Abnormal Behavior Detection of Human by Video Surveillance System

Anisul Amin

ID: 10301001

Mohammad Farhan Anzum

ID: 12201101

Mark Himel Mondol

ID: 14141010

Supervised by

Md. Zahangir Alom

Co-Supervised by

Samiul Islam

Department of Computer Science & Engineering

December 2014

BRAC University, Dhaka, Bangladesh

Abstract

In recent years, the number of surveillance cameras installed to monitor private and public spaces and areas has increased dramatically. There is an increasing demand for smarter video surveillance of public and private space using intelligent vision systems which can distinguish what is semantically meaningful to the human observer as ‘normal’ and ‘abnormal’ behaviors. Usually, the video streams are constantly recorded or observed by operators. In these cases an intelligent system can give more accurate performance than a human.

In this thesis we present a video surveillance system that detects and predicts abnormal behavior of human. The system acquires color images from a stationary camera and analyzes the behavior of human. Behaviors that are common or frequent will not be given much attention by the system.

Acknowledgement

We would like to thank Md. Zahangir Alom for agreeing to supervise us with our thesis. His patience and confidence in us has been a source of encouragement and this thesis would not have been possible without the great support, inspiration and influence of him. We believe his dedication to this paper deserves to be reciprocated with great gratitude.

We would like to thank Samiul Islam for supporting us as co-supervisor in our thesis.

A special thanks to the Thesis committee for taking the time to review and evaluate our thesis as part of our undergraduate program.

Supervised By

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

Nowadays with the increasingly growing needs of protection of people and personal properties, video surveillance has become a big concern of everyday life. A consequence of these needs has led to the deployment of cameras almost everywhere. At present, using video in machine understanding is a significant research topic. One of the most active areas is activity understanding from video surveillance system. Understanding activities involves being able to detect and classify targets of interest and analyze what they are doing. One crucial aspect is to detect and report situations of special interests, in particular when unexpected things happen. In this case, a video surveillance system which can interpret the scene and automatically recognize abnormal behaviors can play a vital role. The system would then notify operators or users accordingly.

In thesis, the goal is to develop methods and techniques that enable to approach such an ideal surveillance system. A large part of this project is devoted to visual analysis of human behavior and detection of abnormal gestures of human in indoor scenarios. In this context, a working system would alert of dangerous situations and improve the personal safety of people living on their own.

1.2 Related Works

We mainly focus on the work related to abnormality detection in video. Several research works in the area of video understanding have been carried out in the last decade. Many issues as behavior understanding in crowded scenes, indoors, object detection are combined together in order to perform behavior recognition, so we review the representative papers and research works on these different topics.

A comprehensive overview over the general topics of object recognition, tracking, and action recognition is beyond the scope of this paper and can be found in Ommer *et al*. [1]. The main paradigm for abnormality detection in videos is to extract semi-local features and to learn a model on the normal samples from the training data. Abnormality is estimated by measuring how bad the model fits. The degree of supervision in these models varies to a great extent. Some of the approaches [2] are based on a set of constraints that are introduced to specify the normality, whereas the methods [4] are unsupervised approaches which directly determine normal patterns. The approach by Adam *et al*. [9] can be deemed local since the attention is directed to individual activities occurring in a local area. While this approach provides good results when it comes to implementation and efficiency, its performance suffers from the incapability of the model to incorporate temporal aspects of relationships among activities. Xiang and Gong [5] proposed a method that automatically recognizes behavior and detects abnormalities without applying any manual labeling. Zhong *et al*. [4] detect objects by thresholding a motion filter and they propose an unsupervised method that integrates the prototypical image features and classifies a group of behavior patterns either as normal or abnormal. Kim and Grauman [8] proposed a method to detect abnormalities in a video sequence based on a space-time Markov random field model. This model dynamically adapts to abnormal activities that consists of unpredictable variations. Some of the current methods for the detection of abnormal behavioral patterns are based on unsupervised one-class learning approaches. These namely include topic models such as in [7] or Markov random field models [8]. Other methods utilize supervised approaches for classifying events and patterns such as [6]. Mahadevan *et al*. [3] presents method for unusual behavioral pattern detection in crowded scenes that is based on mixtures of dynamic textures. Their approach also includes jointly performed modeling of the dynamics and appearance of a scene as well as detection of temporal abnormalities which are represented as low-probability occurrences and unusual spatial activities which are dealt with using the discriminative saliency property. The provided dataset contains low-resolution video sequences of crowds with occlusion.

1.3 Contributions

The contributions of us in this thesis are summarized as follows:

- We worked with each frame in a video. We took a test video as input and took frame by frame to move on to the next step.
- We can detect multiple humans and analyze the behaviors of them at a time.
- Our system will not detect any other object or creatures other than human. We gave a minimum threshold level for detecting human.
- In our thesis, we converted the binary values of the binary images into decimal and created a column vector for training purpose.

1.4 Outline

We divided our whole thesis report into 2 chapters. These two chapters describe the two different approaches. The contents of the chapters are mentioned below:

Section 2 describes Literature Review. Section 3 describes technical overview of image morphology. Section 5 describes the experimental results. Section 6 and 7 deals with accuracy calculation and comparison.

1.4.1 Chapter 1

This is followed by system design, section 4, where how we implemented our system for abnormal behavior detection using Viola Jones Algorithm. Next behavior analysis is described.

1.4.2 Chapter 2

This is followed by system design, section 4, where how we implemented our system for abnormal behavior detection using foreground detection Algorithm. Next behavior analysis is described.

Section 8 discusses the Contributions and Future Work of this project.

2. Literature Review

2.1 Abnormal Behavior

In order to provide security, it is necessary to analyze the behaviors of people and determine whether these behaviors are normal or abnormal. Before we mention what is meant by abnormal behavior, we should notice that several keywords were used by many research works [10, 11] to refer to the same notion (unusual, rare, atypical, interesting, suspicious, and anomalous). With Mahadevan *et al* [3], the *abnormalities* are defined as measurements whose probability is below a certain threshold under a normal model. Rare behavior and unusual behavior are not same. There are some differences between them. Rare behavior means that has not been observed before, those that have been seen once are considered as rare but not necessarily abnormal. On the other hand, unusual behaviors can be predicted as abnormal.

2.2 Digital Image Processing

Digital image processing is the use of computer algorithm to perform image processing on digital images. In the field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

According to *Rafael C. Gonzalez* [12], an image may be defined as a two-dimensional function, f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of at any pair of coordinates (x, y) is called the Intensity or gray level of the image at that point. When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a digital image. The field of digital image processing refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements and pixels. Pixel is the term most widely used to denote the elements of a digital image. Vision is the most advanced of our senses, so it is not surprising that images play the single most important role in human perception.

2.2.1 Coordinate Convention

The result of sampling and quantization is a matrix of real numbers. Let us assume that an image f(x, y) is sampled so that the resulting image has M rows and N columns. We say that the image is of *size MN**. The values of the coordinates are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates. In many image processing books, the image origin is defined to be at (x, y) = (0, 0). The next coordinate values along the first row of the image are (x, y)= (0,1). The notation (0, 1) is used to signify the second sample along the first row. It *does not* mean that these are the *actual* values of physical coordinates when the image was sampled. Figure 2.1 shows this coordinate convention. In the figure x ranges from 0 to M-1 and y from 0 to N-1 in integer increments.

The coordinate convention used in the Image Processing Toolbox to denote arrays is different from the preceding paragraph in two minor ways. First, instead of using (x, y), the toolbox uses the notation (r, c)to indicate rows and columns. Note, however, that the order of coordinates is the same as the order discussed in the previous paragraph, in the sense that the first element of a coordinate tuple, (in the figure) refers to a row and the second to a column. The other difference is that the origin of the coordinate system is at (r, c) = (1, 1); thus, r ranges from 1 to M, and c from 1 to N, in integer increments.

The result of sampling and quantization is a matrix of real numbers. We use two principal ways in this book to represent digital images. Assume that an image f(x, y) is sampled so that the resulting image has M rows and N columns. We say that the image is of *size MN**. The values of the coordinates are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates. In many image processing books, the image origin is defined to be at (x, y) = (0, 0). The next coordinate values along the first row of the image are (x, y)= (0, 1). The notation (0, 1)is used to signify the second sample along the first row. It *does not* mean that these are the *actual* values of physical coordinates when the image was sampled.

Figure 2.1 shows this coordinate convention. Note that *x* ranges from 0 to *M*-1 and *y* from 0 to *N*-1 in integer increments.

The coordinate convention used in the Image Processing Toolbox to denote arrays which are different from the preceding paragraph in two minor ways. First, instead of using (x, y), the toolbox uses the notation (r, c)to indicate rows and columns. Note, however, that the order of coordinates is the same as the order discussed in the previous paragraph, in the sense that the first element of a coordinate tuple (a, b), refers to a row and the second to a column. The other difference is that the origin of the coordinate system is at (r, c) = (1, 1); thus, *r* ranges from 1 to *M*, and *c* from 1 to *N*, in integer increments.

Figure 2.1(b) illustrates this coordinate convention. Image Processing Toolbox documentation refers to the coordinates in Fig. 2.1(b) as *pixel coordinates*.

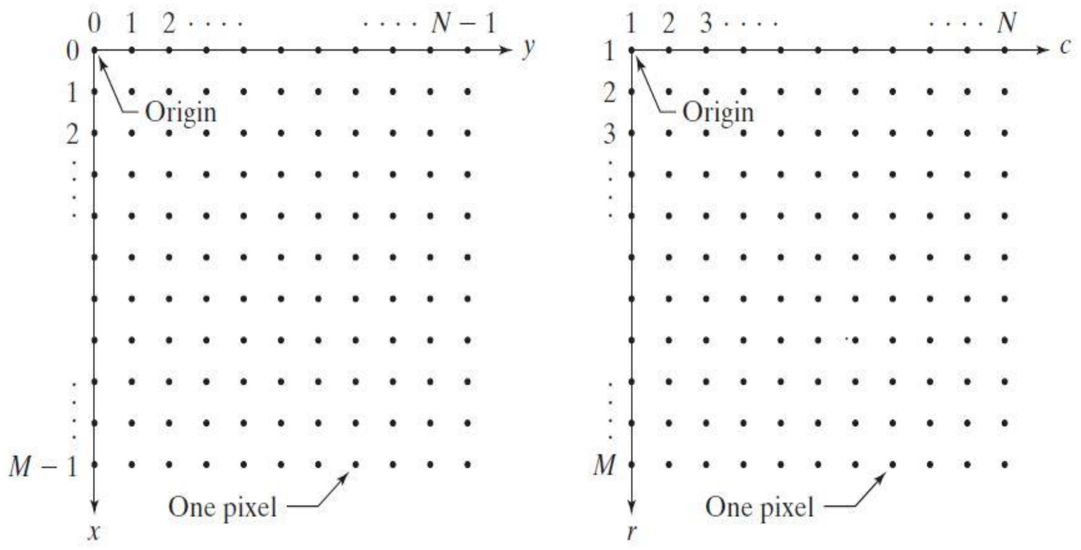


Figure 2.1(a) & 2.1(b)

2.2.2 Matrix form of images

The coordinate system in Fig. 2.1(a) and the preceding discussion lead to the following representation for a digitized image:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

The right side of this equation is a digital image by definition. Each element of this array is called an image element, picture element, and pixel. The terms *image* and *pixel* are used throughout the rest of our discussions to denote a digital image and its elements.
A digital image can be represented as a MATLAB matrix:

$$\mathtt{f} = \begin{bmatrix} \mathtt{f(1,1)} & \mathtt{f(1,2)} & \cdots & \mathtt{f(1,N)} \\ \mathtt{f(2,1)} & \mathtt{f(2,2)} & \cdots & \mathtt{f(2,N)} \\ \vdots & \vdots & & \vdots \\ \mathtt{f(M,1)} & \mathtt{f(M,2)} & \cdots & \mathtt{f(M,N)} \end{bmatrix}$$

Above we used the letters M and N, respectively, to denote the number of rows and columns in a matrix. A 1 X N matrix is called a *row vector*, whereas an M X 1 matrix is called a *column vector*. A 1 X 1 matrix is a *scalar*.

2.3 Image Segmentation

The goal of image segmentation is to simplify the representation of an image into something that is more meaningful and easier to analysis. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics. This section describes some approaches of image segmentation in brief. The morphological image processing which is implemented in the project is described below:

2.3.1 Morphological Image Processing

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. The value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, one can construct a morphological operation that is sensitive to specific shapes in the input image. Morphological operations can also be applied to gray-scale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest.

In this section morphological functions which can be used to perform common image processing tasks, such as contrast enhancement, noise removal, thinning, skeletonization, filling, and segmentation are discussed. Topics those are covered by this include Terminology, Dilation and Erosion, Morphological Reconstruction, Distance Transform, Watershed Segmentation, Objects, Region and Feature Measurement and Lookup Table Operations. Here, for this paper we need to know Dilation & Erosion, Structuring Elements, Dilating an Image, Eroding an Image, Morphological Reconstruction etc. brief introduction of each of these are given below.

2.3.1.1 Dilation and Erosion

Dilation and erosion are two fundamental morphological operations. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image.

The rules or definition of dilation and erosion are shown below by a table:

Rules for Gray scale Dilation and Erosion

Operation	**Rule**
Dilation	The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

Table 1

2.3.1.2 Structuring Elements

The structuring element is sometimes called the *kernel*, but we reserve that term for the similar objects used in convolutions. The structuring element consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. Normally Cartesian coordinates are used and so a convenient way of representing the element is as a small image on a rectangular grid. Figure 2.3.1.2 shows a number of different structuring elements of various sizes. In each case the origin is marked by a ring around that point. The origin does not have to be in the center of the structuring element, but often it is. As suggested by the figure, structuring elements that fit into a 3×3 grid with its origin at the center are the most commonly seen type.

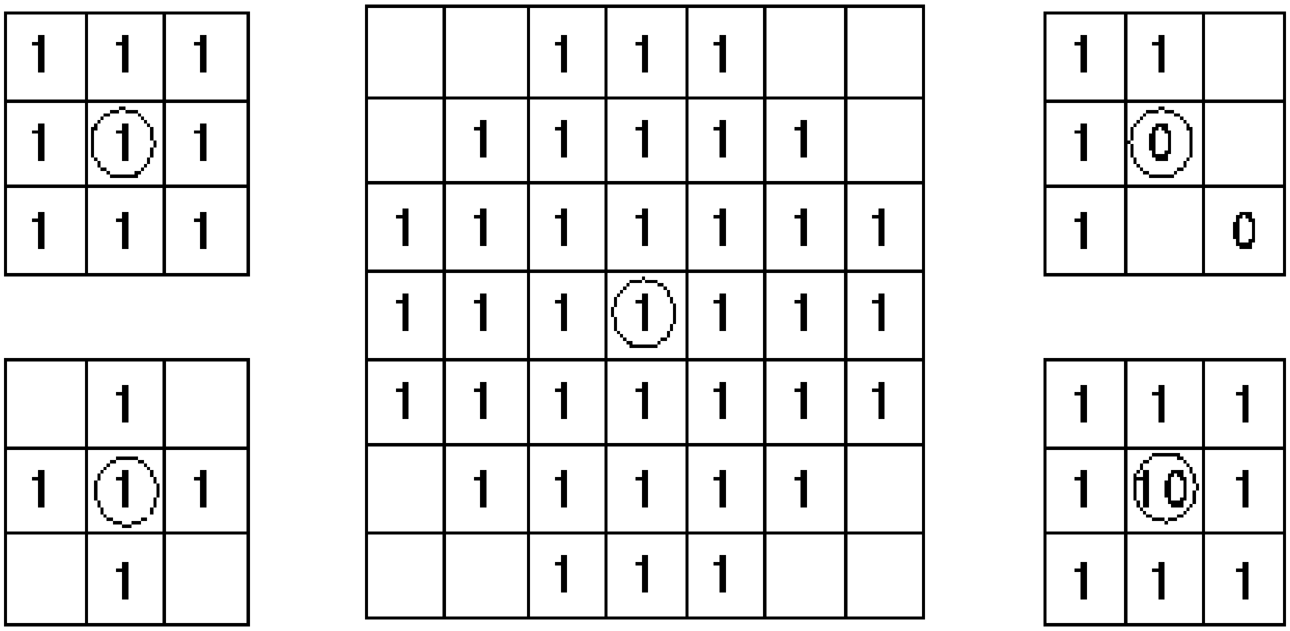


Figure: 2.3.1.2some example of structuring element

Note that each point in the structuring element may have a value. In the simplest structuring elements used with binary images for operations such as erosion, the elements only have one value, conveniently represented as a one. More complicated elements, such as those used with thinning or gray-scale morphological operations may have other pixel values.

2.3.1.3 Opening

The basic effect of an opening is somewhat like erosion in that it tends to remove some of the foreground (bright) pixels from the edges of regions of foreground pixels. However it is less destructive than erosion in general. As with other morphological operators, the exact operation is determined by a structuring element. The effect of the operator is to preserve foreground regions that have a similar shape to this structuring element, or that can completely contain the structuring element, while eliminating all other regions of foreground pixels.

