MULTI IMAGES RETRIEVED FOR AUTOMATIC INTRUDER DETECTION USING KERNEL-BASED LEARNING ALGORITHM

A thesis submitted to the Department of Electrical and Electronic Engineering

By:

S.M FARDIN SHAHREAR (ID-12110021)
KAZI LAMIYAH DARAKSHA KARIM (ID-12110020)
RAKIBUN MUNTAHA (ID-12110031)
AFRA ANIKA (ID-12110019)

In partial fulfillment of the requirements for the Degree of

Bachelors of Science in Electronic and Communication Engineering

School of Engineering & Computer Science

August 2016

BRAC University, Dhaka, Bangladesh

Declaration of Authentication

We certify that the project based thesis titled, “MULTI IMAGES RETRIEVED FOR AUTOMATIC INTRUDER DETECTION USING KERNEL-BASED LEARNING ALGORTHIM” on the arena of image processing has been a bona–fide work under the supervision of Mr. SupriyoShafkat Ahmed. All sources and knowledge for this paper which were found by other researchers are acknowledged by reference. Materials of work such as images, figures, tables and citations in this paper are accepted by our Supervisor. We hereby declare that this thesis has not been previously submitted neither in whole or in part, for any other degree or publication.

Signature of the Supervisor

Supriyo Shafkat Ahmed

Signature of Authors

S.M Fardin Shahrear

Kazi Lamiyah Daraksha Karim

Rakibun Muntaha

Afra Anika

Dedication

This thesis is dedicated to our faculties and parents
For always believing in us, encouraging us and inspiring us
To achieve our goals and bring out the best.

Acknowledgement

We would like to take this opportunity to convey our gratitude to our supervisor Mr. Supriyo Shafkat Ahmed for his immense patience, ample time, guidance and encouragement in conducting the project and preparation of the report. We also wish to acknowledge Mr. Tarem Ahmed for providing us the support and valuable knowledge on the arena of anomaly detection throughout the course of our project.

A special thanks to Ms. Sanjida Sabah for the endless motivation. Last but not the least we are very grateful to our parents and friends for their inspiration and support and also BRAC University for giving us the opportunity to complete our B.Sc. degree.

Abstract

Over the course of the last few decades, security systems have undergone radical overhauls and have transitioned into such sophisticated devices that some even completely overcome any form of human interventions. In this paper we apply a recursive Kernel-based Online Anomaly Detection algorithm to propose an automated, real-time intruder detection mechanism for surveillance networks. Our proposed method is portable and adaptive, and does not require any expensive or advanced components. Real images are collected using a rotating camera over a span of space and time, along with the comparison using common methods based on Principle Component Analysis (PCA), can show that it is possible to obtain high detection accuracy with low complexity. Thus, it could also exhibit sensitivity to threshold choices, and have no natural distributed form.

Table of Contents

CHAPTER-06 LIST OF ILLUSTRATION

CHAPTER-01 INTRODUCTION

There has been a remarkable surge in the usage of extensive network in recent times correlated upon by the fact of rapid growing development of security network over the years. It is needless to mention that the requirement for awareness application become inherently undeniable. The whole world is somehow now attached towards surveillance network. Beijing, London, Chicago, Houston and New York are the five topmost countries that are enriched under security network. Approximately 5, 70,000 cameras has been installed in Beijing whereas UK ensemble around 1.85 million cameras. The word ‘safety’ is the primary concern across the globe. In spite of the progression of modern technology, security ordeals are still endless across the world. For an operator the endeavor upon multiple images to be configured simultaneously is a long process which rather came up with the fault of unnoticed activity that might be suspicious.

The study correlated under specific research on visual automated surveillance camera and few specific algorithms proved the certain criteria with specific results on such research. The issues in automated visual surveillance have been proposed fundamentally with its particular specification. Valera and Velastin in the paper of Advanced Video and Signal Based Surveillance initiates third generation system [1]. The aim of research is to relate an algorithm with the issues based on real time security system processing.

1.1 Motivation and Objectives

Many times in this wide world there are certain issues which are unable to detect perfect images in a system under visual surveillance cameras. From a certain period of time, it becomes difficult for a person to find error in a certain image which might be because of human error. Due to this fact sometimes mysterious activity is left to be unnoticed. The purpose of this research is to modify an applicable algorithm to automatically detect suspicious activity in an image under dynamic situation. Human operator need to be minimally active during this procedure and certain alarming function applied into the algorithm to signal any sort of activity which would be regarded as abruptly abnormal object in the image.

1.2 Thesis contribution

The above mentioned objective in this thesis is based on a real time learning algorithm known as Kernel-based Online Anomaly Detection. KOAD is an algorithm which is able to learn sequentially and customizes a set of features that spans the subspace of normal behavior in an image. It adapts a dictionary of features that spans the subspace of normal behavior approximately. The algorithm signals an anomaly when there is any deviation with regard to the normal value. There are certain paper written on KOAD where certain criteria are expect to fulfill. Ahmed et al. presented a paper where KOAD and anomaly detection in IP networks are introduced [2]. In addition Ahmed et al. has correlates this algorithm and suggested to detect incident on road traffic [3]. Herea sequence ofdifferent

position images collected from a video by the use of the mover and a webcam. Initially we set the input parameters of KOAD to determine the output parameters. Research was taken to extract images from a moving camera and form a frame of 10 images to focus out a stitching view fixed at an angle of the room or any place. It has been shown that it can detect anomaly based on real time with a high level of accuracy and a low false alarm rate. The more the images the better the algorithm run and it application is enhanced into portable function as well. That is one of the reason we tried to input a long stream video clip as possible. In addition the contribution in this thesis expands into detecting face from anomaly images marked with red border.

1.3 Synopsis

In the following chapters, chapter- 01 is the introduction to our project. Followed by chapter -02 is merely the background we have researched to establish this paper with efficient knowledge. Chapter- 03 covers the theoretical framework behind the project implementation, which is explained in chapter -04.

With the Results and discussion in chapter -05, we have concluded our project report and also have shown precise evidence on starting to work on our future perception of this subject.

CHAPTER-02 BACKGROUND

With the advent of various program under computational activities the interest upon different algorithm lead under a huge platform. Principle that relies upon machine learning was the fundamental start up algorithm in the field of visual surveillance. This is the one of the first recursive application known under machine learning phenomenon followed by the principles of Kernel recursive Least squares [4].The algorithm extensively used in signal processing method and named as effective on line method for finding linear predictors which minimized the mean squared error . Kernel recursive least square algorithm performs linear regression in the feature space. The storage and the computational complexity in Kernel online anomaly detection is time independent.There are few other algorithms where principle component analysis was first used to do reduction dimensionally and in addition the degree of anomaly of each image determined one class Support Vector Machine (SVM) which was included in the technique of Sudo et al. [5]. Another technique involved the potential of performing supervised learning using SVM to detect images that show weapon analyses by Poh et al. [6].

Another method of abnormality detection is developed by Breitenstein et al. [7] in which supervision by a human is not required at all. The normal or usual situation or model is developed by meaningful nearest neighbors and to effectively compare the observations, an incremental learning technique is used to adapt to the change in the data stream.

It is always a hard task for an individual operator to detect events that are unusual by monitoring each screen continuously. There are often situation that are dealt with individual operator deals with detection criteria there still left a human error which are not negotiable even if the control screen is in a large multi-screen. Schuster's paper proposed a method for an automatic identification of unusual region in real time video streams data [8]. A pan tilt zoom camera can help to get a frontier image that helps a better real time detection capability of unusual event. Different categories of static and active camera are used for determining events and are used autonomously in an incremental learning method.

Image mining proposed by Gool et al. is different from the normal query method of processing. This method uses data that are coming from devices that produce images comes out from a single location continuously in a range of time which is webcam. This process also correlate the factor of exploring information from the landmark images kept in the public repositories [9]. Kratz and Nishino introduced a novel account for modeling the motion pattern of extremely crowded scenes and in relation to a detection of abnormal event [10].

Chao and Jun used an automated distributed real-time video surveillance system. This is the architecture where target leverages upon the two systems under intelligence video surveillance system. Firstly, agent process information that is raw based on each of the surveillance terminal. The result that comes out from processing transmitted through IP network. Second factor is to ensure about the transmitted control information that depend on the information processing units. The system gives a fast response and information loss is minimized [11].

CHAPTER-03 THEORETICAL FRAMEWORK

The concepts of this paper rely under the certain criteria of algorithm named as KOAD and PCA. The main pattern of this framework depends under the algorithm of KOAD and also use PCA algorithm for comparison under calculated data. PCA is used for comparison because it is viable in terms of many research works and it was just the previous implemented work of KOAD.

3.1 Principle Component Analysis (PCA)

By the name itself, we know Principal Component Analysis (PCA) means to find the principal components of the data set. It is a tool which analysis patterns where there is more data spread and most variance of points, which means high dimensional space data such as images. Consideration of the normal and anomalous behavior, PCA widely used to separate the space occupied by the set of the input vectors into the pattern of two disjoint subspaces. When the magnitude of the projection onto the anomalous subspace exceed a PCA Q-statistic threshold an anomaly is signaled [12].

