5-1-2013

Real Time Pattern Recognition in Digital Video with Applications to Safety in Construction Sites

Dinesh Bajracharya
University of Nevada, Las Vegas, dineshbaj@gmail.com

REAL TIME PATTERN RECOGNITION IN DIGITAL VIDEO WITH APPLICATIONS TO SAFETY IN CONSTRUCTION SITES

by

Dinesh Bajracharya

Bachelor's Degree in Computer Engineering
Pulchowk Campus, Institute of Engineering
Tribhuvan University, Nepal
2007

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2013

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Dinesh Bajracharya

entitled

Real Time Pattern Recognition in Digital Video with Applications to Safety in Construction Sites

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Evangelos Yfantis, Ph.D., Committee Chair

Laxmi Gewali, Ph.D., Committee Member

John T. Minor, Ph.D., Committee Member

Peter Stubberud, Ph.D., Graduate College Representative

Tom Piechota, Ph.D., Interim Vice President for Research &
Dean of the Graduate College

May 2013

ABSTRACT

Real Time Pattern Recognition In Digital Video With Applications To Safety In Construction Sites

by

Dinesh Bajracharya

Dr. Evangelos A. Yfantis, Examination Committee Chair

Professor, Department Of Computer Science

University of Nevada, Las Vegas

In construction sites, various guidelines are provided for the correct use of safety equipment. Many fatalities and injuries occur to people because of the lack of exercise of these guidelines and proper monitoring of the violations. In order to improve these standards and amend the cause, a video based monitoring tool will be created for a construction site.

Based on the real time video obtained from cameras on the site, a classification algorithm will be created which has the intelligence to recognize if any safety rules have been violated. A classification vector will be created based on the different classifiers, depending on the properties of the object and the image, to construct a classifier to classify a construction site for a safe state. If any safety rule is being violated the algorithm issues a real time alarm event making the management aware of the violations. The steadiness of the system is indicated by the probability of being in a safe state.

ACKNOWLEDGEMENTS

"I would like to express my inmost gratitude to my supervisor, Dr. Evangelos A. Yfantis, for his guidance, invaluable support, motivation and encouragement throughout the period of this thesis work. His readiness for consultation at all times, his knowledgeable comments, and innovative ideas have been invaluable throughout the academic period.

I would like to thank Dr. Laxmi P. Gewali for his academic guidance throughout the period and also joining in my thesis committee. I would also like to thank Dr. John T. Minor and Dr. Peter A. Stubberud for their help and guidance as the thesis committee members.

I would like to thank my coworker Nishikar Sapkota for his support and effort provided in the beginning phases of the thesis to build a video transmission tool. I would also like to thank all of my friends who have supported me to complete my thesis and been with me throughout my academic period.

Finally and most importantly, I would like to express deepest gratitute to my wife Anjana Shakya and my family members for their encouragement, love and support without which this thesis wouldn't have been possible."

Dinesh Bajracharya

University of Nevada, Las Vegas
May 2013

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Construction sites are considered to be one of the most dangerous land-based work sectors all around the world [1]. Accidents and injuries are inevitable in construction zones [2]. These accidents can be quite costly as well as can hurt the morale of the workers and contractors [2]. Various safety measures are taken in these sites to minimize these accidents and injuries, and to create a zero injury culture in the construction industry [2]. Various organizations give safety training to people to reduce the injuries and to monitor the causes [3].

Fatal injuries in the construction industry can mainly be categorized into i) fall accidents ii) electrocutions iii) struck by objects iv) others [4]. In 2011, Bureau of Labor Statistics (BLS) reported 721 fatalities in US. Of these, 254 (35%) were due to fall accidents and 123 (17%) were due to objects [4]. From this observation, this number can be reduced by about 52% if the use of safety equipment are enforced properly, and produce zero injury zone.

Different types of safety equipment like hard-hats, safety vests, safety footwear and hearing protection have been made mandatory while working in the construction zone [3]. Even though these restrictions have been applied to the field, many incidents have been found where people were breaking the rules of restrictions, which were the causes for the fatal death in numerous occasions. So, the supervisors have to take extra caution towards the use and monitoring of these equipment in the sites.

However, due to responsibilities and tight work schedule, supervisors are not able to monitor each and every worker on the site [2]. To overcome this deficiency, various research on automated systems for detection of safety equipment have been taking place [2]. To support this cause, a video-based monitoring tool is being developed that will alert the supervisor of any violations taking place in the sites.

In this research, the main focus is on the development of a real-time video monitoring system

based on pattern recognition. The real-time video is captured with the help of one or more cameras which are set up in the construction site. These cameras capture high quality uncompressed analog electrical signals which are converted into digital data with the help of a processor in the camera. The converted data is compressed and then transferred to the main server for processing.

The server analyzes the compressed data and converts the digital video into frames of digital image. Each video is converted to 30 frames/second. An edge detection algorithm is applied to each frame to get the edges from the image. Then from these edges, the image is segmented into different Video Object Planes (VOP) consisting of 1 person in each VOP and then applying the classifier to identify people from the image. After breaking down the images into VOPs, a face detection algorithm is applied to segment the image to head and body parts.

After the head region in the image is detected, the segments of the upper part of the head can be found just above the face where a hard hat most likely resides and the lower part of the body above the waist where the jacket is worn. The upper region is analyzed for the detection of the hard hat by using the properties of a hard hat, which is mostly comprised of two orthogonal semicircles. Structure and color of the hat are used to create the classifier to differentiate the hard hat from a normal hat and a head with no hat as well. Similarly, the lower region can be analyzed to detect the presence of a jacket. The presence or absence of this safety equipment can be used to create the terms for a safety violation. Different frames are analyzed to get the accurate reading for the violation. In any case, if a violation is observed by the system, an alarm event is generated which is transferred to the supervisor to take immediate action.