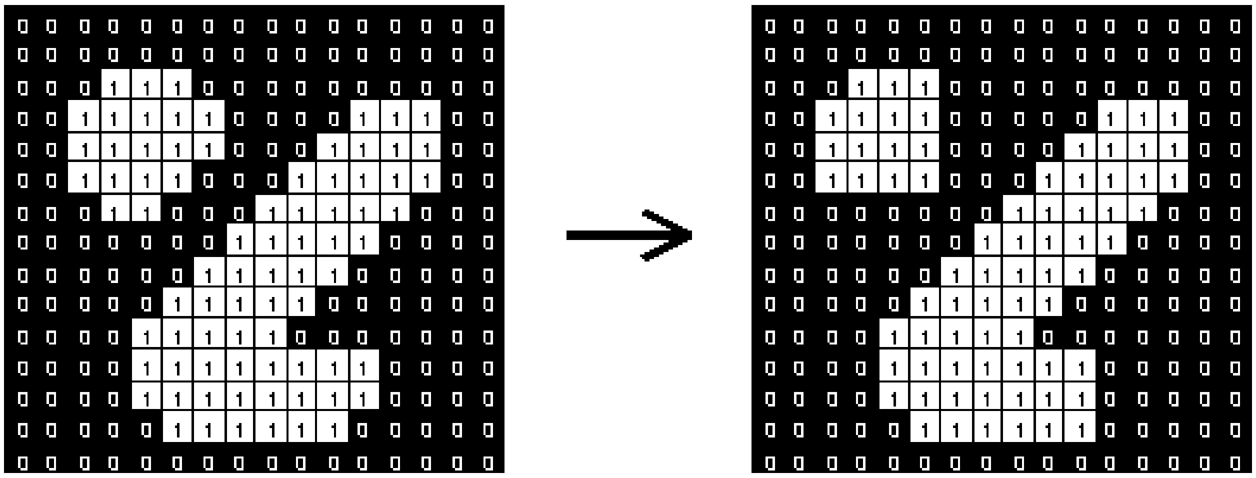


Figure 2.3.1.3: Effect of “Opening” using a 3 X 3 square structuring element

2.3.1.4 Closing

Closing is similar in some ways to dilation in that it tends to enlarge the boundaries of foreground (bright) regions in an image (and shrink background color holes in such regions), but it is less destructive of the original boundary shape. As with other morphological operators, the exact operation is determined by a structuring element. The effect of the operator is to preserve background regions that have a similar shape to this structuring element, or that can completely contain the structuring element, while eliminating all other regions of background pixels.

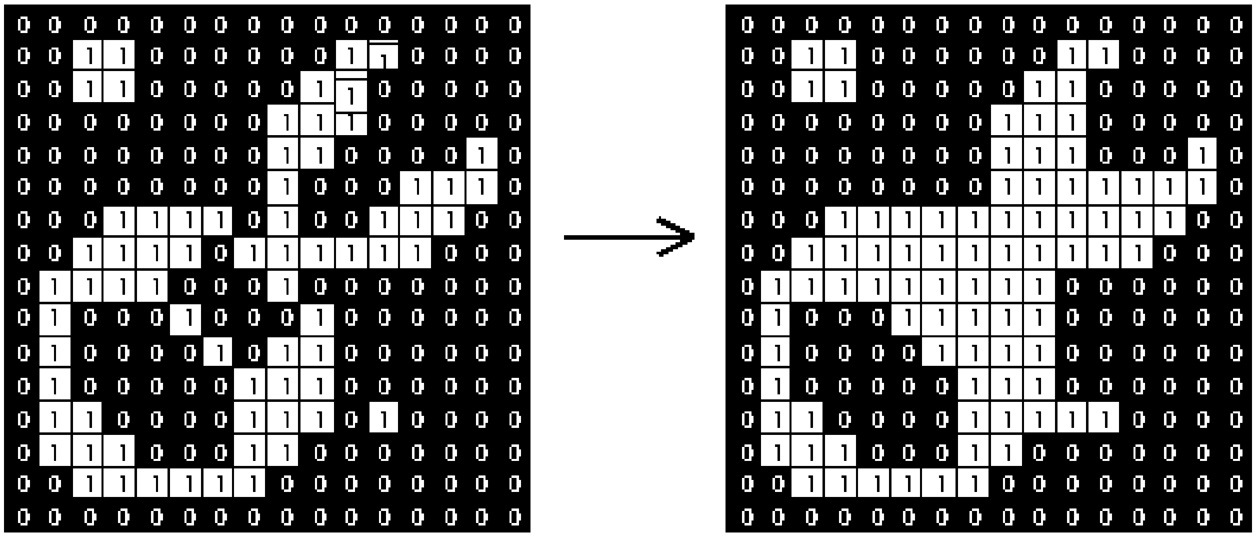


Figure 2.3.1.4: Effect of “Closing” using a 3×3 square structuring element

2.4 Color Conversion

According to Wan Najwa [13], ‘threshold’ is an image segmentation to convert gray-scale to binary image. During the threshold process, individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. This convention is known as threshold above. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled "object" if its value is between two thresholds; and threshold outside, which is the opposite of threshold inside. Typically, an object pixel is given a value of “1” while a background pixel is given a value of “0.” Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label.

3. Technical Review

3.1Overview of Viola Jones Algorithm

The **Viola–Jones object detection framework** is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection.

The basic problem to be solved is to implement an algorithm for detection of faces in an image. This can be solved easily by humans. However there is a stark contrast to how difficult it actually is to make a computer successfully solve this task. In order to ease the task Viola-Jones limits themselves to full view frontal upright faces. That is, in order to be detected the entire face must point towards the camera and it should not be tilted to any side. This may compromise the requirement for being unconstrained a little bit, but considering that the detection algorithm most often will be succeeded by a recognition algorithm these demands seem quite reasonable.

Feature types

The main characteristics of viola-jones algorithm which makes it a good detection algorithm is –

1. Robust – very high Detection Rate (True-Positive Rate) & very low False-Positive Rate always.

2. Real Time – For practical applications at least 2 frames per second must be processed.

3. Face Detection and not recognition - The goal is to distinguish faces from non-faces (face detection is the first step in the identification process)

The algorithm has mainly 4 stages –

1. Haar Features Selection

2. Creating Integral Image

3. Adaboost Training algorithm

4. Cascaded Classifiers

The features employed by the detection framework universally involve the sums of image pixels within rectangular areas. As such, they bear some resemblance to Haar Classifier function, which have been used previously in the realm of image-based object detection. However, since the features used by Viola and Jones all rely on more than one rectangular area, they are generally more complex. The figure at right illustrates the four different types of features used in the framework. The value of any given feature is always simply the sum of the pixels within clear rectangles subtracted from the sum of the pixels within shaded rectangles. As is to be expected, rectangular features of this sort are rather primitive when compared to alternatives such as steerable filters. Although they are sensitive to vertical and horizontal features, their feedback is considerably coarser.

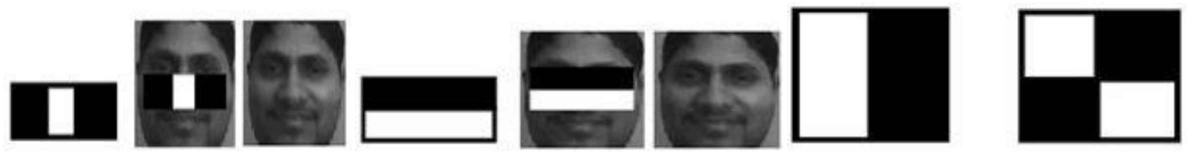


Figure 3.1: Features Using in Viola Jones Algorithm

1. Haar Features – All human faces share some similar properties. This knowledge is used to construct certain features known as **Haar Features**.

The properties that are similar for a human face are:

- The eyes region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.
- That is useful domain knowledge
- Location - Size: eyes & nose bridge region
- Value: darker / brighter

The 4 features applied in this algorithm are applied onto a face and shown on the left.

Rectangle features: - Value = Σ (pixels in black area) - Σ (pixels in white area)

- Three types: two-, three-, four-rectangles, Viola & Jones used two-rectangle features
- For example: the difference in brightness between the white &black rectangles over a specific area
- Each feature is related to a special location in the sub-window

Learning algorithm

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub-window, there are a total of 162,336possible features, and it would be prohibitively expensive to evaluate them all. Thus, the object detection framework employs a variant of the learning algorithm 'adaboost' on both to select the best features and to train classifiers that use them. This algorithm constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers.

$F(x) = a1\ f1(x) + a2\ f2(x) + a3\ f3(x) + \ldots..$

Where 'x' is the object/image, 'f' is the weak classifier, 'a' is the weight, and 'F' is the Strong Classifier

Features as weak classifiers - Each single rectangle feature may be regarded as a simple weak classifier

An iterative algorithm - AdaBoost performs a series of trials, each time selecting a new weak classifier

Weights are being applied over the set of the example images - during each iteration, every example/image receives a weight determining its importance

Initially, all weights set equally

Repeat 'T' times

Step 1: choose the most efficient weak classifier that will be a component of the final strong classifier. This weak classifier has the least weighted error.

Step 2: Update the weights to emphasize the examples which were incorrectly classified

- This makes the next weak classifier to focus on “harder” examples
- The misclassified ones are marked with red circles.

Step 3: Increase the weights on the training examples that were misclassified.

Step 4: Repeat Steps 1 to 3 “T” times where “T” is the number of weak classifiers.

Final (strong) classifier is a weighted combination of the T “weak” classifiers weighted according to their accuracy.

In the equation below T is the number of weak classifiers, h(x) is the final strong classifier, ‘a (t)’ are the weights and ‘ht(x)’ is the weak classifier.

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{Otherwise} \end{cases}$$

Equation for Strong Classifier

Advantages of Viola-Jones Algorithm:

- ✓ Extremely fast feature computation
- ✓ Efficient feature selection
- ✓ Scale and location invariant detector
- ✓ Instead of scaling the image itself (e.g. pyramid-filters), we scale the features.
- ✓ Such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, hands)

Disadvantages of Viola-Jones Algorithm:

- ❖ Detector is most effective only on frontal images of faces
- ❖ It can hardly cope with 45 degree face rotation both around the vertical and horizontal axis.
- ❖ Sensitive to lighting conditions

- ❖ We might get multiple detections of the same face, due to overlapping sub-windows.

3.2 Background Subtraction Algorithm

Background subtraction, also known as Foreground Detection, is a technique in the fields of image processing and computer vision wherein an image's foreground is extracted for further processing (object recognition etc.). Generally an image's regions of interest are objects (humans, cars, text etc.) in its foreground. After the stage of image preprocessing (which may include removing image noising, post processing like morphology etc.) object localization is required which may make use of this technique. Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called "background image", or "background model". Background subtraction is mostly done if the image in question is a part of a video stream. Background subtraction provides important cues for numerous applications in computer vision, for example surveillance tracking or human poses estimation. However, background subtraction is generally based on a static background hypothesis which is often not applicable in real environments. With indoor scenes, reflections or animated images on screens lead to background changes. In a same way, due to wind, rain or illumination changes brought by weather, static backgrounds methods have difficulties with outdoor scenes.

A robust background subtraction algorithm should be able to handle lighting changes, repetitive motions from clutter and long-term scene changes. The following analyses make use of the function of $V(x,y,t)$ as a video sequence where t is the time dimension, x and y are the pixel location variables. E.g. V (1, 2, 3) is the pixel intensity at (1, 2) pixel location of the image at $t = 3$ in the video sequence.

Using frame differencing

A motion detection algorithm begins with the segmentation part where foreground or moving objects are segmented from the background. The simplest way to implement this is to take an image as background and take the frames obtained at the time t, denoted by I(t) to compare with the background image denoted by B. Here using simple arithmetic calculations, we can segment out the objects simply by using image subtraction technique of computer vision meaning for each pixels in I(t), take the pixel value denoted by P[I(t)] and subtract it with the corresponding pixels at the same position on the background image denoted as P[B].

In mathematical equation, it is written as:

$$P[F(t)] = P[I(t)] - P[B]$$

The background is assumed to be the frame at time *t*. This difference image would only show some intensity for the pixel locations which have changed in the two frames. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static. A threshold "Threshold" is put on this difference image to improve the subtraction.

$$|P[F(t)] - P[F(t+1)]| > \text{Threshold}$$

This means that the difference image's pixels' intensities are 'thresholded' or filtered on the basis of value of Threshold. The accuracy of this approach is dependent on speed of movement in the scene. Faster movements may require higher thresholds.

Mean filter

For calculating the image containing only the background, a series of preceding images are averaged. For calculating the background image at the instant *t*,

$$B(x,y) = \frac{1}{N}\sum_{i=1}^{N} V(x,y,t-i)$$

Here *N* is the number of preceding images taken for averaging. This averaging refers to averaging corresponding pixels in the given images. *N* would depend on the video speed (number of images per second in the video) and the amount of movement in the video. After calculating the background *B(x, y)* we can then subtract it from the image *V(x, y, t)* at time *t*=t and threshold it. Thus the foreground is

$$|V(x,y,t) - B(x,y)| > \mathrm{Th}$$

Here '*Th'* is threshold. Similarly we can also use median instead of mean in the above calculation of *B*(*x*,*y*).

Usage of global and time-independent Thresholds (same 'Th' value for all pixels in the image) may limit the accuracy of the above two approaches.

Running Gaussian average

For this method, Wren The author proposed fitting a Gaussian probabilistic density function on the most recent n frames. In order to avoid the fitting of from scratch at each new frame time t, a running (or on-line cumulative) average is computed.

The value of every pixel is characterized by mean μ_t and variance σ_t^2. The following is a possible initial condition (assuming that initially every pixel is background):

$$\mu_0 = I_0$$

$\sigma_0^2 = <$Some default value $>$

Here I_t is the value of the pixel's intensity at time t. In order to initialize variance, we can, for example, use the variance in x and y from a small window around each pixel.

Note that background may change over time (e.g. due to illumination changes or non-static background objects). To accommodate for that change, at every frame t, every pixel's mean and variance must be updated, as follows:

$$\mu_t = \rho I_t + (1-\rho)\mu_{t-1}$$

$$\sigma_t^2 = d^2\rho + (1-\rho)\sigma_{t-1}^2$$

$$d = |(I_t - \mu_t)|$$

Here ρdetermines the size of the temporal window that is used to fit the (usually$\rho = 0.01$) and $d$is the Euclidean distance between the mean and the value of the pixel.

We can now classify a pixel as background if its current intensity lies within some confidence interval of its distribution's mean:

$$\frac{|(I_t - \mu_t)|}{\sigma_t} > k \longrightarrow Foreground$$

$$\frac{|(I_t - \mu_t)|}{\sigma_t} \leq k \longrightarrow Background$$

Where the parameter k is a free threshold (usually$k = 2.5$). A larger value for $k$allows for more dynamic background, while a smaller $k$increases the probability of a transition from background to foreground due to more subtle changes.

Chapter One

4. System Design

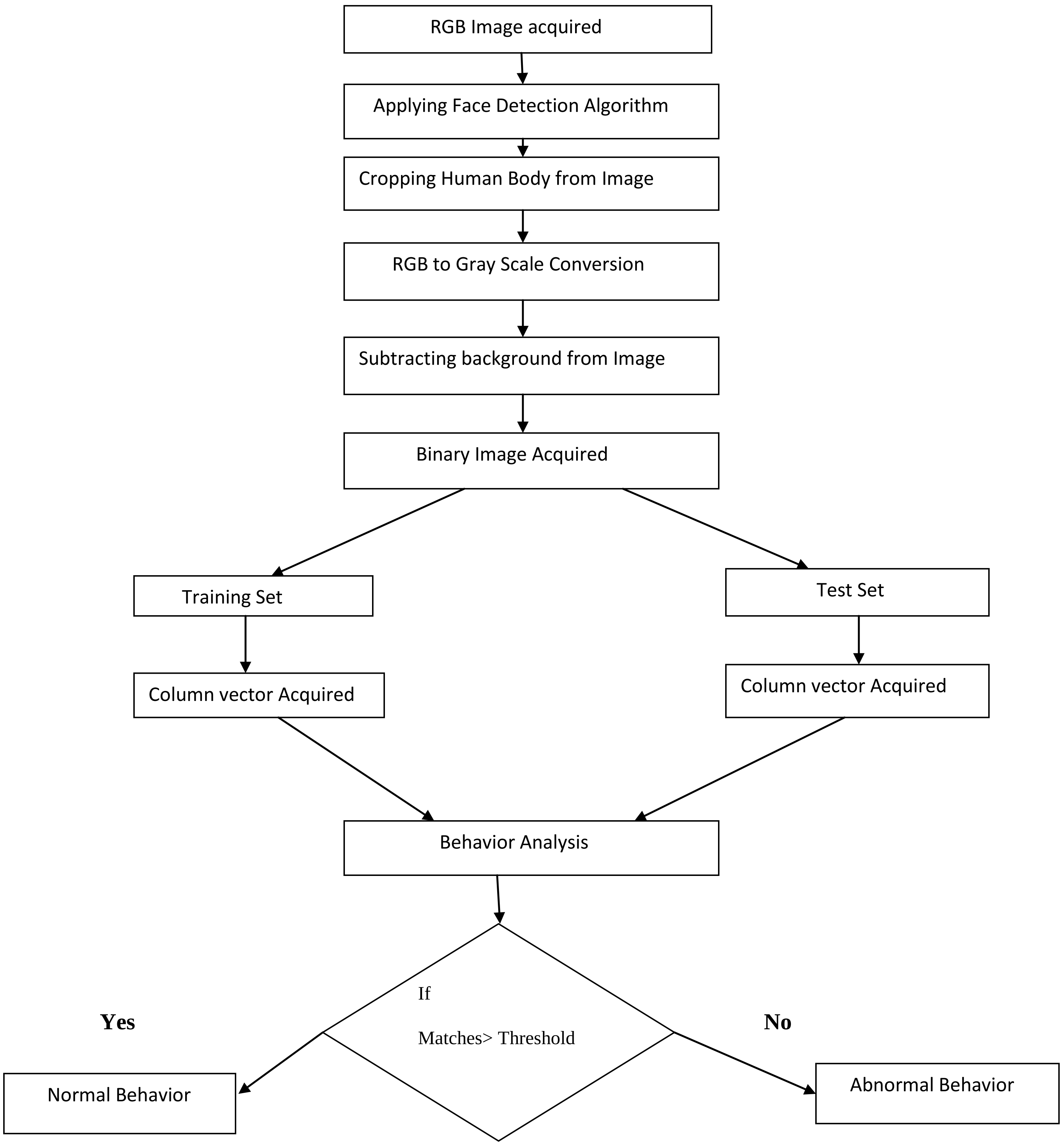
4.1Tools, algorithms and methodology

For our experiment, we used MATLAB R2013a and Image Processing Toolbox.