The obtained eigenvectors are the representation of the principal components used by PCA. The denotations of the components are done by classifying the eigenvectors according to their corresponding magnitude, the Eigenvalues. So, the eigenvector with the highest eigenvalue is the first principal component. Also, the number of eigenvectors/values that exist equals to the number of dimension present in that dataset.

For instance, if we find two eigenvectors/values in a dataset, then the image must be two-dimensional (2D) and similarly for three eigenvector/values, the image is 3D. As a result, the numbers of principal components are the number of dimensions.

PCA uses eigenvector concept and linear regression to draw a straight, explanatory line. Before implementing the reduction of dimensions, PCA prioritizes the dimensions in the descending order of principal components by allowing dropping some of the low variances. For example, if we plot a 3D graph with x, y, z dimensions, and then we find three eigenvectors/values followed by the three dimensions. Now, if we consider one of the eigenvector has the largest eigenvalues, the other has non-zero eigenvalues and the last eigenvector has zero eigenvalues. According to priority, the eigenvector with the largest eigenvalues, have the maximum variance and thus it is considered to be the first principal component. PCA helps to plot a best fit line with most variance and this is the longest dimension of the data set. Now the vector with non-zero eigenvalue is the second principal component and PCA again plots a straight line which cuts through the data at 90 degree angle to the first principal component, converging the errors made by the first. Lastly, the errors produced by the first and second components are converged to a single line and represented as the third principal component. The old axes are rearranged using only the first and second vectors, as the third vector is considered nonexistent for its zero eigenvalue. Hence, the 3D image is reduced to 2D image for the elimination of the third component/eigenvector/value/dimension. Similarly, for a 2D image, we prioritize the principal components depending on the eigenvalues. The one with large variance is the first component and a best fit line is drawn. The other is the error fitting line which is considered nonexistent when using PCA. Therefore, the data is interpreted as a reduced one dimensional data (1D).

3.1.1 Online PCA

The sole aim to use online PCA in the algorithm is to create a sliding window which will buffer along with small number of frames only to detect the environment it is taking place. With the increase number of frames, the buffer will refresh itself. The new sets of data are then mapped back on to the original data. Hence, with every environmental change, the set of predetermined data set changes. This helps to determine the pattern with reduced number of dimensions and minimal loss of data.

3.2 Overview of Kernel Algorithm

Kernel methods are a new sort of pattern analysis algorithm to analyze the relations such as clustering, correlation etc. in any type of data such as images, vectors or points. It is a nonlinear transformation of data into a high dimensional vector space. The idea of kernel is to reach that specific space without any complications. It's best in classification and regression analysis for the supervised learning model known to be Support Vector Machine (SVM). To mark the space in the computation, we need the inner products through the regular quadratic programming problem and for Kernel mapping it is made simpler for the built-in tool Kernel Trick [13]. Basically, kernel trick only aims for the inner product between two vectors, so that it could replace the dot product with a Kernel function. The kernel function denotes an inner product in feature space and is usually denoted as:

$$K(x,y) = \varphi(x),\varphi(y) \qquad (1)$$

The dimensionality of the problem will depend on the number of data being used, not on the dimensionality of the space. After getting the result we count the number of support vectors, which again depends on the number of data not the dimensionality and this helps to map the algorithm into a higher dimension even if the space could be infinite. However, Kernel functions must be continuous, symmetric and most importantly, the matrices should have non- negative eigenvalues.

Firstly, we need to obtain the inner products of the space to get the actual vectors. If map chosen suitably, complex linear relations can be simplified and easily detected.

The simplest kernel function among the many types is Linear Kernel Function. It gives the best classification performance for the text categorization. It is simple and easy to map the large number of instances and features of data into a higher dimensional space, rather than focusing on the performance. It is denoted by the inner product <x, y> with an optional constant C.

$$k(x, y) = x^T y + c \tag{2}$$

When using SVM with the assistance of linear kernel, the classification is faster than with another kernel. This is because LinearSVM commonly uses a specific library known as LibLinear which only needs to improve the C tuning parameter and the *y* parameter optimization is used by other kernels. However when using it with PCA, it performs just like the standard PCA. Standard PCA effortlessly rotates the set of points around the mean for the best fit regression so that linear transformation can occur into the few dimensions. The rest of the unfit points are highly correlated and may be dropped with minimum loss of information.

The most common form of radial basis function is a Gaussian distribution, calculated as:

$$K(\vec{x}, \vec{z}) = e^{-(\vec{x}-\vec{z})^2/(2\sigma^2)} \quad (3)$$

A Radial Basis Function (RBF) is equivalent to mapping the data into a Hilbert space. It is a common function used in SVM. Usually, kernel trick does not support mapping large numbers of features in the input space. With the denotation of function *z*, it maps a single vector into a higher dimension.

A non-stationary kernel is known to be polynomial kernel. It is usually used for data which are normalized.

$$k(x, y) = (\alpha x^T y + c)^d \quad (4)$$

Changing parameters are the slope alpha, the constant term c and the polynomial degree d.

3.2.1 Kernel-based Online Anomaly Detection Algorithm (KOAD)

As we know the most common mapping function used in learning machines is known to be Kernel algorithm. It uses the concept of “kernel trick” to map the input data onto a specific space of a much higher dimension with the goal that the same behavioral points will cluster in that space.

For instance, if a set of multivariate measurements such as $\{\mathbf{x}_t\}_{t=1}^{T}$ is mapped on a space *F*, the similar behavioral points are expected to cluster denoted as such,

$\{\Phi(\mathbf{x}_t)\}_{t=1}^{T}$. This makes it easier to detect the region of normality in that space using a small dictionary of approximate number of linearly independent values, $\{\Phi(\tilde{\mathrm{x}}_j)\}_{j=1}^{m}$ [4]. The data entered in the dictionary which are $\{\mathbf{x}_t\}_{t=1}^{T}$ are represented by $\{\tilde{\mathrm{x}}_j\}_{j=1}^{m}$. Here,*T*, the number of time steps has to be more than the size of the dictionary, *m*. The purpose for this inequality saves the time for computation and also saves storage. Therefore, $\Phi(\mathbf{x}_t)$ [14], the feature vector for this particular set has to satisfy the equation below to be known as linearly dependent on $\{\Phi(\tilde{\mathrm{x}}_j)\}_{j=1}^{m}$.

$$\delta_t = \min_{\mathbf{a}} \left\| \sum_{j=1}^{m} a_j \phi(\tilde{\mathbf{x}}_j) - \phi(\mathbf{x}_t) \right\|^2 < \nu. \tag{5}$$

Where, $\{a_j\}_{j=1}^{m}$ is the optimal coefficient vector, $\delta_{t,}$is the projection error and*v*, is the threshold value.

Similarly, the function of Kernel-based Online Anomaly Detection (KOAD) algorithm is to run Kernel function on every time step, *t*, on a given vector, $\mathbf{x}_t$. Initially, it evaluates the projection error $\delta_{t,}$ of the input $\mathbf{x}_t$ on to the featurespace dictionary using the above equation ( ), which uses the kernel function without the parameters of the feature vectors themselves.

The error obtained from the computation, $\delta_{t,}$ is compared with two thresholds, v_1 and v_2. We make sure that $v_1 < v_2$, setting v_1 at a fixed point and v_2 is varied along the data set. If $\delta_t < v_1$, KOAD assumes that $\mathbf{x}_{t,}$ is linearly dependent on the dictionary and it is stated as a

normal behavior. If $\delta_t > v_2$, it is assumed that $\mathbf{x}_t$ is not even near to the normality and it immediately raises a “Red1” alarm to notify for an anomaly.

Suppose $v_1 < \delta_t < v_2$, KOAD then concludes that $\mathbf{x}_t$ is independent from the dictionary and there may be unusual activity. So, it notifies us by raising an “Orange” alarm and at the same time KOAD keeps record of the relevant input vectors that are corresponding to the ℓ time steps. These detected inputs are kept for later inspection. As the time step moves from t to $t + \ell$, the dictionary may have changed and this also changes the value of δ_t. When KOAD re-evaluates the error δ, for the time step time step $t + \ell$, it obtains a new δ. Now, these value of $\delta$is with v_1 and if $\delta < v_1$, KOAD lowers the orange alarm and the dictionary remains the same. If $\delta > v_1$, KOAD executes a “usefulness” test to determine the orange alarm which helps to compare between a new detection set as anomaly and with the one which the reason for change in region of normality.

To determine the usefulness of$\mathbf{x}_t$ the kernel values are observed by $\{\mathbf{x}_i\}_{i=t+1}^{t+\ell}$. When the kernel values are greater than threshold d, it is considered to be high and close enough to $\Phi(\mathbf{x}_i)$. If there are a large number of such values, then $\mathbf{x}_t$ cannot be considered as anomaly and it is allowed into the dictionary. In contrast, when almost all kernel values are low, then $\mathbf{x}_t$ may be considered very far from the space and considered as anomaly.

$$\left[\sum_{i=t+1}^{t+\ell} \mathbb{I}(k(\mathbf{x}_t, \mathbf{x}_i) > d)\right] > \epsilon\ell, \tag{6}$$

Here, $\mathbb{I}$ is the indicator function and$\in$ $(0, 1)$ is a selected constant.