Furthermore, the implementation can be improved not only to just trigger the alarm event, but also to detect the exact location of the violation, people violating the rules, time and duration of the violation, and history of the violation by each person by analyzing the data. Time, duration and point of violation can also be detected from the history of video in the database. Along with that the person violating the rules can also be identified by using different properties of that person that has already been stored in the database like the facial features, height, width, gait of walking and other various factors. This can not only monitor the safety violations but also minimize the fatalities caused due to these violations.

1.1 Goals and Scope

The major objectives of this thesis are:

1. To monitor the proper use of safety equipment in a construction site with the help of video pattern recognition.
2. To develop an algorithm to transfer the real time video from the construction site to the processing server and from server to the client.
3. To develop an algorithm to check and segment the safety equipment from the image if they exist.
4. To trigger an alarm for any safety violations.

The scope of this study is limited to monitoring safety equipment in the lab with a given set of standards. This technique can be used in a construction site to check for safety violations. It does not identify the person involved in the violation and the location of the violation. These can be incorporated in later versions.

CHAPTER 2

BACKGROUND

From our observation, safety in construction sites is a very important part for reducing the number of fatalities. Different standards for the mandatory use of certain equipment have been applied according to the safety rules in different safety sites. Fig. 2.1 shows different equipment such as hard-hats, safety vests and safety footwear used in a construction site.

Figure 2.1: A construction site showing different equipment such as hard-hats, vests and footwear

Due to violations of these rules, there have been various cases of fatalities and injuries which could have been avoided if safety procedures had been followed correctly [2]. Supporting this reason, an

automated real time monitoring system using pattern recognition is being developed in this thesis. In this chapter, the basic concepts that have been used will be introduced alongwith the review of the different techniques that exist.

2.1 Digital Image

A digital image is a numeric representation of a two-dimensional image [5]. The term digital image usually refers to raster images with fixed resolutions, also called bitmap images. Bitmap is the type of memory organization used to store digital images. A bitmap image takes the form of an array, where the value of each element, called a **pixel** picture element, corresponds to the color of that portion of the image. A 1 bit bitmap representation can store 2 colors representing white and black colors in an image.

In case of color images, an 8 bit RGB (red, green and blue) model is used. Each pixel is represented by three RGB values, using 8 bits which results in 2^8=256 different values for each color. For example, white is represented as (R, G, B) = (255, 255, 255). Similarly yellow: (R, G, B) = (255, 255, 0), cyan: (R, G, B) = (0, 255, 255). In other words, color images are two dimensional arrays of color values of pixels, each of which is coded in 3x8 bits (3 bytes) of RGB. This allows the image to contain a total of 256x256x256 = 16.8 million different colors. This technique is also known as RGB encoding, and is specially adapted to human vision. Cameras are capable of capturing thousands of values for each color, but 256 are enough for the human vision and this also reduces the amount of memory that is needed to store images of color representation.

2.2 Digital Video and Compression

Digital video comprises of a series of orthogonal bitmap digital images displayed in rapid succession at a constant rate. In the context of video these images are called frames. The rate at which frames are displayed is measured in frames per second (FPS). Hence, digital video is a sequence of frames. If a 1 minute video is considered, with 640x480 digital image frame with 3 byte RGB encoding at a rate of 30 FPS, then 640x480x3x30x60 = 1,658,880,000 bytes = 1.55 GB of storage is needed. Thus a very huge amount of storage is needed for just 1 minute of video.

Due to this storage overhead, most videos are compressed before storing into the system. The most commonly used technique for compressing an image is JPEG and for video is MPEG which are both lossy compressions, as losing some of the frames or some pixel values generally does not

degrade the quality of the output. This compression is even more necessary when data need to transfered from one place to another through the internet or other mediums. Mostly in the modern era, digital cameras already have built in hardware to compress videos to the standard format so that there is no problem while transmitting huge amount of data from one place to another.

2.3 Grayscale

Grayscale of a digital image is an image in which the value of each pixel is a single sample; that is, it carries only intensity information. The images of this type look like a black-and-white images with shades of different intensities of gray. The grayscale image of the color image can be computed by either using all the RGB channels or just using one of them. Mostly all the channels are mixed for computing the gray scale. Fig. 2.2 shows grayscale computation with different colors. The formula used for computing the grayscale for each pixel is:

$$Y = 0.299R + 0.587G + 0.114B$$

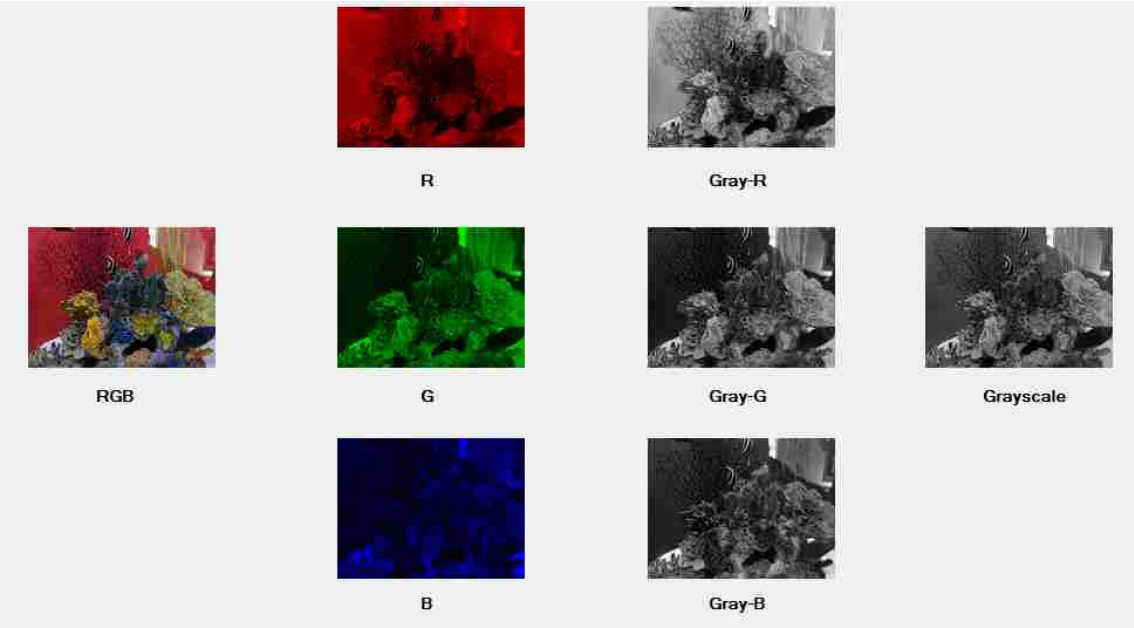

Figure 2.2: Gray Scale computed from different colors

While analyzing images in image processing, mostly a grayscale image is used. Grayscale only contains 255 colors compared to 16.8 million colors in color images. This makes processing easy,

faster to analyze, less effect due to noise and makes our computation very simple.

2.4 Image Segmentation

The goal of image segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects [6]. Segmentation could be used for object recognition, occlusion boundary estimation within motion or stereo systems, image compression, image editing, medical imaging or image database look-up. Segmentation is one of the first steps in image analysis. There are many segmentation techniques based on the discontinuity of the pixels as well as similarity of the regions. Although image segmentation approaches have been greatly developed, there aren't perfect solutions to many problems because of its complexity. The quality of the segmentation techniques also depends on the type of the image. Some of these techniques are described below [7].

2.4.1 Thresholding Methods

The simplest and most widely used segmentation technique is the thresholding method. It converts the multilevel image into a binary image, with only two values 0 (background) and 1 (objects), by using a threshold value. It is mostly applicable for images with bimodal histograms as shown in Fig 2.3.

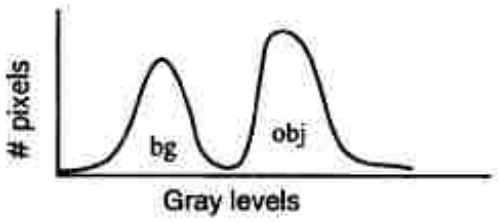

Figure 2.3: A bimodal histogram. One mode represents background pixel and other object pixel

There is a fixed threshold level T that separates the background area from the objects. The two most popular methods are Gaussian filtering and Otsu's method based on discriminant analysis. Mostly using only one threshold cannot result in a better result, in that case more than one thresholding can be used. But it is very hard to determine the thresholds for very complex structures with lots of background variance.

2.4.2 Region Growing Methods

The basic idea of region growing is to use image characteristics to map individual pixels in an input image to sets of pixels called regions that might correspond to an object or a meaningful part of one. Each of the pixels in a region is similar with respect to some characteristics or computed property, such as color, intensity and/or texture.

First, a seed pixel need to be found as a start point for each of the needed segmentation. And then merge the same or similar property of pixels with the seed pixel around the seed pixel domain into the domain of the seed pixel. Then using a new seed pixel the same process of finding other new pixels is repeated in the domain. Finally, the resulting image contains the regions of similar characteristics. Effectiveness of this technique is dependent on the selection of the seed as well as the complexity of the image variations.

2.4.3 Edge Detection Methods

Edge-based segmentation is based on the location of pixels in the image that corresponds to the boundaries of the objects seen in the image. After applying edge detection algorithm, the image is converted into the binary image with the boundary of the edges. If there are discontinuities in the boundaries then the closely related edges are connected to form longer and complete boundaries.

The disadvantage of this approach is that edges are not guaranteed to form closed boundaries and are very susceptible to noise. Edge detection is a well-developed field on its own within image processing. Region boundaries are closely related to the edge of the object, where there is sharp transition of the intensity. Due to this, edge detection has been used as the base of many segmentation techniques. Edge detection will be used for segmentation in this thesis.

2.5 Edge Detection

Edges are the boundary of the object. Edges are the places where something changes sharply or has some discontinuity in the image. The purpose of detecting sharp changes in an image is to capture important events and changes in properties of the image. Changes are measured in the first derivative and the biggest changes have maximum first derivate OR zero second derivative. Edges in an image may be due to various factors as shown in Fig 2.4.

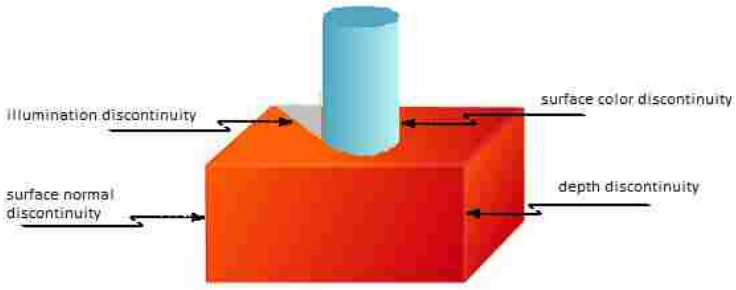

Figure 2.4: Various factors of edges

Edge detection [8] is one of the fundamental steps in image analysis. In the ideal case, results of applying an edge detector to an image will lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation. Thus, applying an edge detection algorithm to an image will significantly reduce the amount of data to be processed and therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. However, it is very difficult to find ideal edges in real life images. There can be missing edges and false edges that can mislead our study. So the use of an accurate edge detection algorithm is very necessary. Some of the algorithms [9] studied for the cause are described below.