Algorithm we used is Viola Jones Algorithm.

Methodologies we followed are Morphologically Open Image, Subtracting Back ground from image, Converting Gray scale to Binary Image, Converting Binary Image to Column Vector etc.

4.2Flowchart



4.3Data Collection

For this experiment, we could not find any universal image set. Therefore we need to go different places to collect images. The pictures were taken by 5 mega pixel mobile phone and 12 mega pixel digital camera. As we are only interested in abnormal behaviors, so we took lots of images with different poses. The pictures were taken in this way so that camera angle varied from human about 45 to 90 degrees. The distance between camera and human is about 3 to 5 meters. 5 meters is a standard distance because increasing the distance can make the system inefficient. Less than 3 meters brings human too close. As a result, sometimes image gets blurred as we are using simple digital camera. Besides, we took the image under different scenario for example, low-lights, sunny weather, static background etc. Here we have shown some of the data that we collected

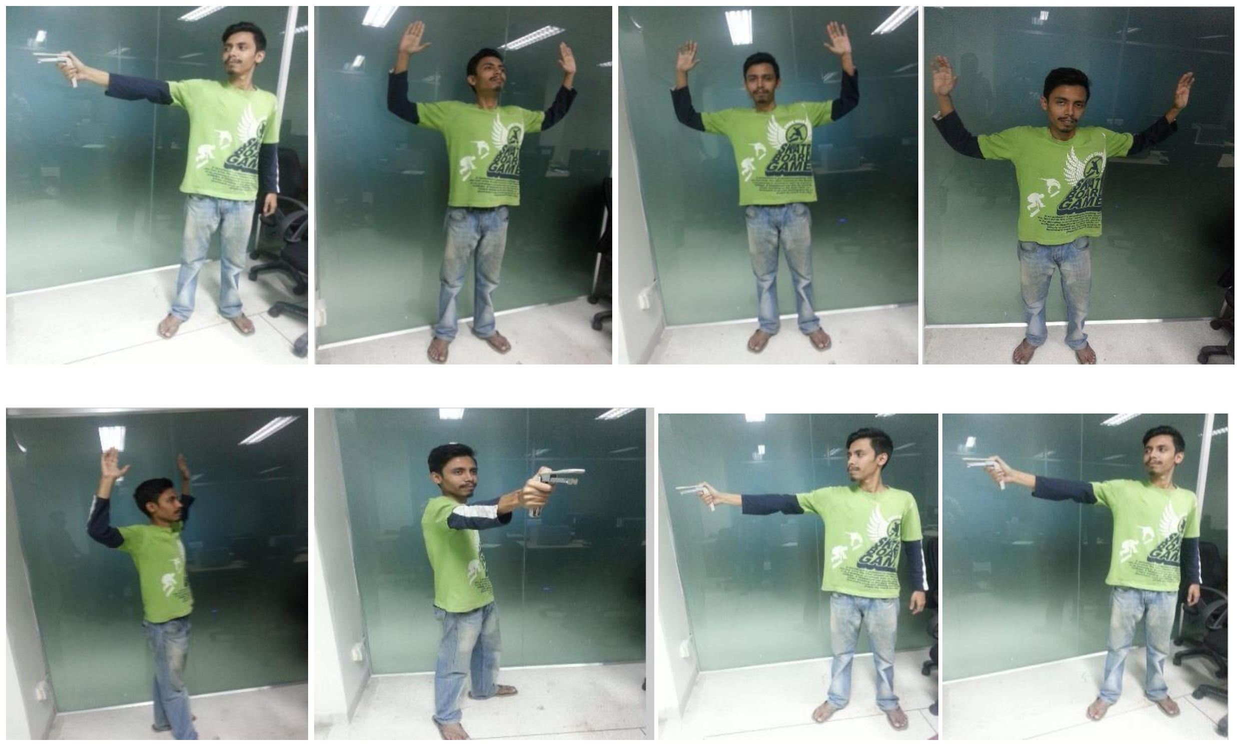


Figure 4.1: Sample of data collection

We have had some photos where the face is not clearly visible. We are ignoring those images for this algorithm. . The following figures show such scenarios.

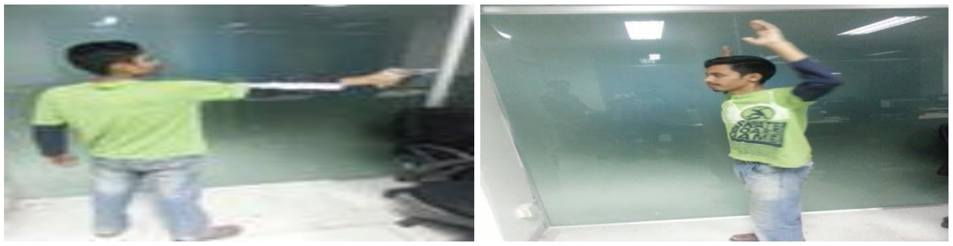


Figure 4.2: Images which are not considerable

We have taken all the images around BRAC University, Dhaka. All the images are considered as training image. We have trained 100 images for this experiment. After acquiring image, the first step of the experiment is to detect face of human.

4.4 Detect face Using Viola Jones Algorithm

This algorithm is used to detect face. The algorithm has mainly 4 stages –

1. Haar Features Selection

2. Creating Integral Image

3. Adaboost Training algorithm

4. Cascaded Classifiers

Viola Jones Algorithm detects faces based on some human face properties. For example, most of the time eyes region is darker than upper-cheeks. The nose region is brighter than the eyes. Based on these features it makes a decision whether it is face or not. These features are also known as Haar Features. After successfully detecting the face, it draws a rectangular shape around the face to indicate the confirmation.

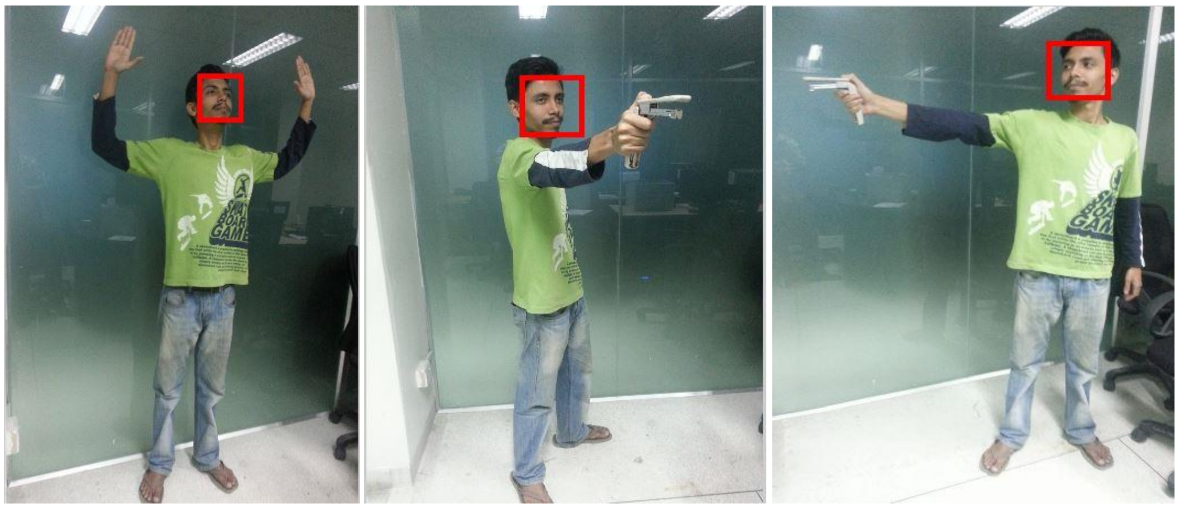


Figure 4.3: Images of Detecting Human faces.

4.5 Measure human body from image after detecting face

As Viola Jones algorithm detects face and we store the matrix value in a variable. Now we know that there is a ratio relation between human face and human body. Calculating the ratio (mention the ratio), we manually add an extra matrix (dimension of extra matrix) value with the variable where the result of face is stored. After adding two matrices, we get the entire human body and store it in a variable.

4.6 Crop Human body from Image

From the above method, we have found the entire human body. Now we need to crop only human from image. The idea of cropping image is we are only interested on human behavior plus we ensure to reduce noise and unnecessary stuffs from the image. Since we stored the human body, we just crop it from the original image using the same dimension of the box.

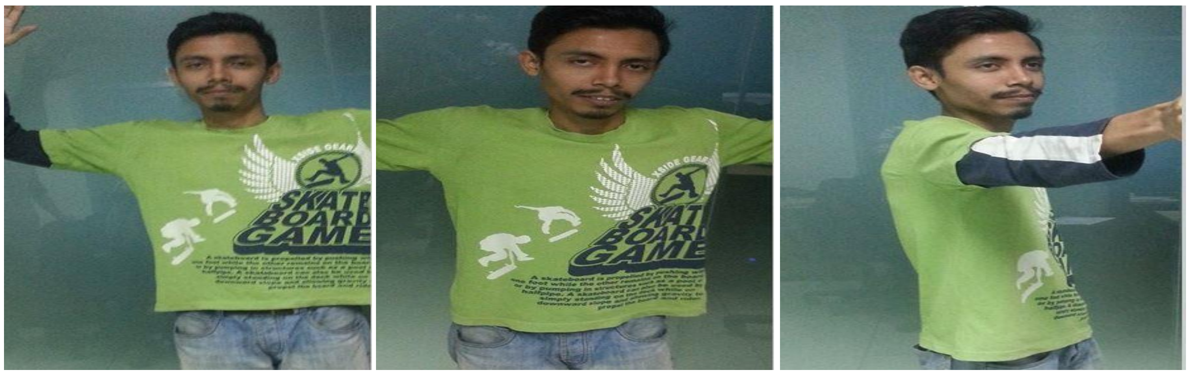


Figure 4.4: Cropping human body from original images

4.7 Convert the Cropped image From RGB To Gray Scale

In order to move further experiment, we need to convert the cropped image from RGB to Gray scale. The idea behind this conversion is to reduce space and processing time. RGB color model is a model where three colors Red, Green and Blue are added together to form an array of color. RGB consists of three channels. On the other hand, Gray scale image is an image in which the value of each pixel is single sample. It means it only carries the information of intensity. Basically, gray scale image contains two colors Black and White. Between these two colors Gray scale images have many shades. Moreover, Gray scale consists of one channel. The conversion is done by calculating the weighted sum of three colors.

The linear I is calculated by .2989* R + .5870* G + .1140* B in MATLAB, the toolbox we have used in our work.

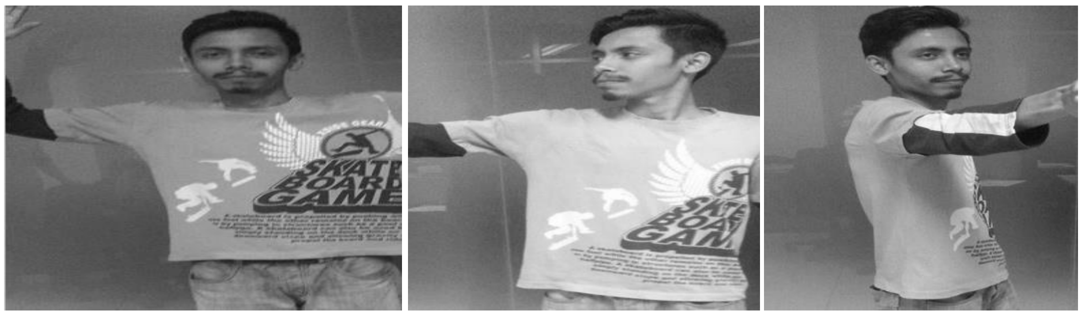


Figure 4.5: RGB to Gray Scale Conversion

4.8Removing Background From Gray Scale Image

In this part, we are interested to remove background from gray scale image because removing background reduces the amount of noise from the image which helps us to acquire better accuracy. Removing background consists of some sub stages. Among them, first we need to create a structural element.

4.8.1Creating Structural Elements

Structural element is an essential part of erosion and dilation. It is mainly used to examine on input image. . Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's.The center pixel of the structuring element, called the origin. In our experiment, we create a structural element using a system which creates a flat, linear structuring element. Here 35 specify the length of structural element and 90 specifies the angle of line. Using this element, we performed morphological operation like erosion and dilation.

4.8.2Morphologically Open Image

Morphological open image performs morphological opening operation in gray scale image or binary image with the structural elements that we created in above. The morphological open operation consists of two operations. At first, there is an erosion of an image and after that dilation operation is performed.

4.8.2.1Eroding Gray Scale Image

This operation erodes the binary image or gray scale image. The created structural element is passed here as an argument. After that the structural element executed over the gray scale image and returns the final eroded image. Here our corresponding image is a gray scale image, so it performs gray scale erosion. After erosion operation, the image lost thickness and details of the image.

4.8.2.2Dilating Gray Scale Image

This operation dilates the binary or gray scale image. The created structural element is passed here as an argument. After that the structural element executes over the gray scale image and returns the final dilated image. Here our corresponding image is a gray scale image. So it performs gray scale dilation. It is used to thicken the objects in a proper way. Since after erosion operation, the image lost thickness and details, Dilation operation brings back the image closed to the previous one.

Finally, we got our desired tools for subtracting background from the gray scale image and we stored that tool in to a variable.

<u>4.8.3Subtracting Back Ground from Gray Scale Image</u>

The operation is used to subtract the background from gray scale image. In this function, the gray scale image and the tools that we saved in a variable are sent as arguments. Later it subtracts each element in variable from the corresponding element of gray scale image and returns the difference in the corresponding element of the output. The output is also a gray scale image having same size of previous gray scale image. Finally, the output is a subtracted background gray scale image.