Finally, in consideration of the equation, if the computation is valid then KOAD lowers the specific orange alarm to green indicating that there is no presence of anomaly and includes the data set into the dictionary. If it is not valid, KOAD changes the orange alarm to “Red2” alarm and deletes the out of order data from the dictionary as the region of normality expands. As this is a learning algorithm, as the time step increases, the past observations are replaced.

3.2.2 Application of KOAD

Set thresholds v_1, v_2; Enter x_1 to D;
For t=2,3,….**do**
Evaluate δ_t;
If $\delta_t > v_2$**then**
Raise Red Alarm;
elseif $v_1 < \delta_t \leq v_2$
Raise Orange Alarm;
Add $x_t$to D;
else /* $\delta_t \leq v_1$ */
/* Do nothing */;
endif
If*Orange* (x_{t-l}) **then**
Evaluate Usefulness of x_{t-l}over past l timesteps;
If *NOTuseful***then**
Elevate Orange (x_{t-l}) to Red;
Remove $x_{t-l}$from D;
else
Delete Orange (x_{t-l});
endif
endif
Remove any useless element from D;
endfor

CHAPTER- 04 PROJECT IMPLEMENTATION

4.1 Basic approach

The target of the project was to extract images from a video shot viewing the panoramic view of a 12x18 room, with or without any presence of anomaly. To achieve these goals, a high definition (HD) web camera was used on a camera mover. The mover was connected to the DC supplier and placed in a position from where the video was shot from one end of the room to another. At an angle of 180 degree, we shot a single 20 seconds video using web cam software, from which 10 images were extracted. Likewise, maintaining the same angle and same direction, total 4300 seconds video was shot for 2150 images. Using the software CyberLinkYouCam 6, every 10 images were extracted from one single shot panoramic video and stitched together as a single image. Thus, a database of 215 images was used as an array of images for our MATLab code implementation.

4.2 Parameter selection

Effectively Kernel is quite a critical measure in terms of homogeneity. According to prior knowledge choosing a kernel sometimes suggested as a very critical issue. It depends on different kind of invariance where Kernel follows a certain category. As mentioned kernel follow parameters that are automated based on model selection. The easiest procedure choosing a kernel is to select the model based on performance in the simple method. In our project under the implementation Kernel Choice 2, Gaussian kernel was used. There are many other parameters that tune the other further implementation. The

performance detection for maintaining the algorithm of KOAD is based on the values of threshold. The direct performance based on threshold v_1 on the other hand threshold $v_2$changes and ascertains the instant flagging of an anomaly. For different application the optimal settings for v1 and v2 vary. Time period does not consider a biggest factor as the performance remains approximately same within the same application. As Kernel represents learning algorithm optimum values set after a series of observation is given with well-known anomaly. The threshold when perfectly aligned gives easily 100% detection. The parameter on the choice of the resolution on orange alarm and the measurement on L and d depend on the storage resources and the time lag of the system. While modifying the implementation process of this project several other parameters involved to maintain accuracy. For instance when working with the images there were few environmental parameters that need to be kept balanced. The lagging of time when the mover rotates kept in a stable form. The angle was perfectly synchronized when the mover rotates and it was kept for about a rotation of 180 degree. The rotation speed was kept fixed under the range of the mover. In this way the whole system is controlled and maintained by altering this parameter.

4.3 Bilinear Interpolation (BIL)

Interpolation is method of numerical analysis which shows the use computer graphics. It fabricates new data point of any graph or image inside the range of distinct set of known data points. Their main ascribe is that they can easy to compute and stable. By interpolation we can get the estimate curve from the image data point. The idea is that the points are in some sense correct and lie on an underlying but unknown curve, the problem is to be able to estimate the values of the curve at any position between the known points.

Main core idea in the curve for interpolation would be the allocated point in an unknown curve. The difficulties face to estimate the values.

Suppose you have the following data:

X Y

3 8

12 17

22 27

32 37

42 47

52 57

By use of this regression model Y as a response to X. Using R: lm(y ~ x)

The results are tackling of 5, and a coefficient for x of 1. Which means an arbitrary Y can be calculated for a given X as X + 5. From the graph, we can see this way:

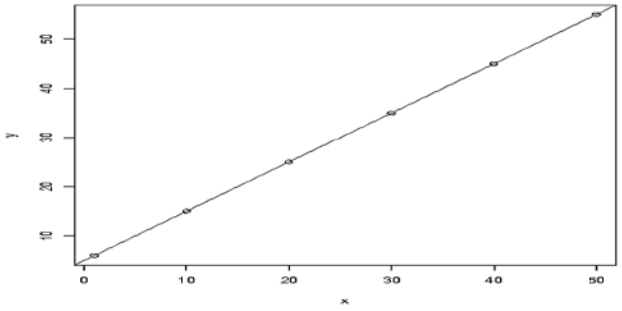

Figure 1. Sample data of regression analysis

To get a good quality image sometimes we require some types of interpolation or filtering. For example, when we scale an image, we can figure out the final color of each

pixel by using some basic advanced interpolation method. However, by this method we get more advance image quality.

4.3.1 Application of BIL

Image processing is like rotating, scaling or twisting done due to the movement of the pixels around the image. Suppose after interpolate an image for rotation, a pixel have the same color as pixel in the original image. The main problem is that there is no such thing as fractional pixels. Instead of simply choosing the color of the nearest available pixel, we can get better results by intelligently averaging the colors of the pixels.

In our project, the matrix is instructed to form a single image that is done in a row form after compression.In this way, we would feed into our anomaly detection algorithm where n images would result in a matrix of n rows. To solve this problem, we have employed to using bilinear interpolation.

4.4 Discrete wavelet transforms

Even though wavelet transform is similar to Fourier transform, they differ in the most significant function. Fourier transform interprets a signal in a form of sine and cosine, which is known to be the functions in Fourier space. Whereas wavelet transform uses the functions from both the Fourier and real space.

The wavelet transform could be denoted as below:

$$F(a,b) = \int_{-\infty}^{\infty} f(x)\, \psi^*_{(a,b)}(x)\, \mathrm{d}x \qquad (7)$$

Here, the * is the symbol of complex conjugate and function ψ is any function. This function can be chosen randomly provided that it follows certain protocols.

This apparently means wavelet transform uses a frame to cut down the signal produced. The frame is shifted over the signal using smaller frame than the previous and at every point the signal is calculated. This process is repeated several times and the result is a time domain signal with multiple resolution i.e. a representation in both time and scale resolution. All wavelets are derived from a single basic wavelet. For the digital applications, it requires small step translation which is possible in discrete wavelet transform. Discrete wavelet is denoted as below:

$$\Psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j}\right) \quad (1.2) \qquad (8)$$

Here, k is integers.

4.4.1 Haarwavelet

This wavelet is usually used for image compression as the method takes less space for storage. When performing any additional edit, very little or no computation is requiredfor its use of fewer bits. This digital image is a representation of 128 by 128 matrix which means it has 128^2 =16, 384 elements. It is further analyzed to be a 5-bit image with

2^5 =32 shades of grey, imposing it to be a large array. However, the Haar wavelet technique emphasizes on a sub matrix of the image, comprising the pixels in representation by numbers as a solution for easy transformation of the data.

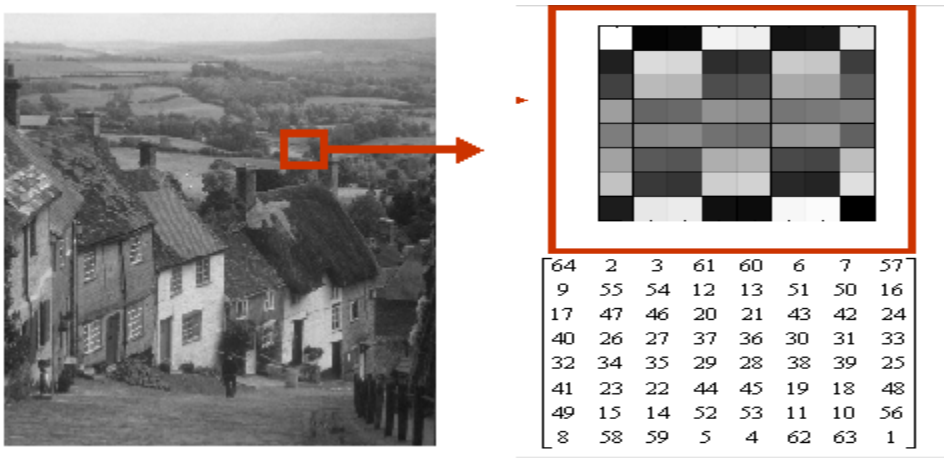

*Figure*2. Transformation from 128*128 images into 8*8 matrix image

The initial step to transform data is to apply a process called averaging and differencing to obtain a new matrix as a representation of the original image in a more compressed manner. This original matrix is represented by M.

The first row is the original data of the matrix M. The average of each pair from the first row are calculated and inserted as the first 4 numbers in the second row of a new matrix. The rest of the 4 numbers in the second row is subtraction of each average from the first number of their respective pairs existing in the first row. This numbers are known to be detail coefficients. However, the third row is constructed with the first number being the average of the first 2 pairs in the second row and second number being the average of the 3^{rd} and 4^{th} number. Similarly, fourth row consists of one number being the average of the two pairs created in the third row. The repetition of this process completes the row which has to be transformed and replaced in place of the original row. The same process is followed to compute for the transformed column [15].