2.5.1 Gradient Edge Detection

Gradient edge detectors are one of the oldest edge detection techniques. These are based on the gradient magnitudes, which are the first order derivative in x and y directions. It simply calculates the gradient magnitude and a threshold is applied to get the edges. These are very simple to apply but are sensitive to noise. Most common gradient operators used are Prewitt, Sobel and Roberts operators.

2.5.2 Marr-Hildreth Edge Detection

Marr and Hildreth developed an edge detection technique based on the second order derivative of the image, called a Marr-Hildreth edge detector. This is also called Laplacian of Gaussian because it uses the laplacian, second order derivative, of the Gaussian filter. A Gaussian filter is a filtering technique to reduce the noise in an image. In this technique, the second order derivative of the Gaussian is taken and then applied to the image. After that, the places are found which have the

zero crossings i.e. positive to negative transitions. These crossings are the ones which are considered as the probable edges. Then thresholding is applied to these crossings to get the actual edges. This technique gives better results than the gradient edge detector, but malfunctions can happen at the corners, curves and where grey level intensity varies.

2.5.3 Canny Edge Detection

Canny edge detector [10] is known to be the optimal edge detector for most of the cases and used as the standard edge detection algorithm. This technique is based on the three criterions as shown below.

1. **Good detection**: The algorithm must have low error rate i.e. must minimize the probability of false positives and false negatives.

2. **Good localization**: Edges must be localized i.e. edges must be as close as possible to the true edges.

3. **Minimal response**: An edge in the image should only be marked once, and where possible, image noise should not create false edges i.e. one point for each edge point.

This is the technique that will be used for the edge detection in the real time images. This has better response in real time scenarios and is also resistant to noise. This algorithm is a little more complex than the other ones but is more accurate. This technique will be described in more detail in Chapter 3.

2.6 Object Detection and Recognition

Object detection means detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. Object recognition is the complex process of identifying the particular class of a certain image from its features. Object detection only finds the object in an image whereas object recognition not only finds it but also classifies and categories the object.

Object detection and recognition is one of the basics of computer vision. This can be used for the development of an algorithm to find and recognize the object we are looking for in an image and then decide the outcome. In case of video surveillance, we are tracking for certain objects like

human, face, objects and their activities. So, accurate detection and recognition is very important in this case. There are various research projects going on for this purpose and different algorithms have been developed depending on the nature of the image and video, object type, similarity in the structures and so on. Some of the cases that were studied and analyzed are described below.

2.6.1 Object Detection by motion tracking

This technique deals with object tracking by the means of motion. This is based on the adaptive background subtraction method. The pixel by pixel mapping of the background is stored without any objects in the picture for a period of time. This is made as the base for the tracking system and a mask is created for it. It then compares the real time video with the base mask using frame by frame of the video. Whenever a new object is introduced into the picture, it finds some difference in the original mask and then detects the object by tracking different frames.

This technique is quite complex because it needs an accurate background mask for standardization. It involves pixel by pixel comparison of different frames so computational complexity is very high. There can be various changes in the background depending upon light changes, climatic changes, moving objects like swaying trees and shadows which are quite difficult to track. Hence, object tracking is quite complicated with this approach.

2.6.2 Object recognition by SURF

SURF (Speeded Up Robust Features) is a robust local feature detector, first presented by Herbert Bay, that can be used in computer vision tasks like object recognition or 3D reconstruction [11]. It is based on the key points in an image called the interest points. It is scale and rotation invariant and works very fast for image retrieval.

SURF first finds the interest points, such as corners and blobs (binary large objects) at distinctive locations in an image. A blob is an area of touching pixels with the same logical state. Next, the neighborhood of every interest point is represented by a feature vector. Finally, the descriptor vectors are matched between different images. This technique is quite useful to detect the exact replica of an image. Problems may occur when similar kind of objects need to be found and when incorrect interest points are selected for object matching.

2.6.3 Pedestrian detection using Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) is a feature based technique for object detection. The essential thought behind the HOG descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and for each cell compiling a HOG direction or edge orientation for the pixels within the cell. The combination of these histograms then represents the descriptor.

Pedestrian detection by HOG uses human body shape as the templates which are then used to match with images. It takes different templates of standing people and then finds out the intensity gradients and edge directions of each region of the image to create the HOG. Then HOG template is matched with the real image in each frame to detect the pedestrian as shown in Fig 2.5.

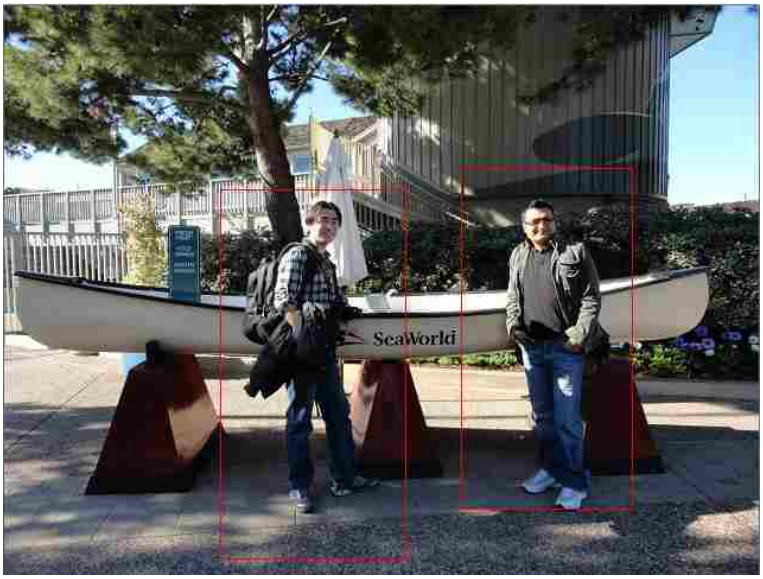

Figure 2.5: Pedestrian detection by HOG

2.6.4 Object detection by shape and color

This technique deals with object tracking by means of shape and color. It differentiates the color of an object from the background color by applying a color filter and then creates the contour,

outline of the figure, from the result which may have been distorted due to the presence of the background. This contour is then compared with the standard shape that has been defined. If the shape and color is the same as the required object then it returns the object as the detected object. This approach is quite simple and also needs much less computation. But this approach is very susceptible to background noise and can give false results as well.