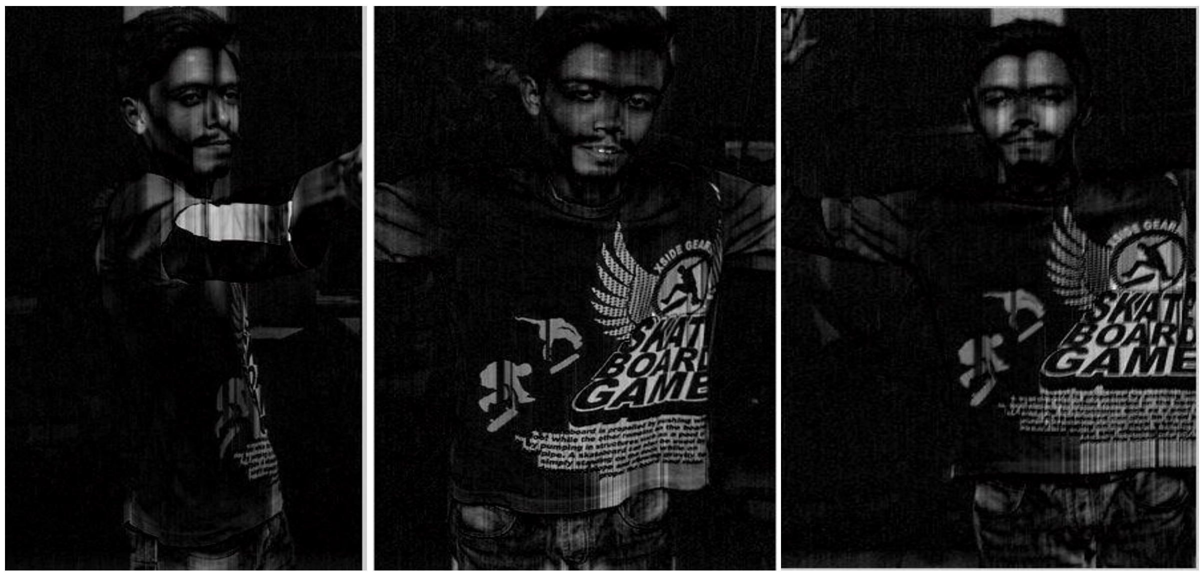


Figure 4.6: Images after subtracting background

<u>4.9 Using Noise Removal Filtering</u>

Wiener2 is a low-pass filter and it is used to reduce noise from a gray scale image.A low-pass filter is a filter that allows signals below a cutoff frequency and reduces signals above the cutoff frequency. By removing some frequencies, the filter creates a smoothing effect. That is, the filter produces slow changes in output values to make it easier to see trends and boost the overall signal-to-noise ratio with minimal signal degradation. Using this filter, the gray scale images break down by constant power additive noise. Wiener2 uses a pixel wise adaptive Wiener method. The result is calculated based on the each pixel value of its neighbor's pixels. In our experiment, we provided 3 by 3 matrix as one of the arguments of Wiener2 filter. Based on this matrix size, it finds out the mean and standard deviation of its neighbor pixels. Here we used 3 by 3 matrix as it gave us optimum result for the all the gray scale images. After performing operation, we saved the value into a variable.

4.10 Converting Gray Scale Image To Binary Image

In order to convert gray scale image to binary image, at first we need to find out the threshold level of gray image. To do so, we used a function which computes a global threshold that can be used to convert a gray image to a binary image. After determining the threshold, we stored in variable called level. In our experiment, we divide the gray scale image by 1.3 as it gave us a better threshold.

After that, we used another function which converts the gray-scale image to a binary image. The output image replaces all pixels in the input image with the value 1 (white) where the pixel value is greater than level and replaces all other pixels with the value 0 (black) where the pixel value is less than level. By doing this, finally, we acquired the desired binary image.

In next page, we showed some of the binary images.

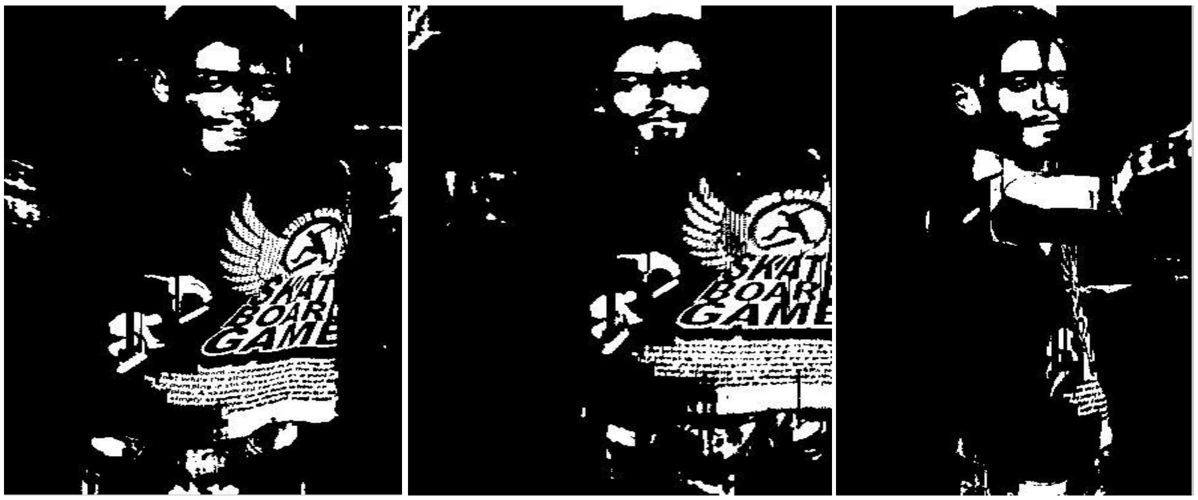


Figure 4.7: Binary Image

4.11 Training

This part is the most important part of our experiment. The efficiency of the output depends on how efficiently the data is trained. In training part, the binary image converts into column vector having same size of rows of binary image with just one column.

4.11.1 Converting Binary Image to Column Vector

From above we acquired the binary image which has size of (425, 125). So we can consider the binary image as matrix having 425 rows and 125 columns. The matrix holds the binary value of 1 and 0. Precisely, we can say that the matrix stores binary value of 1 where the part of human body is found. Otherwise, it stores 0. Now we have to compute the value by iterating every column of each row. It means every row has 125 columns having contains binary value. As the value is binary, we need to convert them into decimal value. In order to do so, we use the formula of binary to decimal conversion given below:

Row1=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

|

|

Row425=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

Now we need to store the value. To do so, we created a matrix having size of (425, 1) which has exactly same rows of the binary image. Actually, this is the column vector which is store the decimal value of each row. After calculating row1, the value was stored act column vector of (1,1). After calculating row2, the decimal value was stored in column vector of (2, 1) and so on. This is how the column vector stored the decimal value of a binary image. Since we trained 100 images, we got 50 column vectors. After that we saved all column vectors in a global array. After training, we saved them in the computer hard drive. This value is needed when compared them with the testing column vector.

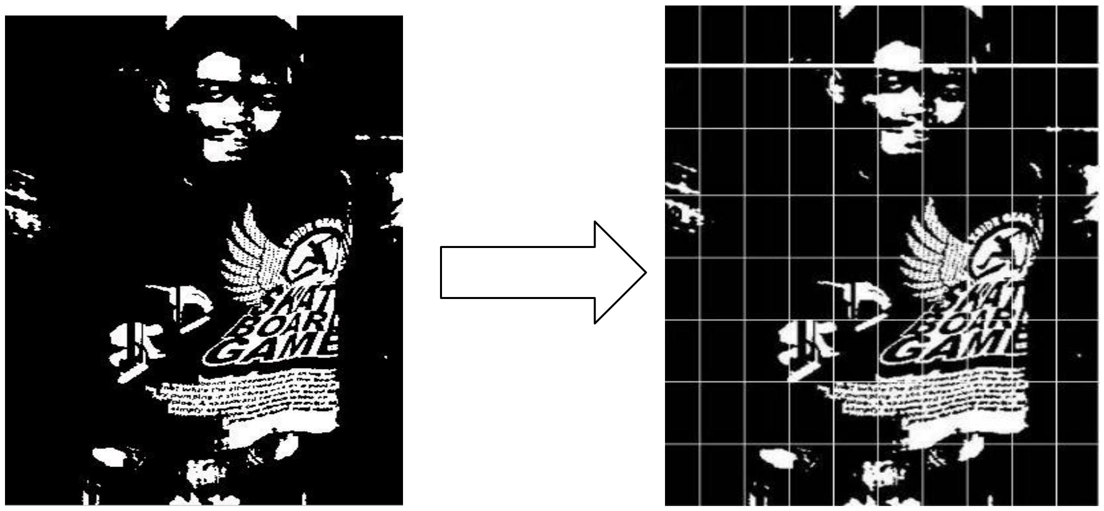


Figure 4.8: Representing Binary image in Matrix form

<u>4.12 Process Testing Image</u>

Since our proposed system detects abnormal behavior from the video. Unfortunately, we did not find any relevant video for our experiments. Consequently, we took a video ourselves which contained full of abnormal behaviors. As video progresses, the frame of video was checked and detects if any abnormal behavior exists. Processing testing image consists of several parts. The first of them was choosing frame from video file.

<u>4.12.1 Choosing Frame from Video</u>

First of all, we are not interested of checking every frame as it is time consuming and inefficient way. We noticed that there were at least 5-6 frames between the major changes of two images. For example, in frame1 a man is walking, suddenly, he sat down. So if we look at the frame no. of man sitting down, we find it frame no 6 or 7. So it clearly indicated that there needed to be at least 5 to 7 frames to get over change. Rightly so, when video progressed, we checked out the first frame and after then we repeat checking after 5 frames until we found out some abnormal behaviors. If the abnormal behavior found, it showed the human and drew attention to take further steps.

After choosing frame, we used that frame for further processing.

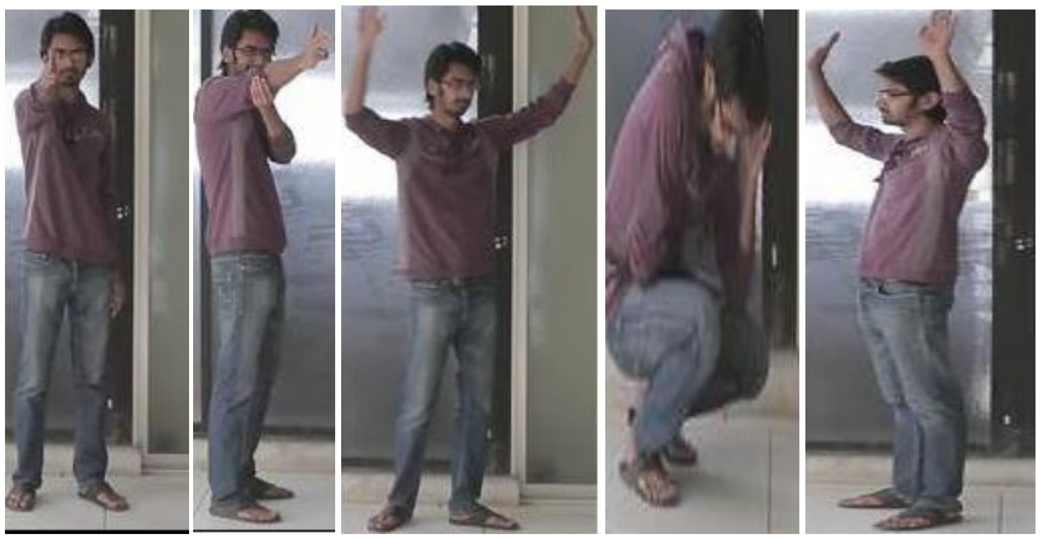


Figure 4.0: Sample Test Frame

4.12.2Detect face Using Viola Jones Algorithm

This algorithm is used to detect face. The algorithm has mainly 4 stages –

1. Haar Features Selection

2. Creating Integral Image

3. Adaboost Training algorithm

4. Cascaded Classifiers

Viola Jones Algorithm detects faces based on some human face properties. For example, most of the time eyes region is darker than upper-cheeks. The nose region is brighter than the eyes. Following these feature it makes a decision whether it is face or not. These features are also known as Haar Features. After successfully detecting the face, it draws a rectangular shape around the faces to indicate the confirmation.

4.12.3 Measure human body from image after detecting face

As Viola Jones algorithm detects face and we store the matrix value in a variable. Now we know that there is a ratio relation between human face and human body. Calculating the ration, we manually add an extra matrix value with the variable where the result of face is stored. After adding two matrices, we get the entire human body and store it in a variable.

4.12.4 Crop Human body from Image

From the above method, we have found the entire human body. Now we need to crop only human from image. The idea of cropping image is we are only interested on human behavior plus we ensure to reduce noise and unnecessary stuffs from the image. Since we stored the human body, we just crop it from the original image.

4.12.5 Convert the Cropped image From RGB To Gray Scale

In order to move further experiment, we need to convert the cropped image from RGB to Gray scale. The idea behind this conversion is to reduce space and processing time. RGB color model is a model where three colors Red, Green and Blue are added together to form an array of color. RGB consists of three channels. On the other hand, Gray scale image is an image in which the value of each pixel is single sample. It means it only carries the information of intensity. Basically, gray scale image contains two colors Black and White. Between these two colors Gray scale images have many shades. Moreover, Gray scale consists of one channel. The conversion is done by calculating the weighted sum of three colors.

The linear I is calculated by .2989* R + .5870* G + .1140* B in MATLAB, the toolbox we have used in our work.

4.12.6 Removing Background From Gray Scale Image

In this part, we are interested to remove background from gray scale image because removing background reduces the amount of noise from the image which helps us to acquire better accuracy. Removing background consists of some sub stages. Among them, first we need to create a structural element.

4.12.6.1 Creating Structural Elements

Structural element is an essential part of erosion and dilation. It is mainly used to examine on input image. Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's.The center pixel of the structuring element, called the origin. In our experiment, we create a structural element using system which creates a flat, linear structuring element. Here 35 specify the length of structural element and 90 specifies the angle of line. Using this element, we performed morphological operation like erosion and dilation.

4.12.6.2 Morphologically Open Image

This function performs morphological opening operation in gray scale image or binary image with the structural elements that we created in above. The morphological open operation consists of two operations. At first, there is a dilation of an image and after that erosion operation is performed.

4.12.6.2.1Eroding Gray Scale Image

This operation erodes the binary image or gray scale image. The created structural element is passed here as an argument. After that the structural element executed over the gray scale image and returns the final eroded image. Here our corresponding image is a gray scale image, so it performs gray scale erosion. After erosion operation, the image lost thickness and details of the image.

4.12.6.2.2Dilating Gray Scale Image

This operation dilates the binary or gray scale image. The created structural element is passed here as an argument. After that the structural element executes over the gray scale image and returns the final dilated image. Here our corresponding image is a gray scale image. So it performs gray scale dilation. It is used to thicken the objects in a proper way. Since after erosion operation, the image lost thickness and details, Dilation operation brings back the image closed to the previous one.

Finally, we got our desired tools for subtracting background from the gray scale image and we stored that tool in to a variable.

4.12.6.3 Subtracting Back Ground from Gray Scale Image

The operation is used to subtract the background from gray scale image. In this function, the gray scale image and the tools that we saved in a variable are sent as arguments. Later it subtracts each element in variable from the corresponding element of gray scale image and returns the difference in the corresponding element of the output. The output is also a gray scale image having same size of previous gray scale image. Finally, the output is a subtracted background gray scale image.

4.12.7 Using Noise Removal Filtering

Wiener2 is a low-pass filter and it is used to reduce noise from a gray scale image. Using this filter, the gray scale images break down by constant power additive noise. Wiener2 uses a pixel wise adaptive Wiener method. The result is calculated based on the each pixel value of its neighbor’s pixels. In our experiment, we provided 3 by 3 matrix as one of the arguments of Wiener2 filter. Based on this matrix size, it finds out the mean and standard deviation of its neighbor pixels. Here we used 3 by 3 matrix as it gave us optimum result for the all the gray scale images. After performing operation, we saved the value into a variable.

4.12.8 Converting Gray Scale Image To Binary Image

In order to convert gray scale image to binary image, at first we need to find out the threshold level of gray image. To do so, we used a function which computes a global threshold that can be used to convert a gray image to a binary image. After determining the threshold, we stored in variable called level. In our experiment, we divide the gray scale image by 1.3 as it gave us a better threshold.

After that, we used another function named which converts the grayscale image to a binary image. The output image replaces all pixels in the input image with the value 1 (white) where the pixel value is greater than level and replaces all other pixels with the value 0 (black) where the pixel value is less than level. By doing this, finally, we acquired the desired binary image.

4.12.9 Converting Binary Image to Column Vector

From above we acquired the binary image which has size of (425, 125). So we can consider the binary image as matrix having 425 rows and 125 columns. The matrix holds the binary value of 1 and 0. Precisely, we can say that the matrix stores binary value of 1 where the part of human body is found. Otherwise, it stores 0. Now we have to compute the value by iterating every column of each row. It means every row has 125 columns having contains binary value. As the value is binary, we need to convert them into decimal value. In order to do so, we use the formula of binary to decimal conversion given below:

Row1=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

|

|

Row425=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

Now we need to store the value. To do so, we created a matrix having size of (425, 1) which has exactly same rows of the binary image. Actually, this is the column vector which is store the decimal value of each row. After calculating row1, the value was stored as column vector of (1, 1). After calculating row2, the decimal value was stored in column vector of (2, 1) and so on. This is how the column vector stored the decimal value of a binary image.

Now we are interested to find out whether the behavior is abnormal or not. In order to do, we have to match the value of testing column vector with trained column vector. The analysis is shown in next step.

4.13 Behavior Analysis:

In our dataset, the column vector of each binary image having size of (425, 1) have been stored. Now, the test image is also converted into the binary image. Then we compute the binary image into a column vector with same size of trained column vector (425, 1). Now we find out which trained image is very close enough to the test image. In order to so, we need to calculate the Euclidian distance between test column vector and trained column vector.

4.13.1Euclidian Distance

Euclidian distance is the ordinary distance between two points. It shows how far two points are located from each other. The Euclidian distance formula is given below:

The **Euclidean distance** between point **p** and **q** is the:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

We used the formula which had given below:

D = sqrt(sum((G - G1) .^ 2))

Here G is the column vector of test image and G1 is the column vector of train image. The explanation of Euclidian distance formula is to subtract two points by element wise and then adding them up by squaring the subtracting the result. After adding all results, finally we perform square root over the result and the final result is our desired Euclidian distance.