First: The rows of matrix M

$$\begin{bmatrix} 64 & 2 & 3 & 61 & 60 & 6 & 7 & 57 \\ 9 & 55 & 54 & 12 & 13 & 51 & 50 & 16 \\ 17 & 47 & 46 & 20 & 21 & 43 & 42 & 24 \\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33 \\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25 \\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48 \\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56 \\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1 \end{bmatrix} \quad \begin{bmatrix} 32.5 & 0 & .5 & .5 & 31 & -29 & 27 & -25 \\ 32.5 & 0 & -.5 & -.5 & -23 & 21 & -19 & 17 \\ 32.5 & 0 & -.5 & -.5 & -15 & 13 & -11 & 9 \\ 32.5 & 0 & .5 & .5 & 7 & -5 & 3 & -1 \\ 32.5 & 0 & .5 & .5 & -1 & 3 & -5 & 7 \\ 32.5 & 0 & -.5 & -.5 & 9 & -11 & 13 & -15 \\ 32.5 & 0 & -.5 & -.5 & 17 & -19 & 21 & -23 \\ 32.5 & 0 & .5 & .5 & -25 & 27 & -29 & 31 \end{bmatrix}$$

Figure 3. Matrix of new rows after evaluating average and differencing.

Second: The columns of matrix M

$$\begin{bmatrix} 32.5 & 0 & .5 & .5 & 31 & -29 & 27 & -25 \\ 32.5 & 0 & -.5 & -.5 & -23 & 21 & -19 & 17 \\ 32.5 & 0 & -.5 & -.5 & -15 & 13 & -11 & 9 \\ 32.5 & 0 & .5 & .5 & 7 & -5 & 3 & -1 \\ 32.5 & 0 & .5 & .5 & -1 & 3 & -5 & 7 \\ 32.5 & 0 & -.5 & -.5 & 9 & -11 & 13 & -15 \\ 32.5 & 0 & -.5 & -.5 & 17 & -19 & 21 & -23 \\ 32.5 & 0 & .5 & .5 & -25 & 27 & -29 & 31 \end{bmatrix} \quad \begin{bmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & .5 & .5 & 27 & -25 & 23 & -21 \\ 0 & 0 & -.5 & -.5 & -11 & 9 & -7 & 5 \\ 0 & 0 & .5 & .5 & -5 & 7 & -9 & 11 \\ 0 & 0 & -.5 & -.5 & 21 & -23 & 25 & -27 \end{bmatrix}$$

*Figure*4. Matrix of new column after evaluating and differencing.

Once the transformation is completed and reversed, the result is a new 8 by 8 matrix such as:

$$N = \begin{bmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & .5 & .5 & 27 & -25 & 23 & -21 \\ 0 & 0 & -.5 & -.5 & -11 & 9 & -7 & 5 \\ 0 & 0 & .5 & .5 & -5 & 7 & -9 & 11 \\ 0 & 0 & -.5 & -.5 & 21 & -23 & 25 & -27 \end{bmatrix}$$

Figure 5. New 8*8 matrix transformation image shown.

The zero elements in the new matrix show that there is little or no variation of image in these regions from the original image.

However, by applying inverse of this N transform, the original data is retrieved. Despite being a technique of lossless compression, we set a non-negative threshold value [5] to prevent the loss of image quality in further compression.

$$\begin{bmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & \mathbf{0} & \mathbf{0} & 27 & -25 & 23 & -21 \\ 0 & 0 & \mathbf{0} & \mathbf{0} & -11 & 9 & -7 & \mathbf{0} \\ 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & 7 & -9 & 11 \\ 0 & 0 & \mathbf{0} & \mathbf{0} & 21 & -23 & 25 & -27 \end{bmatrix}$$

*Figure*6. Inverse of transformed matrix to retrieve original.

The inverse matrix of this averaging and differencing data set, results in a comparison of two data sets from the original matrix.

$$\begin{bmatrix} 64 & 2 & 3 & 61 & 60 & 6 & 7 & 57 \\ 9 & 55 & 54 & 12 & 13 & 51 & 50 & 16 \\ 17 & 47 & 46 & 20 & 21 & 43 & 42 & 24 \\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33 \\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25 \\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48 \\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56 \\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1 \end{bmatrix} \quad \begin{bmatrix} 59.5 & 5.5 & 7.5 & 57.5 & 55.5 & 9.5 & 11.5 & 53.5 \\ 5.5 & 59.5 & 57.5 & 7.5 & 9.5 & 55.5 & 53.5 & 11.5 \\ 21.5 & 43.5 & 41.5 & 23.5 & 25.5 & 39.5 & 32.5 & 32.5 \\ 43.5 & 21.5 & 23.5 & 41.5 & 39.5 & 25.5 & 32.5 & 32.5 \\ 32.5 & 32.5 & 39.5 & 25.5 & 23.5 & 41.5 & 21.5 & 21.5 \\ 32.5 & 32.5 & 25.5 & 39.5 & 41.5 & 23.5 & 43.5 & 43.5 \\ 53.5 & 11.5 & 9.5 & 55.5 & 57.5 & 7.5 & 5.5 & 59.5 \\ 11.5 & 53.5 & 55.5 & 9.5 & 7.5 & 57.5 & 59.5 & 5.5 \end{bmatrix}$$

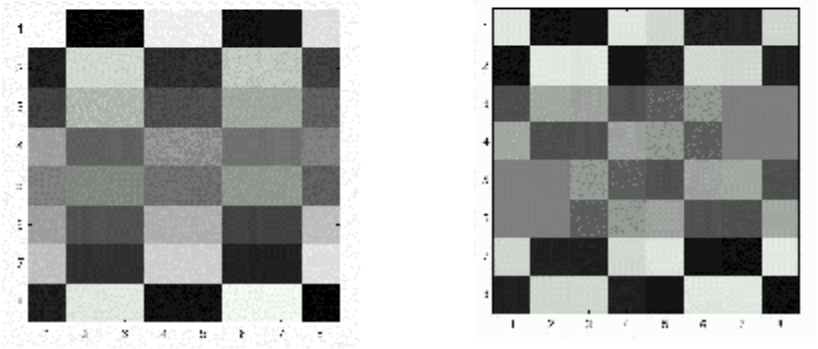

Figure 7. Comparison of the new reduced image with the original image.

4.4.2 Application of Haar Wavelet

In our project, the main aim was to transform the original data set into frequency domain in a compressed form, with minimal loss of data. The image undergoes the same process of transformation thus they remain closely same. This help us to use the principle

component analysis to process the data in the condition of less processing power without deviation.

4.5 Image Processing

Processing of image is one of the most crucial parts in this project. It is one of the rapid evolving subjects over the globe through which certain programmed lead into an emphasized condition. The goal is to make Kernel algorithm applied into dynamic image which was previously done with static images. The images are viewed under the range of continuous values. Most techniques under processing of images fall under two dimensional signals underneath the process of standard signal into it. Images are represented as a function over the finite range of dimension encapsulated within 2 – dimensional matrix. In actuality an image is defined as a function of real variables and amplitude at a position coordinate wise defined. An image is considered to be divided into regions and further its region is divided into dimension. This concept clarify out that the images are collection of objects. Simple statistical values of images and sub images are the intimate concept which required the topic of probability distribution. The probability distribution basically leads the distribution upon signal amplitudes. All over images are engulfed into a dictionary set of values out of which detection undergoes. There are certain ranges of ways about the classification under the characterization under image processing. Through several sort of transformation analysis image processing is ought to break an image into pixels. Inside this phenomenon probability density function and probability distribution function organized up for brightness and configuration of that region. The target is to make a video clip and bring out the images from within it under the range of 180 degree angle. When there are images the main convey is to put it inside

the algorithm and modify the configuration. Kernel based method is well applicable for the processing of images. Kernel in image processing is a simple matrix of two dimensional structures. The video clip composed broke down into images of 2 dimensional in numbers which is then modified by Kernel. Normally in image processing Kernel referred as a two dimensional matrix of numbers. There are several techniques that rely under the classification of images. For instances blurring, detection, sharpening depend upon Kernel. Kernel is fundamentally depending upon small matrixes of number and then these images are considered as a whole. An image is a two dimensional matrixes of pixels. Each pixel is constitute of a number and it configuration lies upon the format of an image. The main function of Kernel in image processing is to operate on these values of pixels and a new image is constructed. All around the procedure Kernel is used for extraction underneath a feature so that even a very small change in images can be detected. In this project we divided our processing of images into the following terms that are used along with MATLAB code:

1. Image stitch
2. Image resize

Image stitch: The paper presents image alignment to discover the corresponding relationship of images with degrees of overlap. Image stitching helps to understand the adjacent relationship among images with degrees of overlap. The need for image stitching is to make multiple images together and named it one. The problem was to have unlimited source of camera features to have a panoramic view. Thus the project undergoes few steps of procedures for image management. Prior to the view only image management is one of the subdivided parts required to give a view of the whole room. The target fulfilled when combination of multiple images lead to a combination of high

resolution images. The sub divider under the alignment or stitching algorithm elaborates the factor of pixel based and feature based algorithm. There are some probable issues related to image stitching. For instances the factor of illumination cannot be guaranteed to make a stitching smoothly. The complicated problem arises due to the parallax error, distortion in lens and the frequent motion in a scene and equipment lagging. There will be significant amount of overlap when the images get stitched. The main error occurs due to the variation of intensity in pictures of a frame. This also distorts the property of contrast and intensity. The widespread view of image stitching related to certain appropriate placement of algorithm. As there was slight change in orientation while taking video due to the error caused by the vibration of a mover stitching of images was necessary. Initially a set of collection of images from a video file are organized. The main instructions through which images are stitched determine by the pixel coordinates. According to panorama view Pixel from one image to pixel of other images coordinates together. Another thing to keep in mind for image stitching is to have a reliable composited base form where the main transform takes place. There are various sort of stitching software however our target fulfilled by using code in the MATLAB. There was certain mechanism involved inside the code for MATLAB to make one image from one rotation of a video clip. Initially the 2150 images are being set into the folder named into “jpgFileName”. Out of that the target is set to stitch 10 images into single images. By using a for loop under the if condition group of images is being set together. The degree of rotation for 180 degree took each clip in a range of 20seconds.