2.6.5 Haar-like feature

Haar-like feature was proposed by Viola and Jones, which is based on Haar wavelets, and was first used as a fast technique for real time face detection, and later enhanced to work for other objects as well. A Haar wavelet is a sequence of rescaled "square-shaped" functions with one high interval and one low interval. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region, and calculates the difference between these sums. This difference is then used to categorize subsections of an image. Fig 2.6 shows the detection of face and eye in an image based on the haar-like feature.

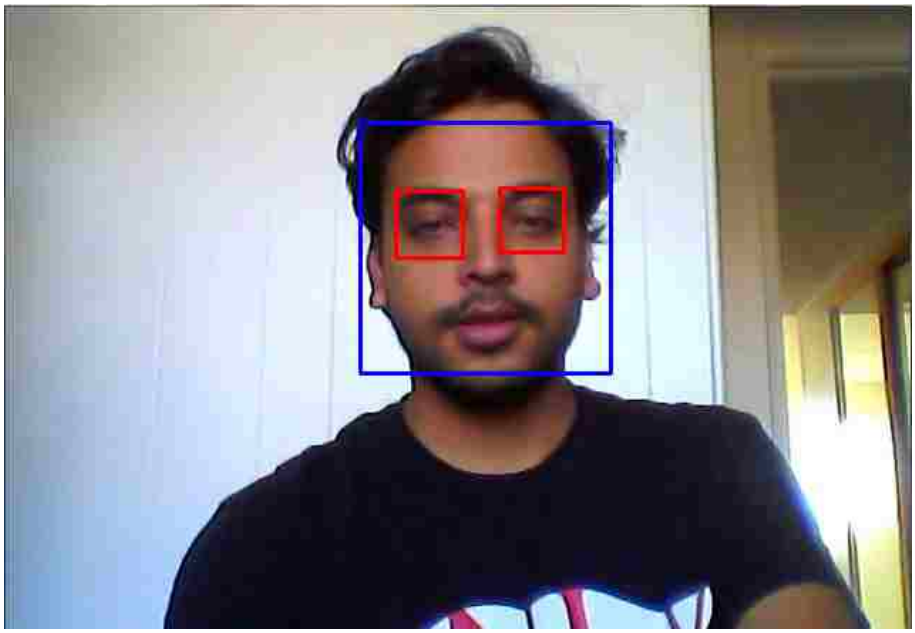

Figure 2.6: Face and eye detection using Haar-like feature

The framework for haar-like feature object detection uses three key terms:

1. **Integral image**: Two-dimensional lookup tables in the form of a matrix with the same size of the original image.

2. **Adaboost**: A machine learning method that combines many "weak" classifiers to create one "strong" classifier.

3. **Cascade**: This allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions.

This is the technique that will be used for the face detection in this thesis and will be described further in Chapter 3.

2.7 Face Detection

Face detection can be regarded as a specific case of object detection. It is a computer technology that determines the locations and sizes of human faces in digital images. It detects facial features and ignores anything else such as buildings, trees and bodies. It has many applications e.g. biometrics, facial recognition, video surveillance, image database etc. Many algorithms have been developed for this with very high success rates. Some of them which were reviewed for our purpose are shown below.

2.7.1 Face detection using skin color

This technique uses color of skin as an information for face detection [12]. The color of skin varies from most of the colors like background, clothes etc. Hence, color segmentation can be implemented to distinguish the skin-like parts from backgrounds. This technique can also detect the face in different orientations, as skin color remains the same even when the face is not facing at the front. But this technique is very dependent on the color so it highly depends on the illumination, intensity of light and background features. Also there should be a large variety of skin color models defined to get accurate results. Presence of exposed necks, hands, arms and other exposed skin regions make the detection even more difficult.

2.7.2 Eigenfaces

Eigenfaces is one of the oldest techniques for face detection and recognition. Eigenfaces are a set of eigenvectors derived from the covariance matrix of the probability distribution of the high-dimensional vector space of possible faces of human beings. Eigenfaces look like some kind of ghostly faces.

Any face image can be represented closely by a set of eigenvectors with each vector contributing to variations in face features. The original image can be reconstructed using only a few eigenvectors and projection values. The reconstructed image of the facial images when projected on eigenfaces looks similar to the original images than non-facial images. This can be used as a base for face detection by eigenfaces. But this technique is also dependent on the different level of light and angles, and also tends to be slower than other algorithms. But eigenfaces are still mostly used for face recognition due to its accuracy and speed.

2.7.3 Haar-like feature

Haar-like feature is the technique used for the face detection in our thesis as it works best in real time processing. This feature has already been explained in section 2.6.5 and will again be dealt in detail in Chapter 3.

CHAPTER 3

THEORY

In this chapter, the framework that has been developed for video surveillance which can be used on a construction site will be discussed. The framework will mostly emphasize on the detection of the safety equipment, which will be tested in the lab, based on the classification algorithm.

3.1 Algorithm

The algorithm for the system is as described below.

1. Capture the video from the site using one or more cameras.
2. Compress and transfer video from the site to the server for processing.
3. Convert the video to frames of images using about 30 frames/second.
4. Convert the images to grayscale for further processing.
5. Apply canny edge detection algorithm to get the edges from the image.
6. Apply Viola and Jones algorithm to detect the face from the image.
7. Compute the head region by using the face as a template.
8. Apply color segmentation to the region.
9. Compute the contour from the segmented image and check if the contour is of the required shape.
10. If the shape is semicircular then a hard hat is present and go back to step 2.
11. If the hard hat is not present generate an alarm event and go back to step 2.