The Euclidian distance is calculated by drawing two curves in graph. One curve is for test column vector and another one is for trained column vector. Below a graph is shown

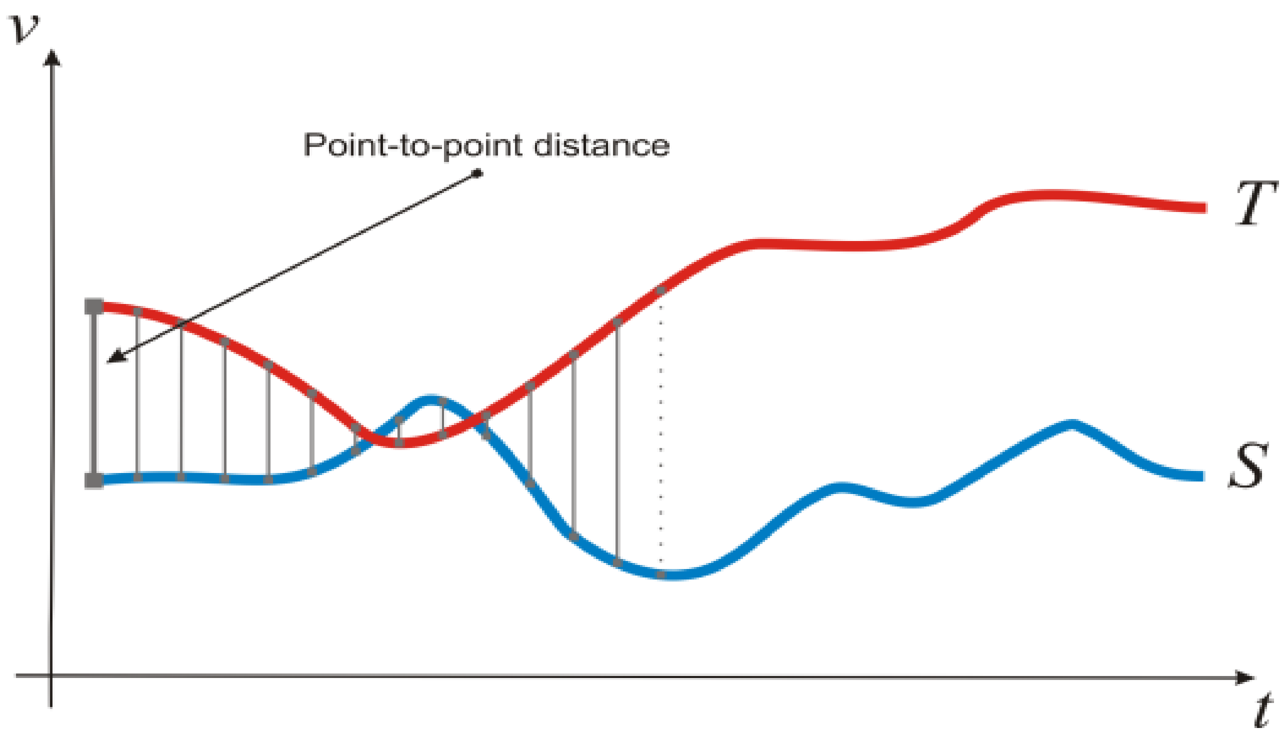


Figure 4.10: Euclidean Distance Measurement Graph

T and S represent the curves of two column vectors. Between each pair of points, a line is drawn to indicate the point to point distance. After drawing the line, the Euclidian formula is used. This methodology will be followed for every pair of test and trained column vector. Then we store the result in an auxiliary array. The result shows that how two vectors are close to each other. It means it shows the similarities or between two vectors. In other words, we can say that the result tells us the difference between two column vectors. Now we find out the minimum value of an array. The minimum value indicates that the result of column vector is the best match or gets the highest similarity compared with test column vector. If the result is very close to our threshold (in that case threshold value is 4), then we can say that, the two column vectors are same. Hence, it is obvious that the test image and trained image is exactly same. As we only train the abnormal behavior of human. We can say that the test image contains abnormal behavior and applicable to draw further attention.

Chapter 2

4. System Design

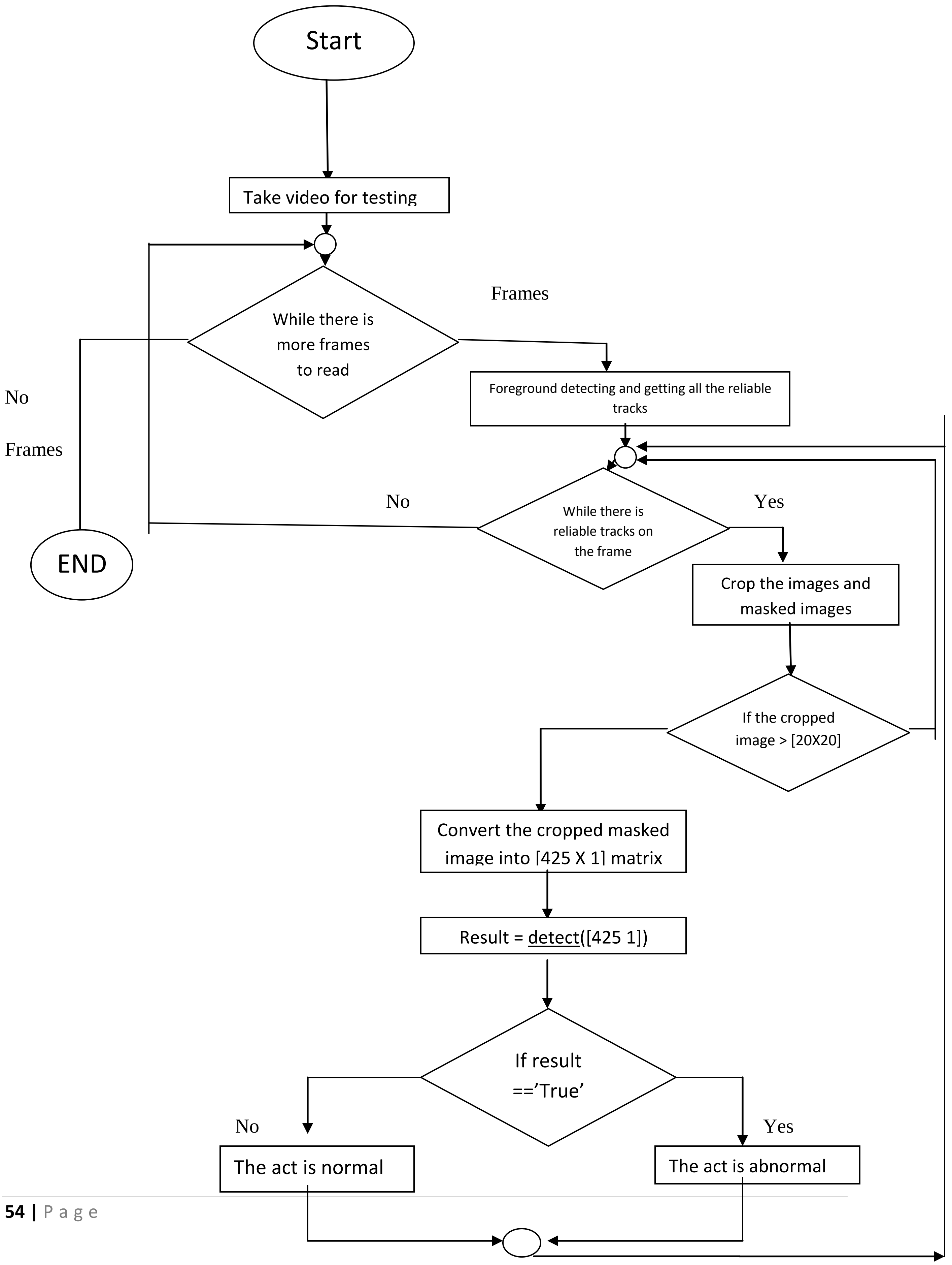
4.1 Tools, algorithms and methodology

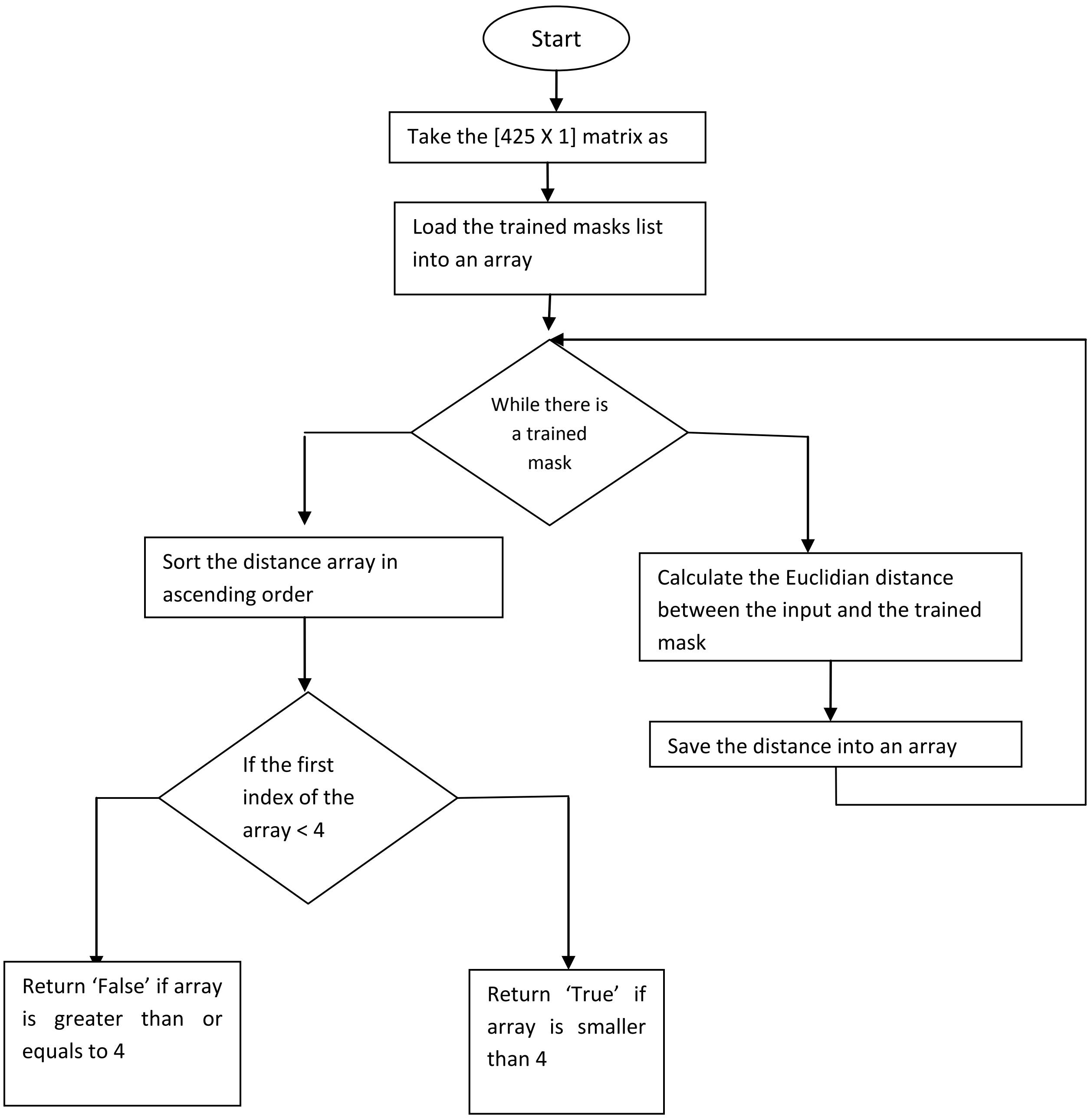
For our experiment, we used MATLAB R2013a and Image Processing Toolbox.

Algorithm we used is Foreground detection Algorithm.

Methodologies we followed are Detect human from image, Morphological image processing, and Crop Binary image from frame and finally convert into column vector.

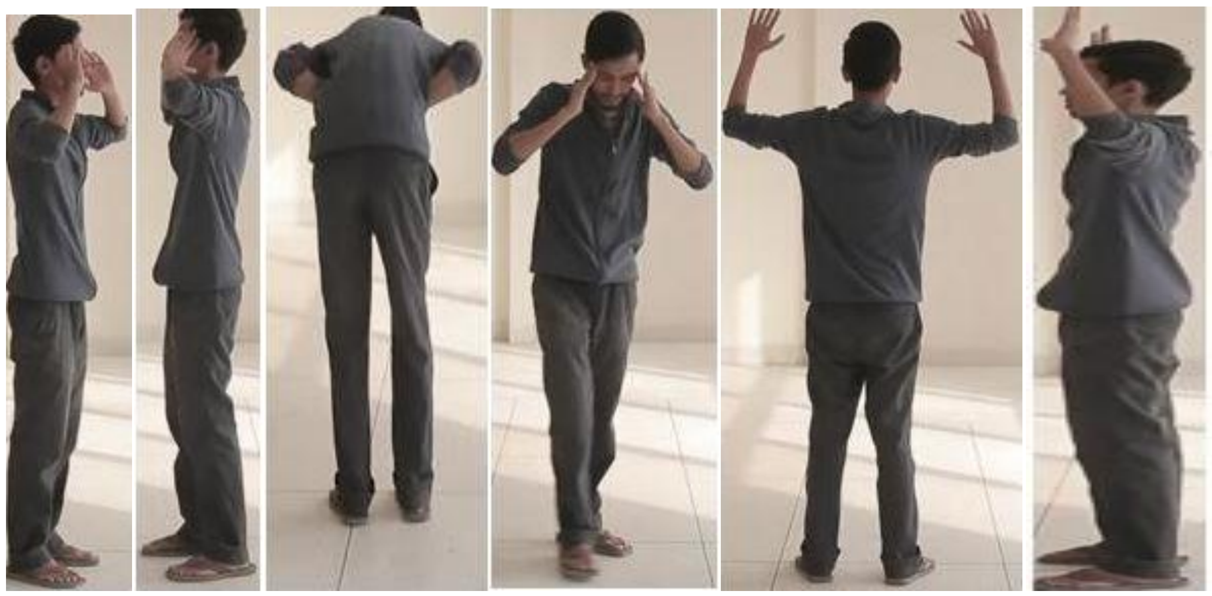
4.2Flowchart


Start
Take video for testing
While there is more frames to read
Frames
No
Frames
Foreground detecting and getting all the reliable tracks
No
Yes
END
While there is reliable tracks on the frame
Crop the images and masked images
If the cropped image > [20X20]
Convert the cropped masked image into [425 X 1] matrix
Result = detect([425 1])
If result =='True'
No
Yes
The act is normal
The act is abnormal


Start
Take the [425 X 1] matrix as
Load the trained masks list into an array
While there is a trained mask
Sort the distance array in ascending order
Calculate the Euclidian distance between the input and the trained mask
Save the distance into an array
If the first index of the array < 4
Return 'False' if array is greater than or equals to 4
Return 'True' if array is smaller than 4

4.3Data Collection:

In our experiment, we did not find any universal data set. So we manually captured a video and used it to train the image. The video was captured at DSLR camera where the camera position was fixed. As we are only keen in abnormal behavior of human, the video which captured was full of abnormal behavior with different poses. We took video under different scenarios such as sunny weather, low-light, normal daylight etc. The place was chosen in this way so that the moving object was only human. This was because in this experiment, we used the algorithm of Motion based Object detecting. So if anything moved other than human, then it also detects as object. So we gave a minimum threshold for detecting only human. Since we are only keen in detecting human, we had to be aware of choosing places. The video was taken at one of the buildings of BRAC University, Dhaka. In chapter 1 experiment, we ignored some of the poses as it failed to reach the result. Here in this experiment, we considered those poses and trained them. Consequently, it makes the system even more efficient than previous one. In this experiment, we trained 50 images with different poses. The images were taken from the particular frame of video file where abnormal behavior appeared. After taking video, the first step is to create system objects.



