sum(X.*X,2)+eps),1,size(X,2)); %normalize to unit circle (i.e. divide by norm)
n = size(X,1);
h_i = zeros(1,n);
```

```
%Get distances to k'th nearest neighbour
clear g g_list;
for i=1:n
   [g(i,:) g_list(i,:)]= M1(X(i,:),X,k); %Sparsity measure M1 is the Euclidean distance
end %for i=1:n

g_sorted = sort(g, 'ascend');

z=1;
for i=1:length(jpgFiles)
   if g(i)>0.03
      figure(i)
      imshow(jpgFiles(i).name)
      if z==1
%
%        load gong;
%       %load chirp;
%        y1 = y; Fs1 = Fs;
%
%        wavplay(y1,Fs1,'sync') % The chirp signal finishes before the
%        wavplay(y,Fs)
%         [y,Fs] = wavread('AVSEQ01');
%            wavplay(y,Fs);
      end
      z=2;
   end
end
figure(i+1)

subplot(2,1,1);stem(g); title('g');
%figure(2)
subplot(2,1,2);stem(g_sorted); title('g_sorted');
```

(8.0) EXPERIMENTAL RESULTS

Our results are very accurate, spotting anomalies with precision. Initially we tested our algorithm using several series of images obtained from a number of webcams collecting data simultaneously. After final completion and rigorous testing of our algorithm we chose BRAC University's closed circuit cameras for video footage for our final data set. However, the videos from 4 cameras were docked into one video sequence. Since the images were pre-grouped and pre-minimized we expected less accurate results. This was not the case; every anomaly was detected including the passing of a car that would have gone unnoticed to the human eye. This shows the robustness of the algorithm, working with low resolution cameras.

Fig(5.1): Normal Data

Fig(5.2): Anomalous Data

Below is a time comparison of the two algorithms using a set of 100 images for both runs. The algorithm was run on a Laptop PC with the following configuration:

Core 2 duo 2.0 GHz, L2 cache 2 MB

4 GB DDR II RAM

Table(4.1) : Algorithm Runtime comparison

Time	Haar Wavelet transform (Seconds)
PCA (Total)	52.603000
Data set preparation	47.023000
KNN(Total)	53.205000
Data set preparation	43.657000

(9.0) Conclusion

Effectively we have chosen the Principal Component Analysis in conjunction with the Haar wavelet transform as the prime candidate for our algorithm. This algorithm has proved itself to be a robust solution to the automation of video surveillance, providing pinpoint accuracy using minimum digital resources. At the same time it offers to be an economic solution, providing stable results when implemented with low resolution cameras.

(10.0) Future Work

We aim to follow up on our research through production of a fully functional commercial version of the algorithm using the online PCA. A problem we hope to overcome is the calculation of the threshold value that we hope our next algorithm will incorporate in itself a function to dynamically set and assign a threshold for differing environments.

LIST OF REFERENCES

[1]. T. Ahmed, B. Oreshkin and M. Coates, “Machine learning approaches to network anomaly detection,” in Proc. USENIX Workshop on Tackling Computer Systems problems with Machine Learning Techniques (SysML), Cambridge, MA, Apr. 2007. [internet] Available at: https://www.usenix.net/events/sysml07/tech/full_papers/ahmed/ahmed.pdf [Accessed 27 December 2008].

[2]. Park Sudhakar, “Discrete Wavelet Transform”, Nov. 2003. [Accessed 27 December 2008].

[3]. Lindsay I Smith, February 26, 2002. A tutorial on Principal Components Analysis [internet] Available at : http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf [Accessed 29 December 2008].

[4]. Vinay Kumar and Manas Nanda, 15 September,2008.Image Processing In Frequency Domain Using MATLAB®: A Study For Beginners.[internet] Available at : http://hal.archives-ouvertes.fr/docs/00/32/16/13/PDF/IMAGE_PROCESSING_IN_FREQUENCY_DOMAIN_USING_MATLAB_A_STUDY_FOR_BEGINNERS.pdf

[5]. Kardi Teknomo, PhD, 2008, K Nearest Neighbors Tutorial, [Online] (Updated 10 Jan 2006) Available at: http://people.revoledu.com/kardi/tutorial/KNN/index.html [Accessed 27 December 2008]