3.2 FlowChart

The flowchart of the framework is shown in Fig 3.1.

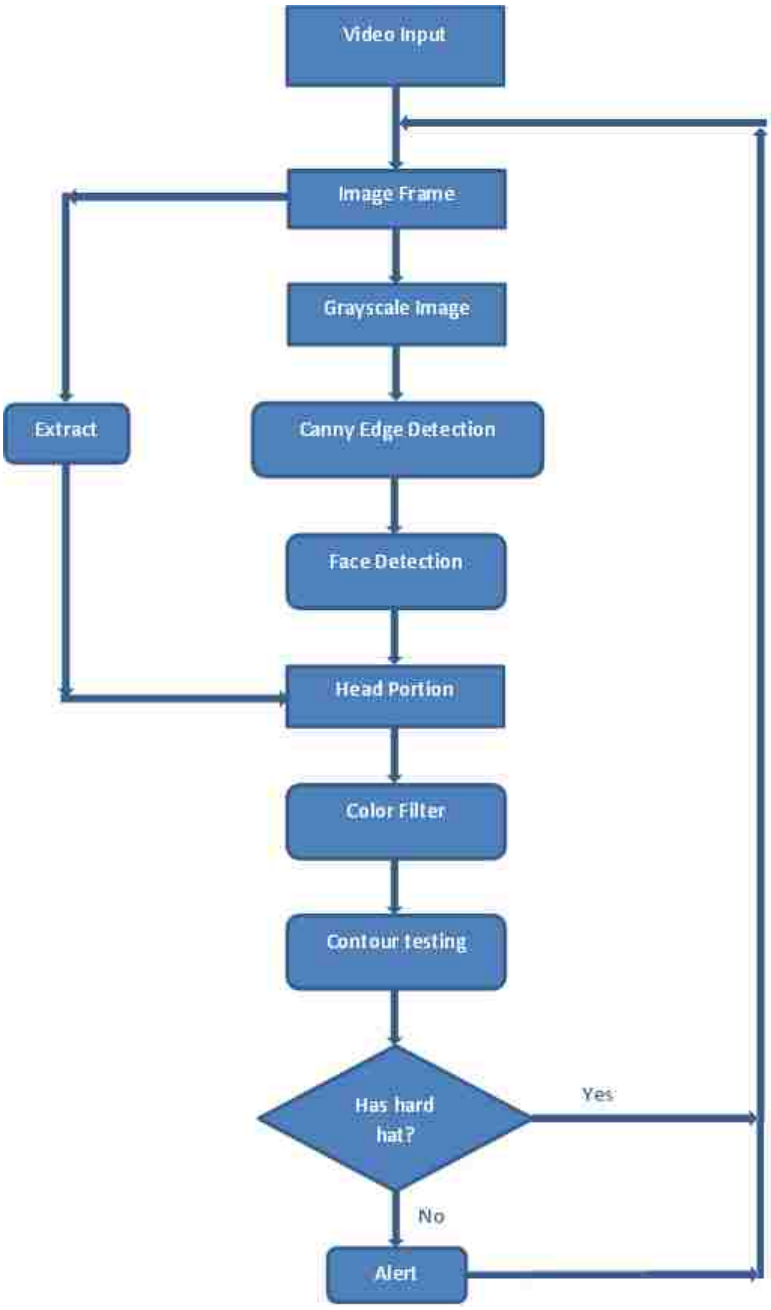

Figure 3.1: Framework Design

3.3 Video Transmission

Transmission of video from site to the server and from server to the client is an important part of video surveillance because data need to be transfered before and after the analysis. As explained in Chapter 2, for a non-compressed video of 1 minute about 1.55GB of storage is required. If we want to transfer that amount of data, it will take about 14.11 (1.55*1024*8/ (15 *60)) minutes on a 15Mbps cable or internet. For real time video surveillance a delay of 14 minute is not appropriate.

Transfer of data also depends on many factors such as machine speed, speed of cable or internet, and handshaking between the machines, which may cause delays or, in the worst case, loss of frames. This loss of data is inappropriate in the case of video surveillance. Even missing 10 frames may lose a chance of catching a violation on the site. So for enhancing accuracy of the transfer and speed, the framework which is described in Fig 3.2 was built.