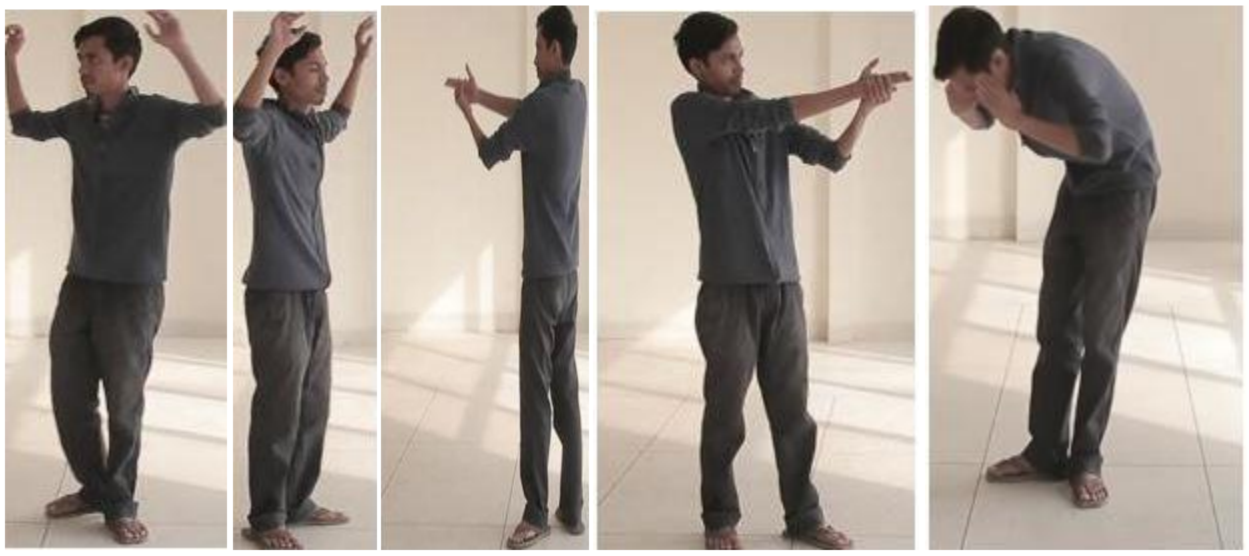


Figure 4.2.1: Sample data collection for training purpose

We also used those images which we ignored in chapter 1

Figure 4.2.2: Images which ignored in chapter 1

4.4 Create System Objects

Create System object is used to read frames from video, detecting foreground objects and displaying results. At first, we needed to initialize the video I/O. After that, we needed to create objects from reading video from a particular file. After that, we enabled to read every frame from video file while video was running. Since we wanted to see the video, we created two video players. First one was to display the original video while another one was to display the foreground mask. Then we needed to create system objects for foreground detection and blob analysis.

4.4.1 Create System Objects for Foreground Mask

The foreground detector is used to separate moving objects from the background. It returns a result which is a binary mask. Here the detector returns result based on Gaussian Mixture Model (GMM). In binary mask, the pixel value sets to be 1 (white) corresponding to the foreground. On the other hand, the pixel value sets to be 0 (Black) corresponding to the background.

In order to create Foreground detector we needed to pass some value as an argument. Here the number of Gaussian modes is 3 in the mixer model, number of training frame is 40 and minimum background ratio is set to be 0.7. This is the standard argument for the Foreground detector.

4.4.2 Create System Objects For Blob Analysis

The blob analysis system object is used to draw a shape into the foreground detected objects. In order to draw the shape, it maintains a bunch of rules. First of all, it finds out in foreground objects how many pixels are gathered together. In other words, how many foreground pixels are connected with each other and form a group. Basically, the blob analysis system object is used to find out such groups. Foreground pixel contains the binary value of 1. So it identifies the group of value 1. These pixels are known as connected components. To find out these groups, the system objects emphasize on some characteristics such as area, centroid and the bounding box.

In order to create the blob analysis system objects, some values needed to be passed as arguments. Here Bounding Box Output Port is true, Area Output Port is true, Centroid Output Port is true and Minimum Blob Area is set to be 400.

4.5 Read Frame From Video

As we mentioned earlier, we needed to choose a frame manually, where the abnormal behavior exists. In order to it, we ran the video and picked up our desired frame. After that, we sent it to go through other operations. This is how, frames were chosen in our experiments.

4.6 Detect Objects From Frame

The function returns the centroids and the bounding boxes of the detected objects. It also returns the binary mask, which has the same size as the input frame. Pixels with a value of 1 correspond to the foreground, and pixels with a value of 0 correspond to the background.

At first, the function performs motion segmentation using the foreground detector. After that it performs morphological operations on the resulting binary mask to remove noise from the pixel and finally fill the holes in the remaining blobs.

4.6.1 Detect foreground

We created the Foreground detect system objects in upper step. Now using this object, we detected foreground objects. The System object compares a color or gray-scale video frame to a background model to determine whether individual pixels are part of the background or the foreground. It then computes a foreground mask which is also known as binary mask.

The foreground detector requires a number of video frames and minimum background ratio in order to initialize the Gaussian Mixture Model (GMM). Here we used number of frames 40 and Minimum Background ratio 0.7. The idea of choosing these values is to provide us optimum result for every frame.

The foreground segmentation process is not perfect and sometimes includes noise which makes the efficiency lower. In order to remove noise we have to do some operations which described below:

4.6.1.1Morphologically Open Image

Morphologically open image function performs morphological opening operation in gray scale image or binary image with the structural elements that we created in above. The morphological open operation consists of two operations. At first, there is an erosion of an image and after that dilation operation is performed.

4.6.1.1.1 Creating Structural Element

Structural element is an essential part of erosion and dilation. It is mainly used to examine on input image. Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's.The center pixel of the structuring element, called the origin. In our experiment, we create a structural element using which creates a flat, linear structuring element. Here Rectangle is indicated the shape and [3, 3] represents a 3 by 3 matrix which actually implements on neighbor's pixels. Using this element, we performed morphological operation like erosion and dilation.

4.6.1.1.2 Eroding Gray Scale Image

This operation erodes the binary image or gray scale image. The created structural element is passed here as an argument. After that the structural element executed over the gray scale image and returns the final eroded image. Here our corresponding image is a binary image, so it performs binary erosion. After erosion operation, the image lost thickness and details of the image.

4.6.1.1.3 Dilating Gray Scale Image

This operation dilates the binary or gray scale image. The created structural element is passed here as an argument. After that the structural element executes over the gray scale image and returns the final dilated image. Here our corresponding image is a binary image. So it performs binary dilation. It is used to thicken the objects in a proper way. Since after erosion operation, the image lost thickness and details, Dilation operation brings back the image closed to the previous one.

4.6.1.2 Morphologically Close Image

This function performs morphologically closing on the gray scale or binary image. At first, we needed to create the structural element. Bu using this elements it performs morphologically image closing. It is used to fill up the gaps of white pixels in binary images.

4.6.1.2.1 Creating Structural Element

The structural element is mainly used to examine on input image. . Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's.The center pixel of the structuring element, called the origin. In our experiment, we create a structural element using which creates a flat, linear structuring element. Here Rectangle is indicated the shape and [15, 15] represents a 15 by 15 matrix which actually implements on neighbor's pixels and provides the desired output.

4.6.1.3 Fill up The Holes In Binary Image

In this step, we used a function that fills up the holes in binary image. In upper method, we use morphologically close image to fill up the gap of white pixels but unfortunately, sometimes it fails to reach some white pixel. That is why, we used this method.

4.6.2 Blob Analysis

In above, we mentioned that we created a blob analysis system object to perform this operation. The operation is performed based on some characteristics like area, centroid and bounding box.

As the functions finds out some groups of connected white pixels, we needed to give a shape on them. To do so, we first saved the result in a variable. After that we created a system object for shape inserter. So now, we used that objects into that variable where the detected human object was stored. The reason of selecting Rectangle shape is, we were interested only o human and rectangle shape was the best way to represent a human rather than using disk or square shape.

Finally, we got a binary image where a human object was detected bounding by a rectangular box and stored in a variable.

4.7 Crop Human Body From the Binary Image

Now we got a binary image with human. We did not need the whole binary image. Here we observed whether the behavior was abnormal or not. So we only needed the human portion. We saved the value of human portion in a variable. In order to get it, we cropped the value from the binary image. The idea of cropping image is to increase processing time and eliminate the noise. The next part was to train the cropped binary image.

4.8 Training

This part is the most important part of our experiment. The efficiency of the output depends on how efficiently the data is trained. This training part consists of some steps. Among them, the first step is pre-processing.

4.8.1 Pre-Processing

In the preprocessing phase every binary image has to be resized into 425 X 125 pixels. Reducing the image size to 425 X125 decreases processing time and space.

4.8.2 Converting Binary Image to Column Vector

From above we acquired the binary image which has size of (200, 200). So we can consider the binary image as matrix having 200 rows and 200 columns. The matrix holds the binary value of 1 and 0. Precisely, we can say that the matrix stores binary value of 1 where the part of human body is found. Otherwise, it stores 0. Now we have to compute the value by iterating every column of each row. It means every row has 200 columns containing binary value. As the value is binary, we need to convert them into decimal value. In order to do so, we use the formula of binary to decimal conversion given below:

Row1=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

|

|

Row200=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

Now we need to store the value. To do so, we created a matrix having size of (200, 1) which has exactly same rows of the binary image. Actually, this is the column vector which is store the decimal value of each row. After calculating row1, the value was stored at column vector of (1, 1). After calculating row2, the decimal value was stored in column vector of (2, 1) and so on. This is how the column vector stored the decimal value of a binary image. Since we trained 50 images, we got 50 column vectors. After that we saved all column vectors in a global array. After training, we saved them in the computer hard drive. This value is needed when compared them with the testing column vector.

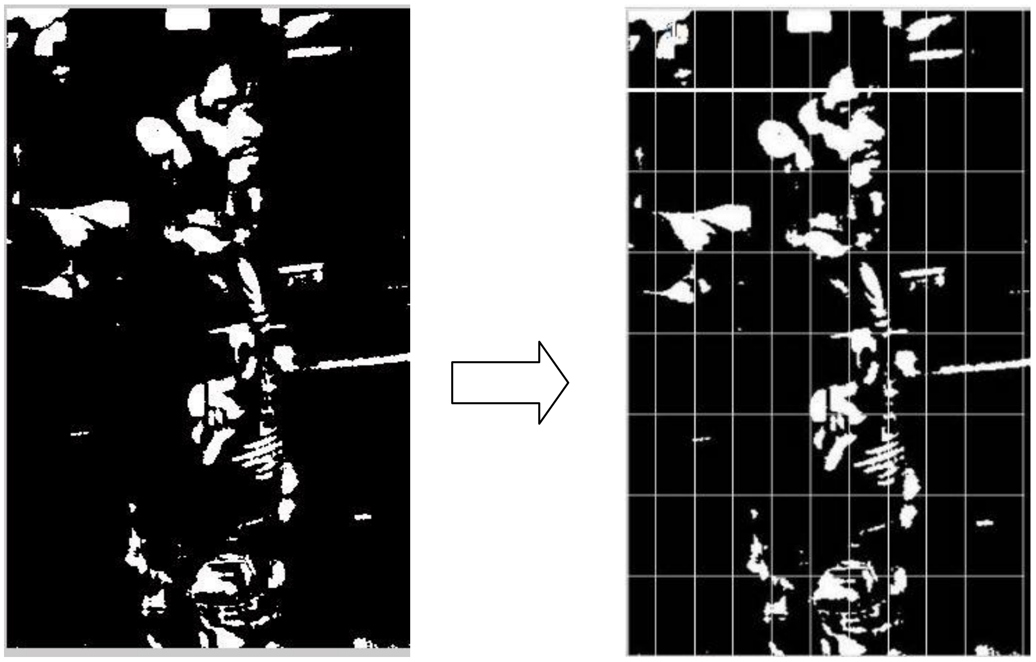


Figure 4.2.3: Representing Binary image in Matrix form

4.9 Processing with Testing Image

In our experiments, we are going to detect abnormal behavior from the video. So our input file is video which is full of abnormal behavior with different poses. As we did not find any universal data set, we made our own video as an input file. After reading a frame from input video, it made a decision if the frame contained abnormal behavior or not. It abnormal behavior exists, it gave us warning. In other case, it just ignored the frame and checks the next frame. Processing the testing image consists of some steps.

4.9.1 Create System Objects

Create System object is used to read frames from video, detecting foreground objects and displaying results. At first, we needed to initialize the video I/O. After that, we needed to create objects from reading video from a particular file. After that, we enabled to read every frame from video file while video was running. Since we wanted to see the video, we created two video players. First one was to display the original video while another was to display the foreground mask. Then we needed to create system objects for foreground detection and blob analysis.

4.9.1.1 Create System Objects For Foreground Mask

The foreground detector is used to separate moving objects from the background. It returns a result which is a binary mask. Here the detector returns result based on Gaussian Mixture Model (GMM). In binary mask, the pixel value sets to be 1 (white) corresponding to the foreground. On the other hand, the pixel value sets to be 0 (Black) corresponding to the background.

In order to create Foreground detector we needed to pass some value as an argument. Here the number of Gaussian modes is 3 in the mixer model, number of training frame is 40 and minimum background ratio is set to be 0.7. This is the standard argument for the Foreground detector.

4.9.1.2 Create System Objects For Blob Analysis

The blob analysis system object is used to draw a shape into the foreground detected objects. In order to draw the shape, it maintains a bunch of rules. First of all, it finds out in foreground objects how many pixels are gathered together. In other words, how many foreground pixels are connected with each other and form a group. Basically, the blob analysis system object is used to find out such groups. Foreground pixel contains the binary value of 1. So it identifies the group of value 1. These pixels are known as connected components. To find out these groups, the system objects emphasize on some characteristics such as area, centroid and the bounding box.

In order to create the blob analysis system objects, some values needed to be passed as arguments. Here Bounding Box Output Port is true, Area Output Port is true, Centroid Output Port is true and Minimum Blob Area is set to be 400.

4.9.2 Read Frame From Video

As we mentioned earlier, we needed to choose a frame manually, where the abnormal behavior exists. In order to it, we ran the video and picked up our desired frame. After that, we sent it to go through other operations. This is how, frames were chosen in our experiments.

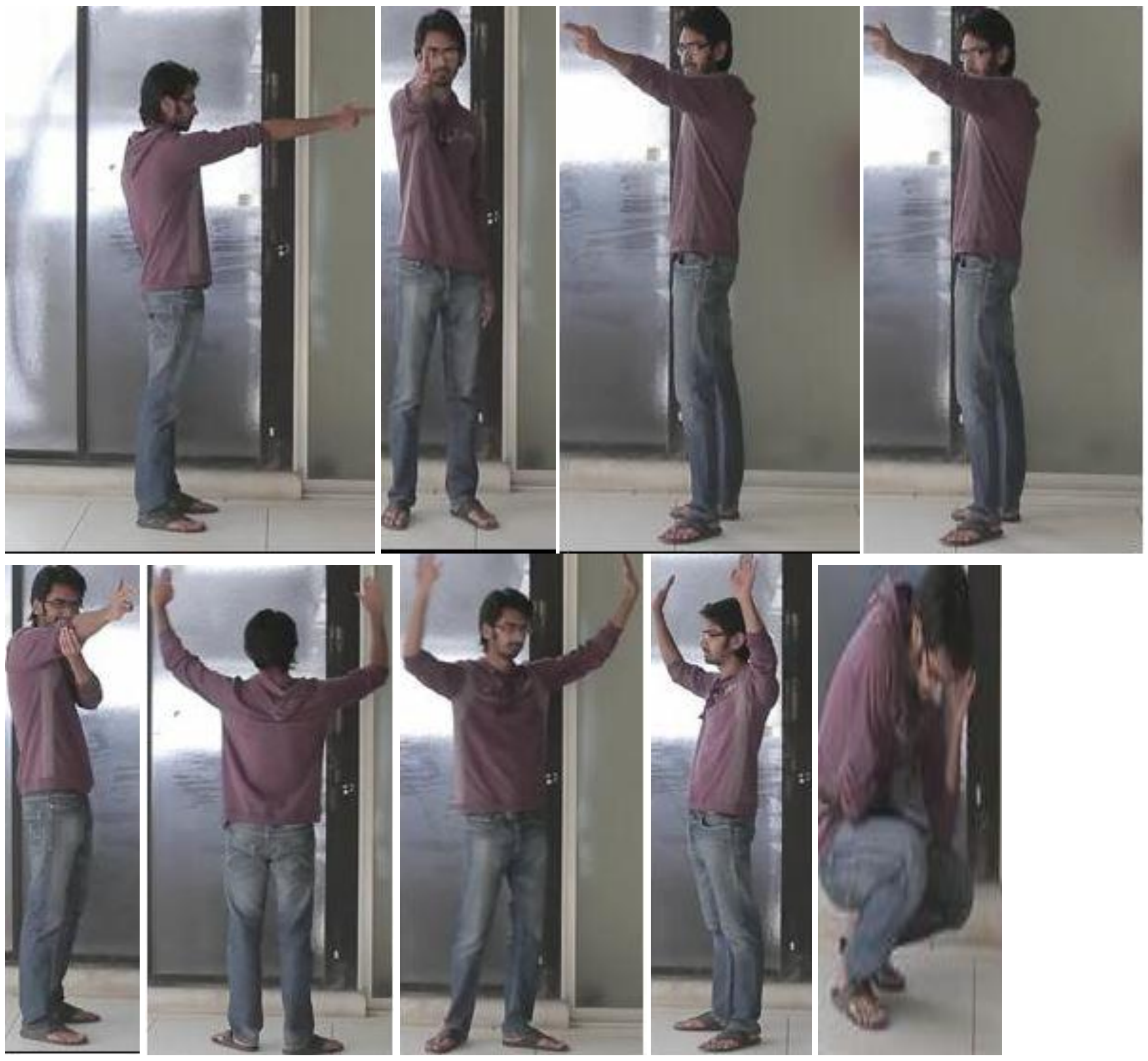


Figure 4.2.4: Sample Images for testing Purpose

4.9.3 Detect Objects From Frame

The function returns the centroids and the bounding boxes of the detected objects. It also returns the binary mask, which has the same size as the input frame. Pixels with a value of 1 correspond to the foreground, and pixels with a value of 0 correspond to the background.