Images resize: In the field of digital processing resizing of image provide a well defined platform to maintain a smooth order of images for a build up procedure. Stitching

pictures lead to the phenomenon of resizing images under certain criteria in order to get a smooth mapping. A sequence of images is being taken from a video measured geometrically at an angle of 180 degree. Eventually it is viable to map the images into a correct resize frame so that the pictures are perfectly aligned. Image resizing helps to enlarge or reduce the image size. The theme under image resizing depends entirely upon the number of pixels in an image. Basically the size of the images that was extracted from video was regulated to 640*480 sizes. Stitching 10 images from 2150 image make a frame 1. This is how 215 frames were allocated together and every image contains expansion of 10240 pixels. By the function “imresize” in Mat lab the image return has number of row and column specified by the output size. The main target of resizing here depends to make all the images in same size so that they can be compressed together.

LOADING IMAGE

```
jpgFiles = dir(' E:\Thesis\collect of images \*.jpg');
%' E:\Thesis\collect of images \*.jpg'is the folder name where we store image
```

Here we put the direction of location folder using “dir” function in matlab. In this folder we put the entire collective image which we are going to process.

STICHING IMAGE

```
%READING IMAGE
jpgFileName = strcat('image- (', num2str(im), ').jpg');
   a = imread(jpgFileName);
```

After giving folder direction this matlab command start to convert as string to character value of the image name. And “imread” function read the image character from the location, which is saved as “jpgFileName”. Also it is important that which format image are store here. We are using here “jpg” format

.

```
sa= size(a); %get the size of the image
```

```
b= imresize(b,[sa(1) sa(2)]); %now resize 'b' as per the size of 'a' in order to get perfect sized image
```

Those commands are use for resize the image as required value before stitching. Here the image is being considered into two parts: left side and right side. It actually fixed the next image stitch position.

```
x1= [b a];% club the image b &a into another new image after making their size same
```

This last command stitches the images with having same size. We are doing the whole process for every 10 images and make a frame of image.

4.6Tools and Technology

The modification under the target to fulfill the work, wrap inside the useful contribution of tools and technology. Due to inefficient amount of supply in terms of tools and technology, a simple usable, cheaper technique is used. To take a set of images from a degree to another edge a mover required with a webcam. Geometrically the angle of a mover connected to webcam is fixed to angle 180 degree though the mover can rotate to

360. To accommodate maximum number of images into one frame a degree of 180 was sufficient. A high resolution or 1000 ATP webcam is merged with mover creates frame though have fault but still enable to take frame with a little amount of time lag. In the sector of software part on the contrary with image sector various sort of software used. First the video is being taken on the basis of YouCam. Secondly, a Video snapshot wizard is used to extract images from video. Using a burst of images from a video in a degree rotation build up a frame in which 10 images need to be stitch. To stitch these imagesresearch on various software were conducted and later we usedcode in MatLab givesbest result accordance to that.

CHAPTER 5RESULTS AND DISCUSSION

The platform begins with a collection of raw data from a video stream of a room from one end to another. The experiment set up using a 1000 ATP or high resolution image webcam along with a mover to take up a set of 10 images from a single video stream. Likewise 215 frames were created from 2150 images. After the stages of image processing, the frames that are allocated one by one are inserted into the KOAD algorithm to testify the wavelet transformation. The research was to detect 100% anomalies with a very low false alarm. Further comparison with online PCA is conducted as the computation when using PCA is independent to time and hence a precise result has been observed.

5.1 Data and Results

The setup is arranged in BRAC University's thesis lab where we took images of 10 for a single frame. The video was shot in a room of 12 by 18 which was in AVI format. It is not possible to extract features using Haar wavelet from a video clip thus each image are deduced in the JPG format from the video at an interval of 1 -2 sec. For feature extraction the decomposition of Haar wavelet used under 128 coefficients. To obtain a single time step stitching of first 10 images is essential. Similarly 215 time step are obtained from 2150 images and from which 22 anomalies are set by us.

1st position

2nd position

3rd position

4th position

5th position 6th position

7th position8th position

9th position10th position

Figure 8. Different position images are shown covering one end of the room to the other end.

Figure 9. Stitch image 1 (10 image in one frame) at t=31 time steps.

Figure 10.Stitch image 2 (10 image in one frame with anomaly) at t=113 time steps.

Figure 8 showsthe images from the 1st to the 10th position in the span of range 180 degree that covers one end of the room and the other end. The other corresponding figure 9 shows the stitch of these images which is considers as a single time step.

In this figure 10 we have potentially set anomaly marked in red which is used to testify if the proposed algorithm could identify the manually set up anomalies.

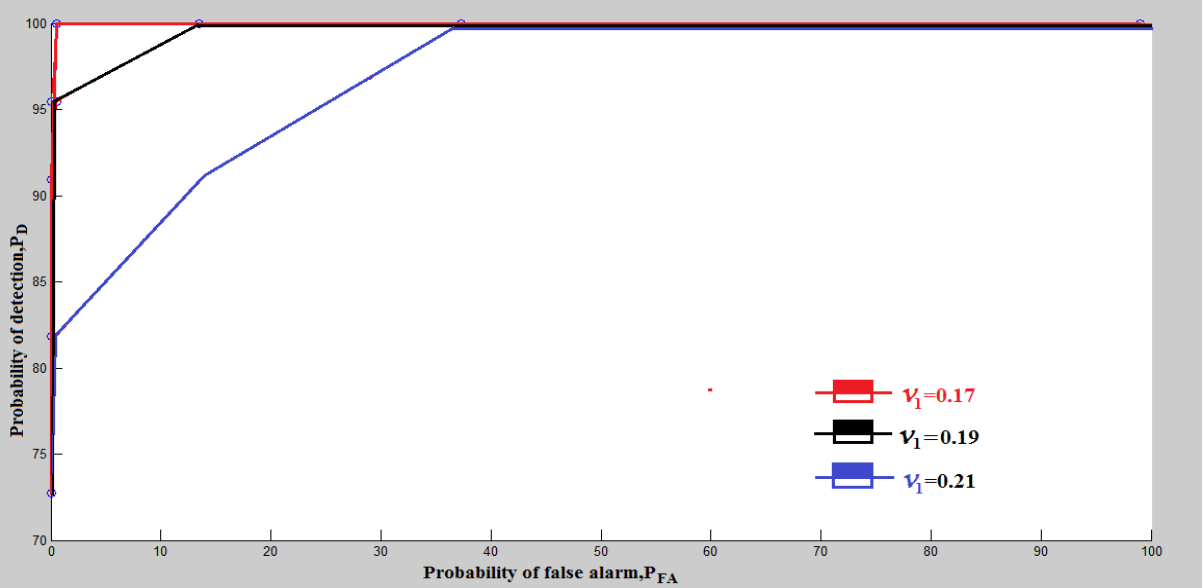

Figure 11.ROC Curve showing KOAD performance

The above figure evaluates the correlation between the probability of detection (P_D) and the probability of false alarm (P_{FA}) asa representation of Receiver operating characteristics (ROC) curve. Basically each ROC curves shows the KOAD performance under different values of threshold (v_1). The curves are sectored up into three different values of threshold and each point depicts the particular choice of V2 for the given choice of V1.

We have chosen Gaussian Kernel functions because we need to cluster the points of an image data in a feature space. So that we can achieve 100% detection at a very low false rate alarm by using the value of v1. The set of values of v1 is range in particular values of 0.17 0.19 and 0.21 where all the three values get 100 % detection with certain portion of false alarm. Under which the detection had done by the threshold value of 0.17 gives a 100% detection with approximately 1.0% probability of false alarm.

5.2 Face Detection

After detecting anomaly in a data set, we have further worked on face detection. Over the course of the recent years recognition of face has grab a lot of attention in the field of engineering. There are widely used potential application in computer vision communication in the topic of face detection. Face detection deal out with factors of pose variation, image orientation and expression. In this project when all the anomalies are found then appearance of any face is being detected by this algorithm. An anomaly image is converted to RGB which further transform into grey scale. The format of double function into the conversion image is used to achieve a full scan image.