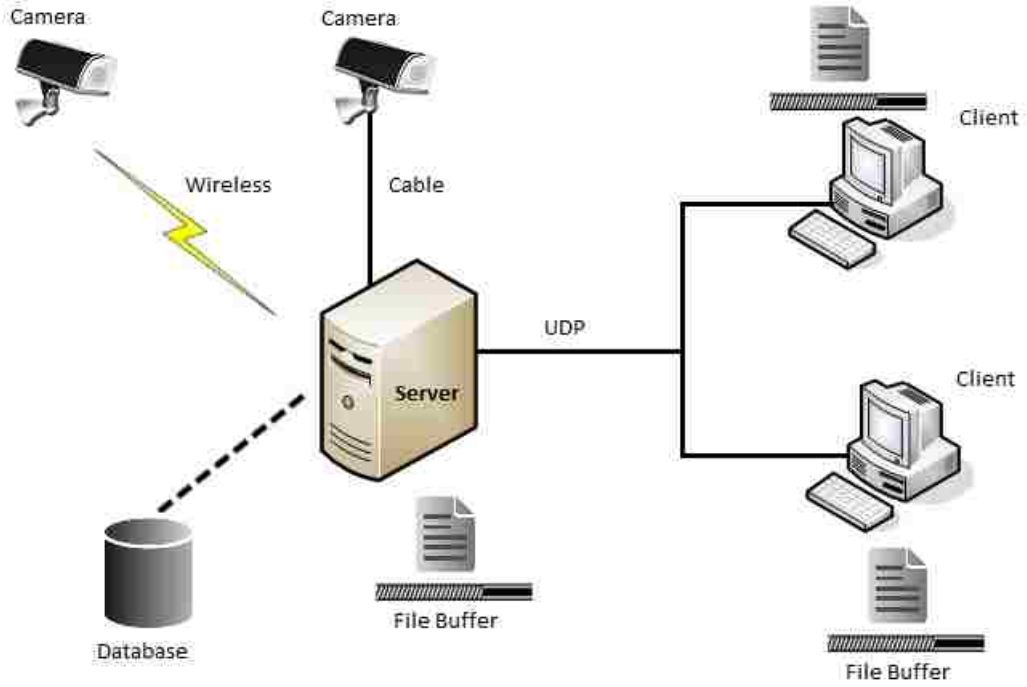

Figure 3.2: Video Transfer Framework

3.3.1 Compression

In the modern era, cameras have built-in compression capabilities to reduce the amount of data that needs to be transferred. In this case, each frame is captured with a resolution of 640x480 pixels at a rate of 30 FPS. Each frame is compressed to JPEG format and then transferred to the server by the camera. This helps in fast capturing and transfer of data by the camera. The compressed frames can be analyzed further for other information that is required.

3.3.2 Video Transfer

Image frames need to be transfered continuously between the server and the client through the network for analysis as well as results. Loss of data and delay is rather inappropriate. In this case, User Datagram Protocol (UDP) is being used as the network protocol for transmission. With UDP, users can send data through the internet without any handshaking. This allows fast transmission of data between peers with a simple protocol. But this may also cause the loss of packets due to the lack of handshaking and error checking.

So to amend this cause, a double buffer technique is being used. Buffers allow the data to be stored in the system and to transfer the data in a cluster. In this case, buffers are used in both the server and the client side in the form of files. A cyclic buffer is used so that the same buffer can be re-used to store all the data. First, the server stores data in the buffer and whenever the buffer reaches a certain amount of data, the data is sent to the client through the network using UDP. The client then fetches data and stores it in the buffer and displays it after reaching a certain limit.

As there is a continuous transfer of data, buffering allows for a smooth transfer of data without any loss in real-time. As data is transferred in a clusters, even losing 1 frame in 100 frames will not cause any problem in reconstruction of the video. Loss of data and delay in data transfer with UDP and buffering is very low which is quite appropriate for real time video transfer.

3.3.3 Video Storage

A database is also added to store the real-time video. Each and every frame of the video is stored into the database, so that the video can be retrieved for future use. The client can request a video stored in the database at any date and time. This history can also be used for the proof of violations in the site and for security analysis. Even if there is a loss of action in real-time, the history can be analyzed to check for any violations. This can not only monitor the violations but also help in stopping such circumstances later on.

3.4 Canny Edge Detection

Canny Edge Detection [13] is one of the most popular algorithms for edge detection. Edge detection can be used for finding the boundary of the images. This drastically reduces the amount of complexity in the analysis. The boundary can be compared with the template of images of the object for the object recognition. Canny is used for the pruning of images in Viola and Jones face detection technique. It removes the non-face like parts so that there is no need to analyze huge amounts of data in the image . The stages of the algorithm are as follows:

3.4.1 Smoothing

Most of the images taken from the camera contain some amount of noise. As a first stage of this algorithm, it uses a filter based on a Gaussian, where the raw image is convolved with a Gaussian filter. After applying the filter the result is a slightly blurred version of the image which has less effect of noise. A 5x5 gaussian filter with with $\sigma = 1.4$ is applied to the image

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}$$

where $\mathbf{B}$ is the resultant blurred image and $\mathbf{A}$ is the original image.

3.4.2 Finding gradients

After suppressing the noise, the algorithm basically finds edges where the grayscale intensity of the image changes the most. These areas are found by determining gradients of the image. Gradients at each pixel in the smoothed image are determined by applying an edge detection operator (Roberts, Prewitt, Sobel etc). The edge gradient strength and direction can be determined using

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$
$$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right).$$

where $\mathbf{G}_x$ and $\mathbf{G}_y$ are the gradients in the x- and y- directions respectively.

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example). These are used to indicate where exactly the edges are.

3.4.3 Non-maximum suppression

The gradient direction is always perpendicular to the edge, and the gray level changes mostly in that direction. If the gray level is not changing significantly in the gradient direction, then it is not an edge point and needs to be suppressed. Each pixels edge strength is compared to the neighboring positive and negative gradient direction pixels edge strength. If it is found to be a maximum, it is preserved, otherwise it is suppressed as it does not belong to local maxima. From this stage, referred to as the non-maximum suppression, a set of edge points, in the form of a binary image, is obtained. These are sometimes referred to as "thin edges".

3.4.4 Double Thresholding

Many of the edge pixel obtained after non-maximum suppression will probably be true edges in the image. Some of the edges may also be caused by the color variation, noise and other factors. This is mostly distinguished by using the threshold. After applying the threshold only edges stronger than a certain value would be preserved. The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong edges, edge pixels weaker than the low threshold are not edges and are suppressed, and edge pixels between the two thresholds are marked as weak edges.

3.4.5 Hysterisis

Edges should always be in continuous curves because edges are closely connected to each other to form the boundary. In this step, strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. Edges due to noise and other small variations are most likely to be removed after the double thresholding by use of proper threshold levels. Even if some unprobable edges are marked as weak edges, they mostly lie far from the strong edges. Only the weak edges that are connected to strong edges can be considered as true edges. This results in a binary image where pixels are mapped by either edges or non-edges with edges being closely connected to each other. The different stages of the Canny Algorithm is shown in Fig 3.3.