At first, the function performs motion segmentation using the foreground detector. After that it performs morphological operations on the resulting binary mask to remove noise from the pixel and finally fill the holes in the remaining blobs.

4.9.3.1 Detect foreground

We created the Foreground detection system objects in upper step. Now using this object, we detected foreground objects. The System object compares a color or grayscale video frame to a background model to determine whether individual pixels are part of the background or the foreground. It then computes a foreground mask which is also known as binary mask.

The foreground detector requires a number of video frames and minimum background ratio in order to initialize the Gaussian Mixture Model (GMM). Here we used number of frames 40 and Minimum Background ratio 0.7. The idea of choosing these values is to provide us optimum result for every frame.

The foreground segmentation process is not perfect and sometimes includes noise which makes the efficiency lower. In order to remove noise we have to do some operations which described below:

4.9.3.1.1Morphologically Open Image

Morphologically open image function performs morphological opening operation in gray scale image or binary image with the structural elements that we created in above. The morphological open operation consists of two operations. At first, there is an erosion of an image and after that dilation operation is performed.

4.9.3.1.1.1Creating Structural Element

Structural element is an essential part of erosion and dilation. It is mainly used to examine on input image. Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's.The center pixel of the structuring element, called the origin. In our experiment, we create a structural element using which creates a flat, linear structuring element. Here Rectangle is indicated the shape and [3,3] represents a 3 by 3 matrix which actually implements on neighbor’s pixels. Using this element, we performed morphological operation like erosion and dilation.

4.9.3.1.1.2 Eroding Gray Scale Image

This operation erodes the binary image or gray scale image. The created structural element is passed here as an argument. After that the structural element executed over the gray scale image and returns the final eroded image. Here our corresponding image is a binary image, so it performs binary erosion. After erosion operation, the image lost thickness and details of the image.

<u>4.9.3.1.1.3Dilating Gray Scale Image</u>

This operation dilates the binary or gray scale image. The created structural element is passed here as an argument. After that the structural element executes over the gray scale image and returns the final dilated image. Here our corresponding image is a binary image. So it performs binary dilation. It is used to thicken the objects in a proper way. Since after erosion operation, the image lost thickness and details, Dilation operation brings back the image closed to the previous one.

<u>4.9.3.1.2 Morphologically Close Image</u>

This function performs morphologically closing on the gray scale or binary image. At first, we needed to create the structural element. Bu using this elements it performs morphologically image closing. It is used to fill up the gaps of white pixels in binary images.

<u>4.9.3.1.2.1 Creating Structural Element</u>

The structural element is mainly used to examine on input image. . Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's.The center pixel of the structuring element, called the origin. In our experiment, we create a structural element using which creates a flat, linear structuring element. Here Rectangle is indicated the shape and [15, 15] represents a 15 by 15 matrix which actually implements on neighbor’s pixels and provides the desired output.

<u>4.9.3.1.3Fill up The Holes In Binary Image</u>

In this step, we used a function that fills up the holes in binary image. In upper method, we use morphologically close image to fill up the gap of white pixels but unfortunately, sometimes it fails to reach some white pixel. That is why, we used this method.

4.9.3.2 Blob Analysis

Above, we mentioned that we created a blob analysis system object to perform this operation. The operation is performed based on some characteristics like area, centroid and bounding box.

As the functions finds out some groups of connected white pixels, we needed to give a shape on them. To do so, we first saved the result in a variable. After that we created a system object for shape inserter. So now, we used that objects into that variable where the detected human object was stored. The reason of selecting Rectangle shape is, we were interested only o human and rectangle shape was the best way to represent a human rather than using disk or square shape.

Finally, we got a binary image where a human object was detected bounding by a rectangular box and stored in a variable.

4.9.4Crop Human Body From the Binary Image

Now we got a binary image with human. We did not need the whole binary image. Here we observed whether the behavior was abnormal or not. So we only needed human portion. We saved the value of human portion in a variable. In order to get it, we cropped the value from the binary image. The idea of cropping image is to increase processing time and eliminate the noise.

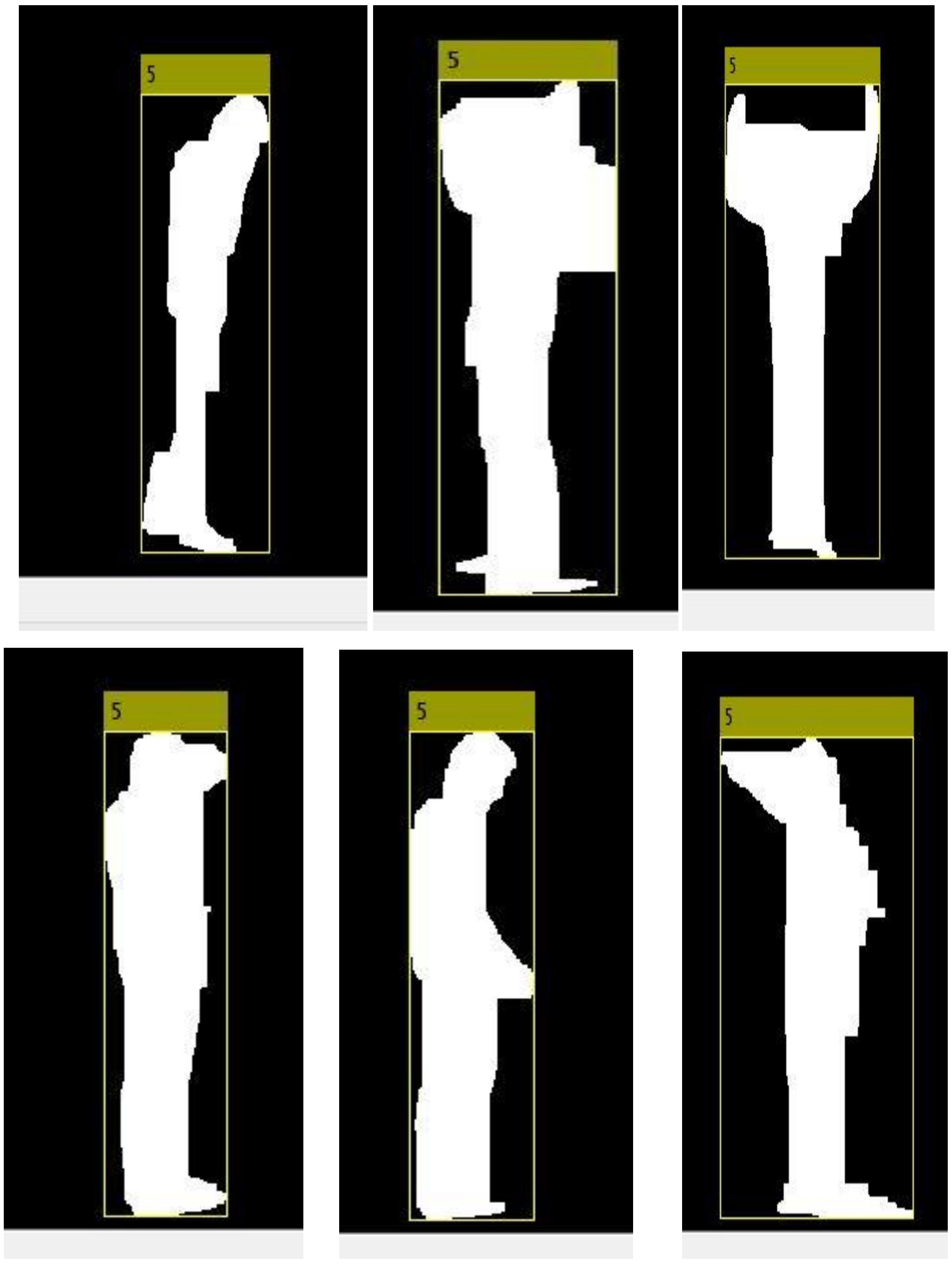


Figure 4.2.5: Binary Image

This part is the most important part of our experiment. In this part, the binary image gets converted to a column vector. The column helps us to determine if the behavior of human in the image is abnormal or not. This part consists of some steps. Among them, the first step is pre-processing.

4.9.5 Pre-Processing

In the preprocessing phase every binary image has to be resized into 200 x 200 pixels. Reducing the image size to 200 x 200 decreases processing time and space.

4.9.6 Converting Binary Image to Column Vector

From above we acquired the binary image which has size of (200, 200). So we can consider the binary image as matrix having 200 rows and 200 columns. The matrix holds the binary value of 1 and 0. Precisely, we can say that the matrix stores binary value of 1 where the part of human body is found. Otherwise, it stores 0. Now we have to compute the value by iterating every column of each row. It means every row has 200 columns having contains binary value. As the value is binary, we need to convert them into decimal value. In order to do so, we use the formula of binary to decimal conversion given below:

Row1=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

|

|

Row200=2^124*val +2^123*val+……………………..+ 2^1*val+ 2^0*val

Now we need to store the value. To do so, we created a matrix having size of (200, 1) which has exactly same rows of the binary image. Actually, this is the column vector which is store the decimal value of each row. After calculating row1, the value was stored at column vector of (1, 1). After calculating row2, the decimal value was stored in column vector of (2, 1) and so on. This is how the column vector stored the decimal value of a binary image.

4.10 Behavior Analysis:

In our dataset, the column vector of each binary image having size of (200, 1) have been stored. Now, the test image is also converted into the binary image. Then we compute the binary image into a column vector with same size of trained column vector (200, 1). Now we find out which trained image is very close enough to the test image. In order to so, we need to calculate the Euclidian distance between test column vector and trained column vector.

4.10.1 Euclidian Distance

Euclidian distance is the ordinary distance between two points. It shows how far two points are located from each other. The Euclidian distance formula is given below:

The **Euclidean distance** between point **p** and **q** is the:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

In Matlab, we use the formula which given below:

D = sqrt (sum ((G - G1). ^ 2))

Here G is the column vector of test image and G1 is the column vector of train image. The explanation of Euclidian distance formula is to subtract two points by element wise and then adding them up by squaring the subtracting the result. After adding all results, finally we perform square root over the result and the final result is our desired Euclidian distance.

The Euclidian distance is calculated by drawing two curves in graph. One curve for test column vector and another one is for trained column vector. Below a graph is shown

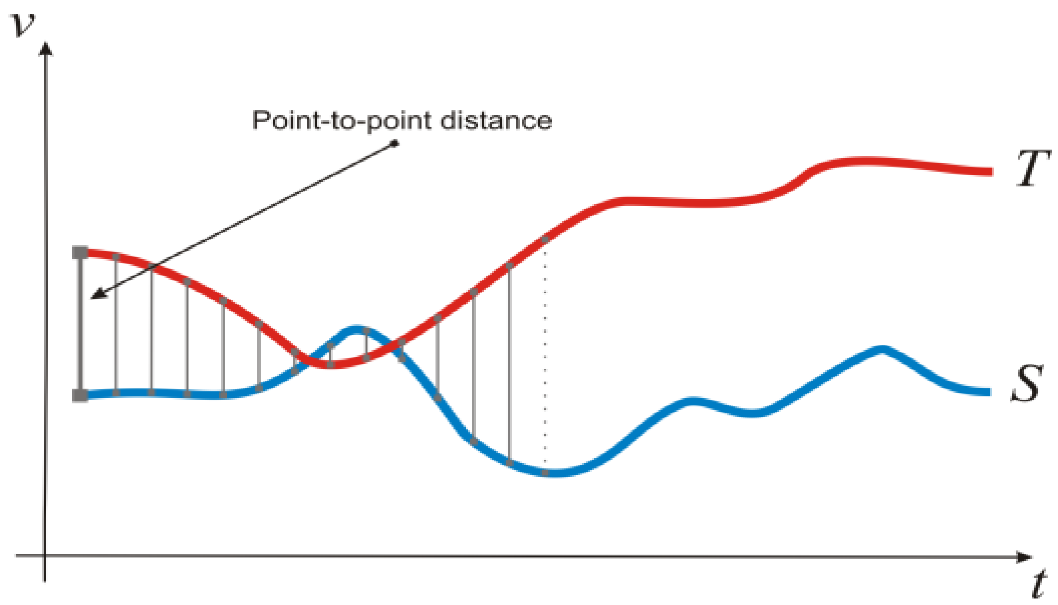


Figure 4.2.6: Euclidian Distance graph

T and S represent the curves of two column vectors. Between each pair of points, a line is drawn to indicate the point to point distance. After drawing the line, the Euclidian formula is used. This methodology will be followed for every pair of test and trained column vector. Then we store the result in an auxiliary array. The result shows that how two vectors are close to each other. It means it shows the similarities or between two vectors. In other words, we can say that the result tells us the difference between two column vectors. Now we find out the minimum value of an array. The minimum value indicates that the result of column vector is the best match or gets the highest similarity compared with test column vector. If the result is very close to 0 or exactly 0, then we can say that, the two column vectors are same. Hence, it is obvious that the test image and trained image is exactly same. As we only train the abnormal behavior of human, we can say that the test image contains abnormal behavior and applicable to draw further attention.

5. Experimental Result & Analysis

Chapter 1

Frame	Abnormal Behavior	Condition	Experimental Result
1.	True	Sunny	✓
2.	True	Sunny	✓
3.	True	Normal Daylight	✗
4.	True	Low Light	✗
5.	True	Normal Daylight	✗
6.	True	Normal Daylight	✓
7.	False	Low Light	✗
8.	True	Low Light	✓
9.	False	Sunny	✓
10.	True	Sunny	✓
11.	True	Sunny	✓
12.	False	Normal Daylight	✓
13.	False	Normal Daylight	✗
14.	True	Normal Daylight	✓
15.	True	Sunny	✗
16.	True	Low Light	✗
17.	False	Low Light	✓

18.	True	Sunny	✓
19.	True	Sunny	✓
20.	True	Sunny	✓
21.	True	Normal Daylight	✓
22	True	Normal Daylight	✗
23	False	Low Light	✗
24	True	Low Light	✗
25	False	Sunny	✓
26	True	Normal Daylight	✓
27	True	Low Light	✗
28	False	Low Light	✓
29	True	Normal Daylight	✗
30	True	Normal Daylight	✓
31	True	Low Light	✗
32	True	Sunny	✓
33	True	Sunny	✓
34	True	Sunny	✓
35	False	Sunny	✗
36	True	Low Light	✓
37	False	Normal Daylight	✓
38	True	Normal Daylight	✓

39	False	Low Light	✓
40	True	Low Light	✗
41	False	Normal Day Light	✓
42	True	Sunny	✓
43	True	Sunny	✓
44	True	Sunny	✓
45	True	Normal Daylight	✗
46	False	Low Light	✓
47	False	Low Light	✓
48	True	Sunny	✗
49	False	Low Light	✓
50	True	Sunny	✓

Table2: Experimental result of Abnormal Behavior Detection Using Viola Jones Algorithm

<u>Chapter2</u>

Frame	**Abnormal Behavior**	**Condition**	**Experimental Result**
1.	True	Sunny	✓
2.	True	Sunny	✓
3.	True	Normal Daylight	✓
4.	True	Low Light	✓
5.	True	Normal Daylight	✓

6.	True	Normal Daylight	✓
7.	False	Low Light	✓
8.	False	Low Light	✓
9.	True	Sunny	✗
10.	True	Sunny	✓
11.	False	Sunny	✓
12.	False	Normal Daylight	✓
13.	True	Normal Daylight	✗
14.	True	Normal Daylight	✓
15.	True	Sunny	✗
16.	True	Low Light	✗
17.	False	Low Light	✓
18.	True	Sunny	✓
19.	True	Sunny	✓
20.	True	Sunny	✓
21.	True	Normal Daylight	✓
22	True	Normal Daylight	✗
23	False	Low Light	✓
24	True	Low Light	✗
25	False	Sunny	✓
26	True	Normal Daylight	✓

27	True	Low Light	✓
28	False	Low Light	✓
29	True	Normal Daylight	✓
30	True	Normal Daylight	✓
31	True	Low Light	✓
32	True	Sunny	✓
33	True	Sunny	✓
34	True	Sunny	✓
35	False	Sunny	✗
36	True	Low Light	✓
37	True	Normal Daylight	✓
38	False	Normal Daylight	✓
39	True	Low Light	✓
40	True	Low Light	✗
41	False	Normal Day Light	✓
42	True	Sunny	✓
43	True	Sunny	✓
44	True	Sunny	✓
45	False	Normal Daylight	✗
46	False	Low Light	✓
47	False	Low Light	✓