Figure 12a: face detection

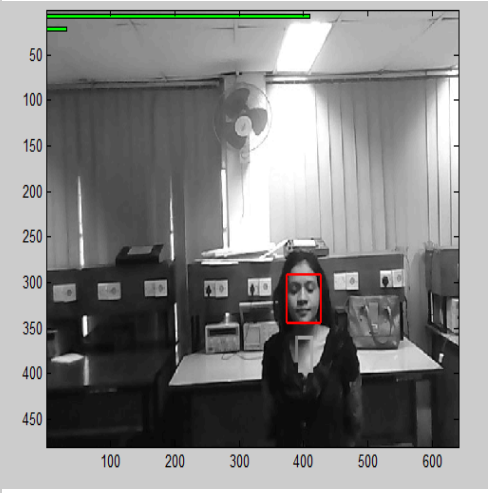

Figure 12b: face detection

5.3Comparison between PCA and KOAD

By default KOAD dropping parameter such as L and d and parameter *l* and *epsilon* are selected to determine the orange alarm resolution. In the figure 14 below shows the experimental comparison study between KOAD detection statistics $\delta_t$and the magnitude of PCA residual component with the corresponding number of time steps. Here, with thresholds v1=0.2 and v2=0.5 are set by us to run KOAD algorithm. The settings strictly depend on a fixed value of v1 whereas the values of v2 varied along the spectral values of delta store. As the threshold line of v2 changes it computes the false detection rate. On the other hand PCA is composed with residual principal components placed in the subspace of normality. In comparison with effectiveness KOAD takes away with more advantage of being learning adaptive and can dealt smoothly with bounded complexity. Online KOAD indicates not only the anomalies that are detected by PCA but also the once that PCA missed. All in all, the output in KOAD is better than PCA in terms of ROC curve and the anomalies detection rate.

In comparison with the previous studies our research has reached to an accuracy of 100% with a probability of false alarm of 1% which proves to be efficient in subject of being a learning algorithm.

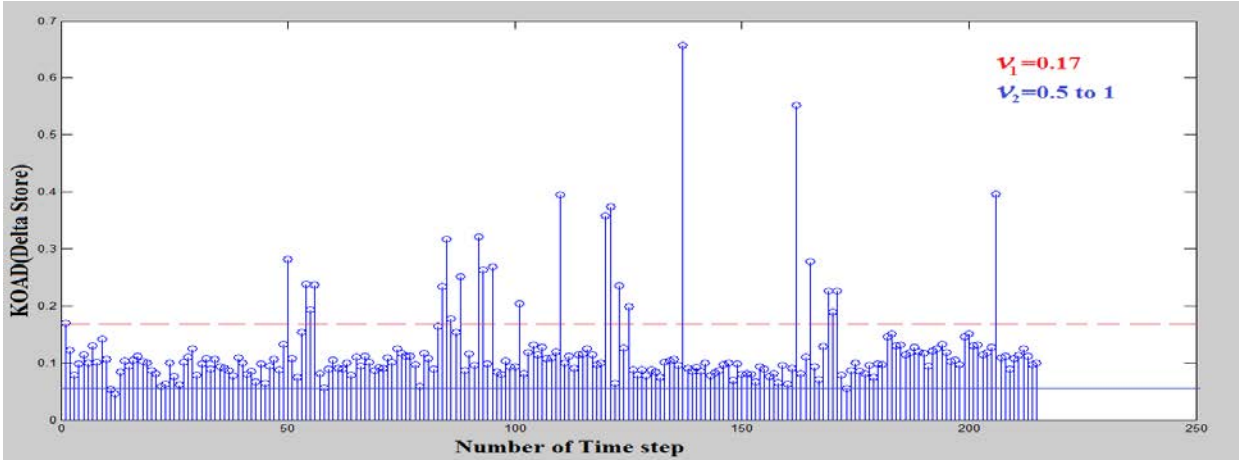

Figure 13. Progressionof the KOAD detection static δ_t

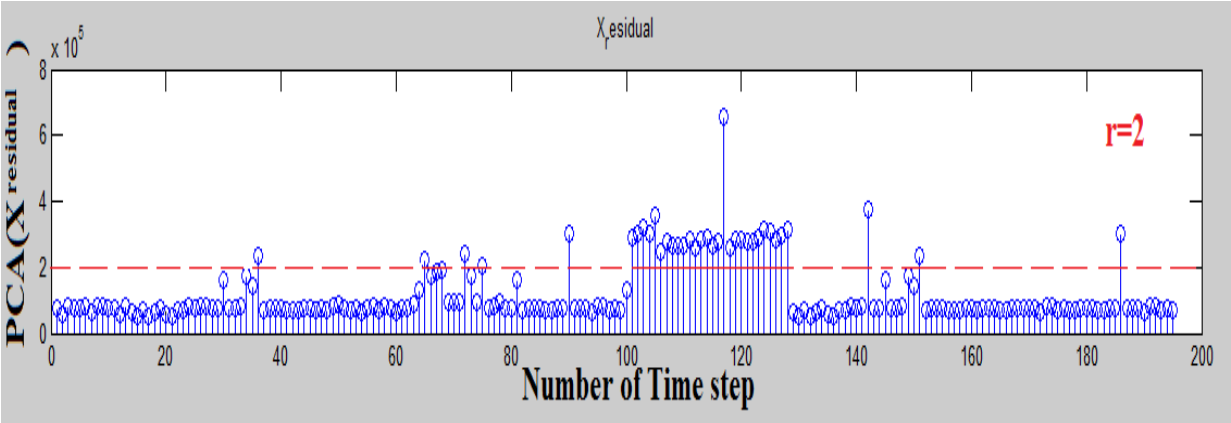

Figure 14. Magnitude of the PCA residual X_r

5.4Project Challenges

The main adjustment of KOAD rely on the fact that it is recursive in nature and have a sense of adaptation even when there is a variation of data stream. The situation depends in the term of appropriate parameter settings and sensitivity toward threshold choices. The architecture covered up in this work is in a platform of sub optimal system. On the contrary, when the system undergoes with the additional face detection algorithm along with it then a huge amount of time required to give an output. Our current and future research focuses on the challenge of rectifying these outcomes and exploring learning based alternatives on the algorithm.

5.5Conclusion and future work

In this thesis the target is to perform to detect anomaly, keeping in mind the component used will be economically stable. The KOAD is based on the images taken at a different position from a moving camera. In the initiative of Kernel algorithm, the KOAD works on cluster points and mapped by a kernel function chosen to an approximate value depending on performance. The algorithm learns is recursive and signal intruder reached into complexity is independent of time. The actual runtime for a single time step of 10 images in a platform required twenty second. The learning algorithm is adaptive toward any changes in normal environment where it has been shown that 100% detection with low false alarm rate activated. To learn the detection outcomes more specifically KOAD is compared with Principle Component Analysis (PCA). The characteristics of KOAD cluster around the feature space while the PCA cluster around the original input space.

Additionally in account of that, a face detection algorithm is implemented up to an approximate extent where a detection of anomaly features read and marked by red lines.

The future work will focus centrally on summing up KOAD extended with the generated face detection algorithm automatically. The main focus to the future work is to create a database where for instance few face already given. After matching the algorithm along with face detection the algorithm can help to recognize the content which was given and the rest faces which are unable to read detected as an anomaly.

CHAPTER – 07 APPENDICES

```
Cam=videoinput (‘winvideo’,1, ‘YUY2_640*480’);
        Cam.ReturnedColorSpace= ‘RGB’;%Return in RGB format
        Cam.TriggerRepeat=Inf;%triggers the camera
        CamFrameGrabInterval=2;
        Cam.on= ‘false’;
Try
    Start(Cam);%start capturing the video
```

Image Stitch

```
jpgFiles = dir('E:\Thesis\collection of images\*.jpg');
k=0;
for im=1:10:length(jpgFiles)
for z=1:10
 if mod(im,10)==1;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
a = imread(jpgFileName);
elseif mod(im,10)==2;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
b= imread(jpgFileName);
elseif mod(im,10)==3;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
c= imread(jpgFileName);
```

```
 elseif mod(im,10)==4;
  jpgFileName = strcat('image- (', num2str(im), ').jpg');
c= imread(jpgFileName);
 elseif mod(im,10)==4;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
d= imread(jpgFileName);
 elseif mod(im,10)==5;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
e= imread(jpgFileName);
  elseif mod(im,10)==6;
   jpgFileName = strcat('image- (', num2str(im), ').jpg');
f= imread(jpgFileName);
 elseif mod(im,10)==7;
  jpgFileName = strcat('image- (', num2str(im), ').jpg');
                g= imread(jpgFileName);
elseif mod(im,10)==8;
 jpgFileName = strcat('image- (', num2str(im), ').jpg');
h= imread(jpgFileName);
elseif mod(im,10)==9;
  jpgFileName = strcat('image- (', num2str(im), ').jpg');
i= imread(jpgFileName);
elseif mod(im,10)==0;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
```

```
j= imread(jpgFileName);

end

 im=im+1;

end

k=k+1;


sa= size(a); %get the size of the left image
sb = size(b);%get the size of the left image

sc= size(c); %get the size of the left image
sd = size(d);%get the size of the left image

 se= size(e); %get the size of the left image
sf = size(f);%get the size of the left image

 sg= size(g); %get the size of the left image
sh = size(h);%get the size of the left image

si= size(i); %get the size of the left image
sj = size(j);%get the size of the left image

b= imresize(b,[sa(1) sa(2)]); %now resize 'b' as per the size of 'a' in order to get perfect sized image
c=imresize(c,[sb(1) sb(2)]);%now resize 'c' as per the size of 'b' in order to get perfect sized image
d= imresize(d,[sc(1) sc(2)]); %now resize 'd' as per the size of 'c' in order to get perfect sized image
e=imresize(e,[sd(1) sd(2)]);%now resize 'e' as per the size of 'd' in order to get perfect sized image
f= imresize(f,[se(1) se(2)]); %now resize 'f' as per the size of 'e' in order to get perfect sized image
g=imresize(g,[sf(1) sf(2)]);%now resize 'g' as per the size of 'f' in order to get perfect sized image
h= imresize(h,[sg(1) sg(2)]); %now resize 'h' as per the size of 'g' in order to get perfect sized image
```

```
i=imresize(i,[sh(1) sh(2)]);
 j=imresize(j,[si(1) si(2)]);
x1= [b a];
x2= [d c];
x3= [f e];
x4= [h g];
x5= [j i];
y1=[x5 x4];
y2=[x3 x2];
y3=[y1 y2];
y=[y3 x1];
imshow(y) % show the image
```