48	True	Sunny	✓
49	False	Low Light	✓
50	True	Sunny	✓
51.	True	Sunny	✓
52.	True	Sunny	✓
53.	True	Normal Daylight	✓
54.	True	Low Light	✓
55.	True	Normal Daylight	✓
56.	True	Normal Daylight	✓
57.	True	Low Light	✓
58.	True	Low Light	✓
59.	False	Sunny	✗
60.	True	Sunny	✓
61.	True	Sunny	✓
62.	False	Normal Daylight	✓
63.	False	Normal Daylight	✓
64.	True	Normal Daylight	✓
65.	True	Sunny	✗
66.	True	Low Light	✓
67.	False	Low Light	✓
68.	True	Sunny	✓

69.	True	Sunny	✓
70.	True	Sunny	✗
71.	True	Normal Daylight	✓
72	False	Normal Daylight	✓
73	True	Low Light	✓
74	True	Low Light	✓
75	False	Sunny	✗
76	True	Normal Daylight	✓
77	True	Low Light	✗
78	False	Low Light	✓
79	True	Normal Daylight	✓
80	True	Normal Daylight	✓
81	True	Low Light	✓
82	True	Sunny	✓
83	True	Sunny	✓
84	True	Sunny	✓
85	False	Sunny	✗
86	True	Low Light	✓
87	False	Normal Daylight	✓
88	True	Normal Daylight	✓
89	False	Low Light	✓

90	True	Low Light	✓
91	False	Normal Day Light	✓
92	True	Sunny	✓
93	True	Sunny	✓
94	True	Sunny	✓
95	False	Normal Daylight	✓
96	True	Low Light	✓
97	False	Low Light	✓
98	True	Sunny	✓
99	True	Low Light	✓
100	True	Sunny	✓

Table3: Experimental result of Abnormal Behavior Detection Using Foreground Detection Approach

6. Accuracy Calculation

<u>For Chapter1</u>

Test images based on Face detection approach		
Frame Type	Test Frame	Correct Output
Frames With Normal Behavior	15	11
Frames With Abnormal Behavior	35	22
Total	50	33

Table 4

Test Frame Classification based on Different Atmosphere For Face Detection Approach		
Image Environment	**Test Image**	**Correct Output**
Sunny	19	16
Low Light	16	8
Normal Daylight	15	9
Total Image	50	33

Table 5

Accuracy for Abnormal behavior using Face Detection Algorithm:

$$\frac{\textbf{Correct Output}}{\textbf{Test Frame}}\times 100\%$$

$$=\frac{22}{35} \text{ x } 100\%$$

$$= 62.86\%$$

Accuracy for normal behavior using Face Detection Algorithm:

$$\frac{\textbf{Correct Output}}{\textbf{Test Frame}}\times 100\%$$

$$=\frac{11}{15} \text{ x } 100\%$$

$$= 73.33\%$$

Total Accuracy using face detection Algorithm:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{33}{50} \text{ x } 100\%$$

$$= 66\%$$

<u>Accuracy Level in Different Environment:</u>

Sunny Condition:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{16}{19} \text{ x } 100\%$$

= 84.21%

Low Light Condition:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{8}{16} \text{ x } 100\%$$

= 50%

Normal Day Light Condition:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{9}{15} \text{ x } 100\%$$

= 60%

<u>For Chapter2</u>

Test Frames Based on Foreground Detection		
Frame Type	Test Frame	Correct Output
Frames With Normal Behavior	30	25
Frames With Abnormal Behavior	70	60
Total	100	85

Table 6

Test Frame Classification based on Different Atmosphere For Foreground Detection Approach		
Image Environment	**Test Image**	**Correct Output**
Sunny	38	30
Low Light	32	28
Normal Daylight	30	27
Total Image	100	85

Table 7

Accuracy for Abnormal behavior using Foreground Detection Algorithm:

$$\frac{Correct\ Output}{Test\ Frame} \times 100\%$$

$$= \frac{60}{70} \text{ x } 100\%$$

$$= 85.71\%$$

Accuracy for normal behavior using Foreground Detection Algorithm

$$\frac{Correct\ Output}{Test\ Frame} \times 100\%$$

$$= \frac{25}{30} \text{ x } 100\%$$

$$= 83.3\%$$

Total Accuracy using Foreground Detection Algorithm

$$\frac{Correct\ Output}{Test\ Frame} \times 100\%$$

$$= \frac{85}{100} \text{ x } 100\%$$

$$= 85\%$$

Accuracy Level in different Environments:

Sunny condition:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{30}{38} \text{ x } 100\%$$

= 78.95%

Low Light Condition:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{28}{32} \text{ x } 100\%$$

= 87.5%

Normal Day Light Condition:

$$\frac{Correct\ Output}{Test\ Frame}\times 100\%$$

$$=\frac{27}{30} \text{ x } 100\%$$

= 90%

7. Comparison

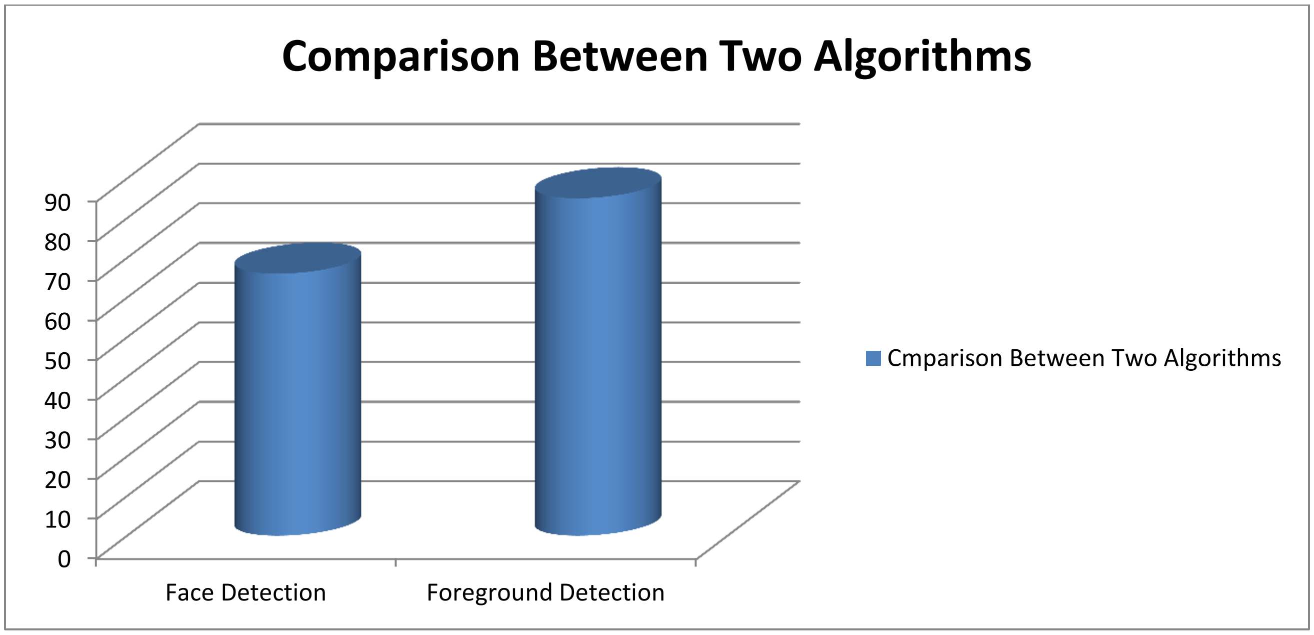


Figure 8.1: Accuracy level comparison between two approaches

In our experiments, the system using face detection (Viola Jones) algorithm achieved 66% accuracy whereas the system using foreground detection achieved 85% accuracy.

<u>Comparison in Different Environments:</u>

<u>Sunny Condition</u>

In the sunny condition the foreground detection approach is giving less accuracy than the face detection approach. Because in the sunny condition the human has a dark shadow which creates a problem for detecting human accurately.

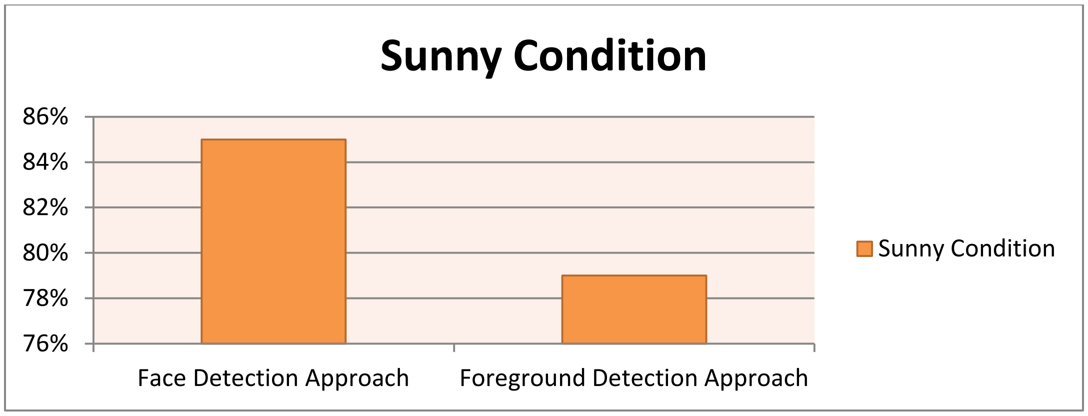


Figure8.2: Comparison of two approaches in sunny conditions

Normal Daylight Condition:

In the normal daylight the accuracy level of Foreground detection approach is much higher than the Face detection approach. The comparison is shown in bar chart below:

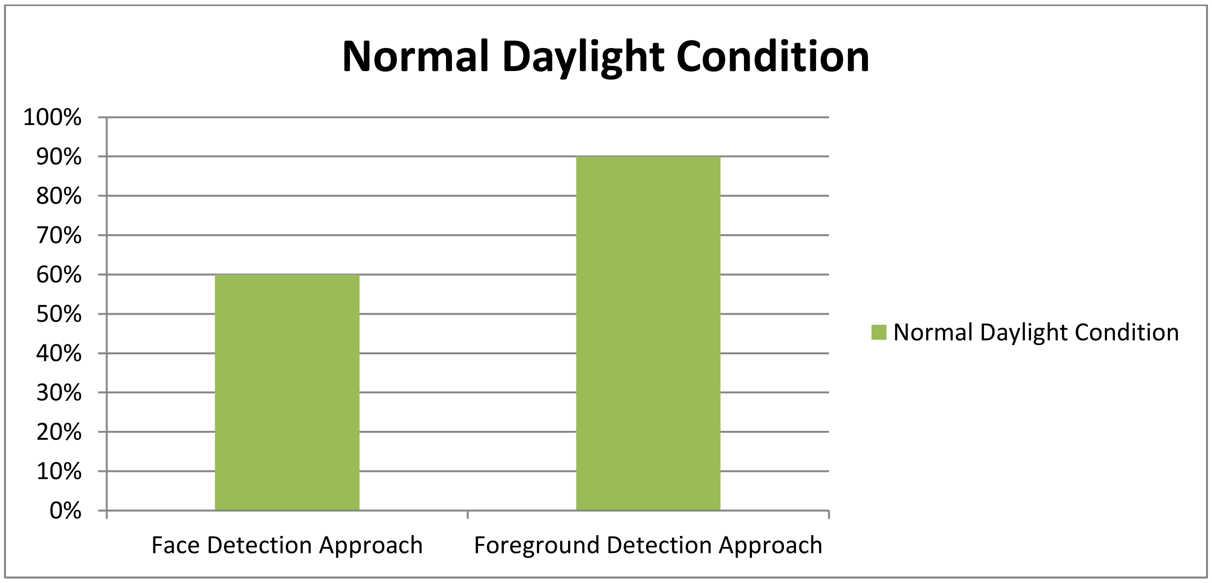


Figure8.3: Comparison between two approaches in Normal Daylight Condition

Lowlight Condition:

The Foreground approach has much higher accuracy than the Face detection Approach in lowlight condition. The comparison in bar-chart is given below:

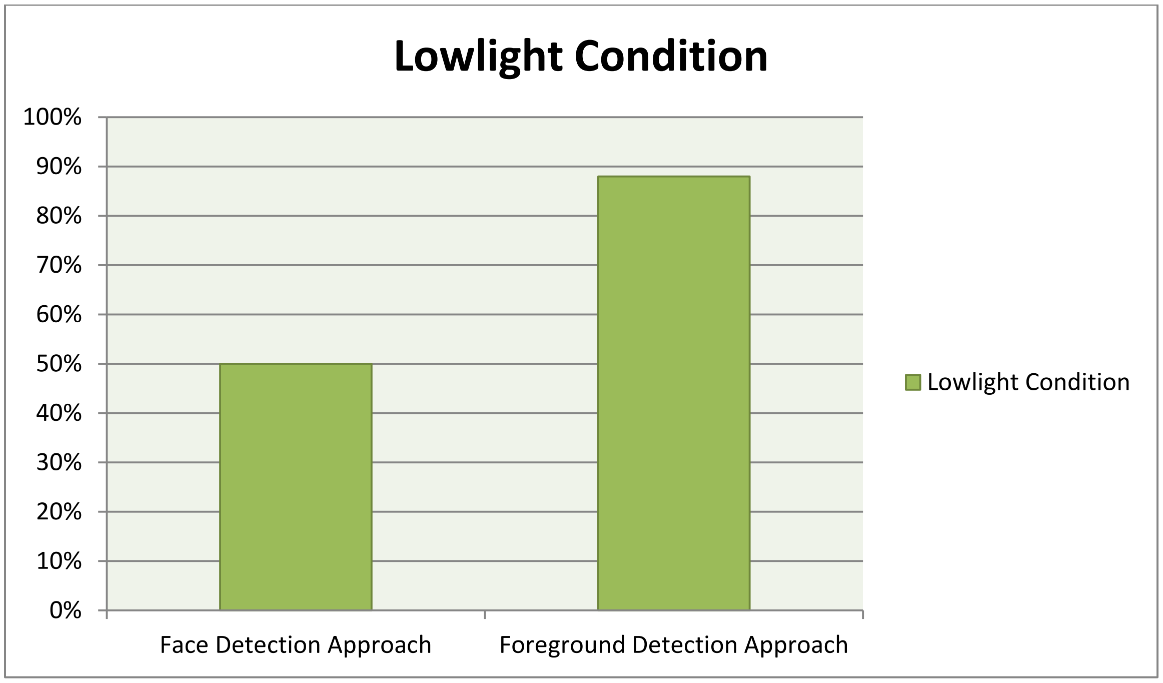


Figure8.4: Comparison between two approaches in Lowlight Condition

8. Conclusion and Future Work

8.1 Conclusion

In this paper, we proposed an automated video surveillance system which can analyze the behaviors of human and detect the abnormal behaviors. In “chapter 1” we applied Face Detection method for detection purpose. Nevertheless, the accuracy we got by applying this approach was much less. Therefore, we moved to another approach which provided us much better result than the previous approach. The approach is ‘Foreground Detection Approach’. This approach uses GMM to extract the human portion from the background and later on using morphological image processing we converted the RGB image into binary image. Our proposed algorithm gave efficient results and a decent accuracy.

8.2 Future Work

In our next step of research, we plan to detect abnormal behaviors in real time. We are planning to create a connection between the surveillance system and a device which will be controlled by the operator in charge of the system. When our system will find any abnormal behavior it will immediately send the detected steaming part to the connected device. Thus the operator or the user who is in charge of that will be notified instantly. As a result, the operators don’t have to monitor the surveillance videos all the time. By using this kind of intelligence system the necessity of manpower can be reduced.

References

1. B. Ommer, T. Mader, and J. M. Buhmann. Seeing the objects behind the dots: Recognition in videos from a moving camera. *Int. J. Comput. Vision*, 83:57–71, June 2009. (ref-18)
2. H. Dee and D. Hogg. Detecting inexplicable behavior. In *BMVC*, 2004.
3. V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *CVPR*, 2010.
4. H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *CVPR*, pages 819–826, 2004.
5. T. Xiang and S. Gong. Video behaviour profiling and abnormality detection without manual labelling. In *ICCV*, pages 1238–1245, 2005.
6. P. Remagnino and G. A. Jones. Classifying surveillance events from attributes and behaviour. In *BMVC*, 2001.
7. X. Wang, X. Ma, and W. Grimson. Unsupervised activity perception in crowded and bayesian models. *PAMI*, 31, 2009.
8. J. Kim and K. Grauman. Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates. In *CVPR*, 2009.
9. A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixedlocation monitors. *PAMI*, 30, 2008.
10. R. Mehran, A. Oyama, and M. Shah, “Abnormal crowd behavior detection using social force model,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.
11. M. D. Breitenstein, H. Grabner, and L. V. Gool, “Hunting Nessie – realtime abnormality detection from webcams,” in IEEE International Workshop on Visual Surveillance, 2009.
12. Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins "Digital Image Processing Using MATLAB® (Second Edition)" A Division of Gatesmark, LLC, 2009.
13. Wan NajwaBinti Wan Ismail, Faculty of Electrical & Electronics Engineering University Malaysia Pahang, MAY, 2009.
14. Ole Helvig Jensen, Technical University of Denmark Informatics and Mathematical Modeling, “Implementing the Viola-Jones Face Detection Algorithm”, 2008.