Image Expansion

```
a= double(y);
a1=dwt2(a,'haar');
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
% X=dlmread('X.txt');
[T f] = size(X);
trueint=zeros(1,T);
act=[50 54  55  56  85  86  88 92  93 95 101 110 120 123 125 137 162 165  169 170 171 206];
trueint(act)=1;
k=1;l=1;
```

```
%KRLS online anomaly detection algorithm.
%Requires the following inputs:
% Mandatory: data matrix (X), lower threshold (nu1), upper threshold (nu2).
% Parameters for dropping obsolete elements; default is d = 0.9, L = 100.
% Parameters for resolving orange alarm; default is epsilon = 0.2, el = 20.
% Parameters for resetting P; default is R = 10000, r = 1.
% Forgetting factor; default is gamma = 1;
%Yields the following outputs:
% Always: Red1 and Red2 alarm positions.
% If desired: records of delta and prediction error.
%function [Red1_out, Red2_out deltaStore_out Error_out] = KRLS(X, nu1, nu2, d, L, epsilon, el, R, r, gamma)
%if nargin < 11 = 1; end %Needed only if using Gaussian kernel
gamma = 1; %Forgetting factor
r = 1;  %Parameters for resetting P
R = 10000;
el = 10;  %Parameters for resolving orange alarm
epsilon = 0.2;
L = 100;  %Parameters for dropping obsolete elements
d = 0.9;
kernelChoice = 2;
sigma = 0.09;
Y = sum(X,2); %Add after normalizing
[T f] = size(X);
sumdec=0;


 nu1=0.17; %nu1=0.25:0.02:0.29;
 for nu2=0.05:0.1:1
sumdec=sumdec+1;
flagint=zeros(1,T);
 %  reply = input('Do you want more? Y/N [Y]: ','s');
% if isempty(reply)
```

```
%     reply = 'Y';
%     close all
  %  nu1 = 0.17 ; nu2 = 0.15; %Threshholds
%d = 0.9; L = 100; %Parameters for dropping obsolete elements
%epsilon = 0.20; el = 20; %Parameters for resolving orange alarm
%R = 10000; r = 1; %Parameters for resetting P
%gamma = 1; %Forgetting factor
%sigma = 1; %Needed only if using Gaussian kernel
Red1 = []; Red2 = [];%Clear alarms
Orange = []; x_Orange = []; %Store x in timesteps when Orange alarm is raised
   % Initialize %
t = 1;
x = X(t,:)';
y = Y(t);
k11 = kernel(x, x,kernelChoice, sigma);
K_tilde = [k11];
K_tilde_inv = [1/k11];
    Dictionary = [x];
index_m = [t]; %Keeps track of timesteps when elements are added (+2), deleted (-1) or no change to D (0); for debugging only
Orange = [Orange t];
x_Orange = [x_Orange x];
drop_index = [0];
P=[1]; %P=inv(A'A)
m=1;
m_t(t) = m; %Keep track of m, for debugging only
index_m(t) = 2; %index_m(t)=2 implies x(t) is being added to Dictionary
alpha = y(t)/k11;
deltaStore(1) = nu1+eps;  %For debugging\
% % Evaluate y_hat %
y_hat = zeros(T,1);
```

```
Error = zeros(1,T);
for j=1:m
   y_hat(t) = y_hat(t) + alpha(j)*kernel(Dictionary(:,j),x,kernelChoice, sigma);
end %for j=1:m
Error(t) = (Y(t)-y_hat(t))/Y(t)*100;
%clear Lambda lambda dotProd;
Lambda = kernel(Dictionary(:,j),x,kernelChoice, sigma);
%Keep track of all dot product (kernel) values; for debugging only
for j=1:m
dotProd(t,j) = kernel(Dictionary(:,j),x,kernelChoice, sigma);
end
for t=2:T
   x = X(t,:)';
   y = Y(t);
% Evaluate current measurement %
   k_tilde = zeros(m,1);
for j=1:m
      k_tilde(j) = kernel(Dictionary(:,j),x,kernelChoice, sigma); %Computing k_tilde{t-1}
 end %for j=1:m
   a = K_tilde_inv*k_tilde;
   delta = kernel(x,x,kernelChoice, sigma) - k_tilde'*a;
%      deltaCheck = a'*K_tilde*a - 2*a'*k_tilde + kernel(x,x); %Verify delta; not part of
algorithm
   deltaStore(t) = delta;  %Keep track of delta, for debugging only
   if t>L
      Lambda = [Lambda(2:end,1:end) ; ceil(k_tilde'-repmat(d,1,m))]; %Append with 1 or 0
   else
      Lambda = [Lambda ; ceil(k_tilde'-repmat(d,1,m))]; %Append with 1 or 0
   end %if t>L
   if (delta>=nu1 & delta<nu2) %Orange alarm, add to Dictionary
      x_Orange = [x_Orange x];
```

```
        Orange = [Orange t];
        Dictionary = [Dictionary x];
    drop_index = [drop_index 0];
        a_tilde = a;
        K_tilde_inv = [ (delta*K_tilde_inv+a_tilde*a_tilde') (-1*a_tilde) ; (-1*a_tilde') (1) ] /
delta;
        K_tilde = [ K_tilde k_tilde ; k_tilde' kernel(x,x,kernelChoice, sigma) ];
        if t>L
            lambda = [zeros(L-1,1) ; 1];
        else
            lambda = [zeros(t-1,1) ; 1];
        end %if t>L
        Lambda = [Lambda lambda];
a = [zeros(m-1,1) ; 1];
        P = [ P zeros(m,1) ; zeros(m,1)' gamma ]/gamma;
        alpha = [ (gamma^(-0.5)*alpha - a_tilde*(y-gamma^(-0.5)*k_tilde'*alpha)/delta) ; ((y-
gamma^(-0.5)*k_tilde'*alpha)/delta) ];
        m=m+1;
        m_t(t) = m;
        index_m(t) = 2; %Element added to D in this timestep
    else %delta<nu1 or delta>=nu2, Dictionary unchanged
        if delta>nu2 %Red1 alarm
            Red1 = [Red1 t];
        end %if delta>nu2;
        P = (1/gamma)*[P - q*a'*P];
alpha = alpha + K_tilde_inv*q*(y-k_tilde'*alpha);
%Keep track of all dot product (kernel) values; for debugging only
    for j=1:m
        dotProd(t,j) = kernel(Dictionary(:,j),x,kernelChoice, sigma);
    end
 % Process previous orange alarm %
```

```
if t>el & sum(Orange==t-el)==1 %means orange alarm at timestep t-el
      %Identify Dictionary element j corr. to the orange alarm at timestep t-el
      for j=1:m
         if x_Orange(:,Orange==t-el)==Dictionary(:,j)
            break;
         end %if x_Orange(:,Orange==t-el)==Dictionary(:,j)
      end %for j=1:m
if sum(Lambda(end-el+1:end,j)) <= epsilon*el
         %Orange turns Red
         Red2 = [Red2  Orange(Orange==t-el)]; %Red2 alarm
         x_Orange(:,Orange==t-el) = [];
         Orange(Orange==t-el) = [];
         drop_index = [zeros(1,j-1) 1 zeros(1,m-j) ];
      else
         %Orange turns green
         x_Orange(:,Orange==t-el) = [];
         Orange(Orange==t-el) = [];
      end %if size(find(Lambda(end-el+1:end,j)<d),1) >= 0.80*25
end %if t>el & sum(Orange==t-el)==1
  % Remove obsolete elements %
   for j=1:m
      %Dropping condition: kernel exists for past L timesteps, and is always < d
      if  ( t>L & sum(Lambda(1:end,j))==0 )
         drop_index(j) = 1;
      end %if  ( t>L & gt(Lambda(:,j),0) & lt(Lambda(:,j),d) )
   end %for j=1:m
   % DropElement(p) %
   if ( find(drop_index==1) & m>1 & t>r ) t;
p = min(find(drop_index==1)); %Drop Dictionary element # p
      %Reorganize K_tilde_p and K_tilde_inv_p, with p'th row/col moved to the end
K_tilde = [ K_tilde(1:p-1,1:p-1) K_tilde(1:p-1,p+1:m) K_tilde(1:p-1,p)  ;
```

```
K_tilde(p+1:m,1:p-1) K_tilde(p+1:m,p+1:m) K_tilde(p+1:m,p)  ;  K_tilde(p,1:p-1)

K_tilde(p,p+1:m) K_tilde(p,p) ];
K_tilde_inv = [ K_tilde_inv(1:p-1,1:p-1) K_tilde_inv(1:p-1,p+1:m) K_tilde_inv(1:p-1,p);
K_tilde_inv(p+1:m,1:p-1) K_tilde_inv(p+1:m,p+1:m) K_tilde_inv(p+1:m,p)  ;
K_tilde_inv(p,1:p-1) K_tilde_inv(p,p+1:m) K_tilde_inv(p,p) ];
delta_p = 1/(K_tilde_inv(m,m));
 a_tilde_p = -delta_p*[K_tilde_inv(1:m-1,m)];
K_tilde_inv = K_tilde_inv(1:m-1,1:m-1)-a_tilde_p*a_tilde_p'/delta_p;
alpha = alpha - (1/delta_p)*[a_tilde_p*a_tilde_p'  -a_tilde_p  ;  -a_tilde_p'  1] *K_tilde*alpha;
alpha = alpha(1:m-1);
K_tilde = K_tilde(1:m-1,1:m-1);
     Dictionary(:,p) = [];
end
Red1_out = Red1; Red2_out = Red2;
Red1_out = Red1; Red2_out = Red2; deltaStore_out = deltaStore;
  Red1_out = Red1; Red2_out = Red2; deltaStore_out = deltaStore; Error_out = Error;

Red1
Red2
flagint(Red1)=1;
flagint(Red2)=1;
flagint(1)=0;
flagint;
detected=bitand(flagint,trueint);
false=bitxor(flagint,trueint);
falsem=false;
false(act)=0;
missed=bitxor(falsem,false);
mis(sumdec)=sum(missed);
dec(sumdec)=(sum(detected)/22)*100;
```

```
fal(sumdec)=(sum(false)/(215-22))*100;
% if valu==1

scatter(sort(fal),sort(dec));
hold on
plot(fal,dec,'r');
%     end
 end
%   clear Red1 Red2 flagint detected false falsem missed

figure(215+1)
stem(deltaStore_out)
dotProd;
% end
% if valu==1
scatter(sort(fal),sort(dec));
hold on
plot(fal,dec,'r');
%     end
 end
%   clear Red1 Red2 flagint detected false falsem missed
figure(215+1)
stem(deltaStore_out)
dotProd;
% end
```

Anomaly Image

```
Red2;
for q=1:length(Red2)
cal=10*Red2(q);
c=cal-9;
for ir=c:cal
if mod(ir,10)==1;
      jpgFileName = strcat('image- (', num2str(ir), ').jpg');
```

```
a = imread(jpgFileName);
            elseif mod(ir,10)==2;
jpgFileName = strcat('image- (', num2str(ir), ').jpg');
b= imread(jpgFileName);
            elseif mod(ir,10)==3;
jpgFileName = strcat('image- (', num2str(ir), ').jpg');
c= imread(jpgFileName);
elseif mod(im,10)==4;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
d= imread(jpgFileName);
           elseif mod(im,10)==5;
 jpgFileName = strcat('image- (', num2str(im), ').jpg');
e= imread(jpgFileName);
elseif mod(im,10)==6;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
f= imread(jpgFileName);
elseif mod(im,10)==7;
 jpgFileName = strcat('image- (', num2str(im), ').jpg');
g= imread(jpgFileName);
           elseif mod(im,10)==8;
 jpgFileName = strcat('image- (', num2str(im), ').jpg');
h= imread(jpgFileName);
elseif mod(im,10)==9;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
i= imread(jpgFileName);
elseif mod(im,10)==0;
jpgFileName = strcat('image- (', num2str(im), ').jpg');
j= imread(jpgFileName);
 end
    ir=ir+1;
 end
sa= size(a); %get the size of the left image
```

```
sb = size(b);%get the size of the left image
sc= size(c); %get the size of the left image
 sd = size(d);%get the size of the left image
 se= size(e); %get the size of the left image
 sf = size(f);%get the size of the left image
sg= size(g); %get the size of the left image
 sh = size(h);%get the size of the left image
 si= size(i); %get the size of the left image
 sj = size(j);%get the size of the left image
b= imresize(b,[sa(1) sa(2)]); %now resize 'b' as per the size of 'a' in order to get perfect sized image
 c=imresize(c,[sb(1) sb(2)]);%now resize 'b' as per the size of 'a' in order to get perfect sized image
d= imresize(d,[sc(1) sc(2)]); %now resize 'b' as per the size of 'a' in order to get perfect sized image
 e=imresize(e,[sd(1) sd(2)]);%now resize 'b' as per the size of 'a' in order to get perfect sized image
  f= imresize(f,[se(1) se(2)]); %now resize 'b' as per the size of 'a' in order to get perfect sized image
g=imresize(g,[sf(1) sf(2)]);%now resize 'b' as per the size of 'a' in order to get perfect sized image
h= imresize(h,[sg(1) sg(2)]); %now resize 'b' as per the size of 'a' in order to get perfect sized image
i=imresize(i,[sh(1) sh(2)]);
 j=imresize(j,[si(1) si(2)]);
x1= [b a];
x2= [d c];
x3= [f e];
x4= [h g];
x5= [j i];
% club the image a & b into another new image after making their size same
```

```
%note the missing semicolon in the above line inside bracket
y1=[x5 x4];
y2=[x3 x2];
y3=[y1 y2];
y=[y3 x1];
   figure,imshow(y);
end
```

Face Detection

```
Red2;
for q=1:length(Red2)
cal=10*Red2(q);
c=cal-9;
for i=c:cal
  jpgFileName = strcat('image (', num2str(i), ').jpg');
  x = imread(jpgFileName);
%
  if (size(x,3)>1)%if RGB image make gray scale
    try
       x=rgb2gray(x);%image toolbox dependent
    catch
       x=sum(double(x),3)/3;%if no image toolbox do simple sum
    end
end
x=double(x);%make sure the input is double format
[output,count,m,svec]=facefind(x);%full scan
figure,imagesc(x), colormap(gray)%show image
plotbox(output,[],8)%plot the detections as red squares
plotsize(x,m)
  figure,imshow(x);
end
```

CHAPTER- 08 BIBLOGRAPHY

[1] M. Valera and S. Velastin, “Intelligent distributed surveillance systems: a review,” IEE Proc.- Vis., Image and Signal Process., vol. 152, pp. 192–204, Apr. 2005.

[2] T. Ahmed, “Online anomaly detection using KDE,” in Proc. IEEE Global Communications Conf. (GLOBECOM), Honolulu, HI, USA, Nov. 2009.

[3] 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.

[4] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least squares algorithm,” IEEE Trans. Signal Proc., vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[5] K. Sudo, T. Osawa, X. Wu, K. Wakabayashi, and T. Yasuno, “Detecting the degree of anomal in security video,” in Proc. IAPR Conf. on Machine Vision Applications, Tokyo, Japan, May 2007.

[6] T. Poh, N. Lani, and L. Kin, “Multi-dimensional features reduction of PCA on SVM classifier for imaging surveillance application,” Int. J. of Systems Applications, Engineering and Development, vol. 1, pp. 45–50, Jan. 2007.

[7] Michael D. Breitenstein, Helmut Grabner, Luc Van Gool, "Hunting Nessie – Real-Time Abnormality Detection from Webcams," IEEE Int. Workshop on Visual Surveillance, October 2009.

[8] R. Schuster, R. Mörzinger, W. Haas, H. Grabner, and L. Gool. "Real-time Detection of Unusual Regions in Image Streams," in Proc. ACM International Conference on Multimedia - Video Program 2010.

[9] L. Gool, M. Breitenstein,S. Gammeter,H. Grabner,T. Quack, "Mining from large image sets," in Proc. ACM International Conference on Image and Video Retrieval Article No. 10, 2009.

[10] L.Kratz, K. Nishino, "Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models," In Proc. IEEE CVPR, Miami, FL, August 2009.

[11] W. Chao and X.M. Jun, "Multi-agent Based Distributed Video Surveillance System over IP", ;in Proc. ISCSCT (2), 2008, pp.97-100.

[12] J. Jackson and G. Mudholkar, “Control procedures for residuals associated with principal component analysis,” Technometrics, vol. 21, no. 3, pp. 341–349, Aug. 1979

[13] B. Sch¨olkopf and A. Smola, Learning with Kernels. Cambridge, MA: MIT Press, Dec. 2001.

[14] T. Ahmed, M. Coates, and A. Lakhina, “Multivariate online anomaly detection using kernel recursive least squares,” in Proc. IEEE INFOCOM, Anchorage, AK, May 2007.

[15] Park sudhaker, “Discreate Wavelet Transform”, Nov. 2003.