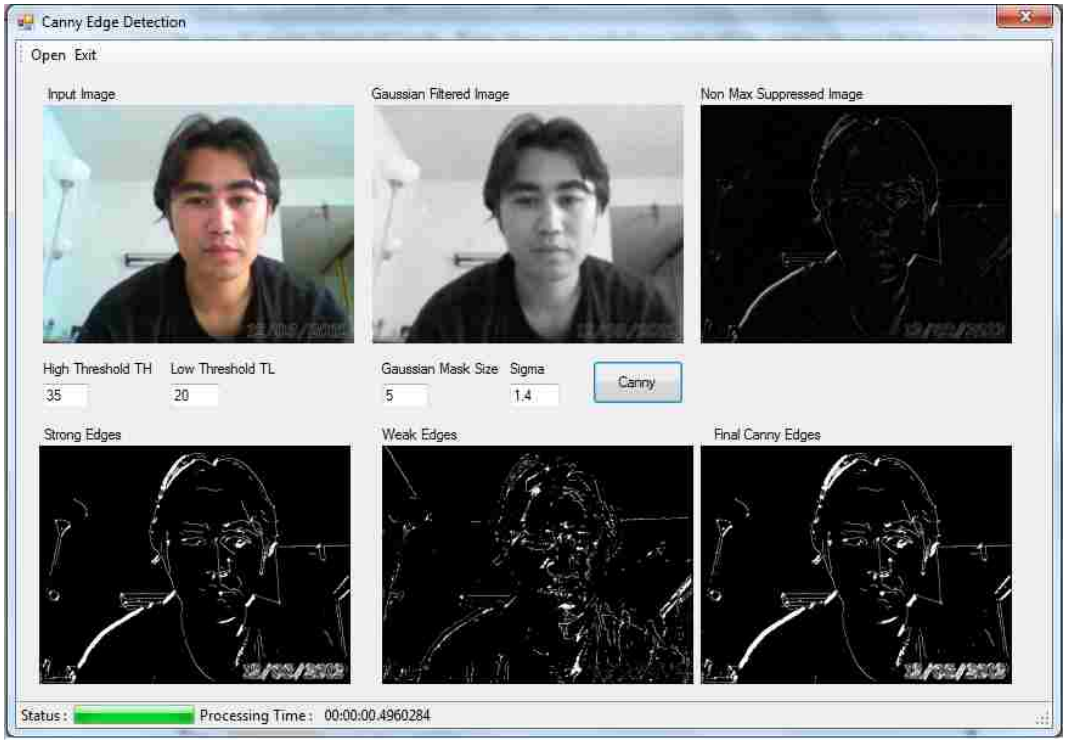

Figure 3.3: Different stages of Canny Algorithm

3.5 Face Detection

Face detection is based on the Viola and Jones method developed for object detection [14] [15]. This technique is very fast and works with very high accuracy. This technique combines the following key concepts.

3.5.1 Haar-like feature

A digital image is composed of RGB colors in two dimensional arrays. The computation with all of these values is quite complicated. So a feature set based on Haar wavelets instead of the usual image intensities was developed which was used by Viola and Jones to create Haar-like features as shown in the Fig 3.4. A Haar-like feature [16] considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image.

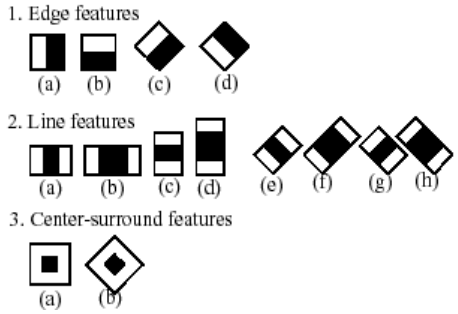

Figure 3.4: Structure of different Haar-like features used

For example, let us say we have an image database with human faces. It is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. A common haar feature for such a case is a set of two adjacent rectangles that lie above the eye and the cheek region. Similarly, there can be another Haar feature for the nose region being lighter than the eyes region. These two Haar-like features are as shown in the Fig. 3.5.

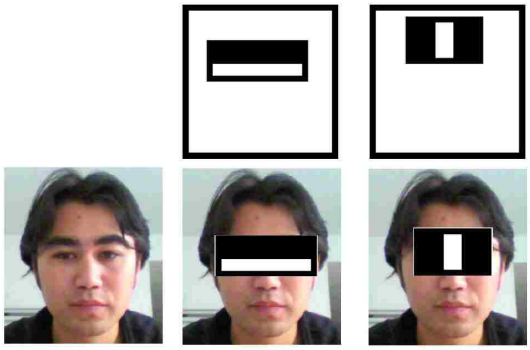

Figure 3.5: First two Haar features in the cascade

The presence of a Haar-like feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present.

3.5.2 Integral Image

To determine the presence or absence of hundreds of Haar features at every image location and at several scales efficiently, Viola and Jones used a technique called an Integral Image. In general, "integrating" means adding small units together. In the case of an image, the small units are pixel values. The integral value for each pixel is the sum of all the pixels to its top left. The entire image can be integrated with a few integer operations per pixel by using the concept of integral image.

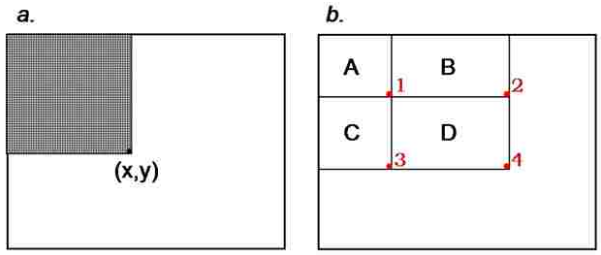

Figure 3.6: Finding the Integral Value of an area

For example, for calculating the integral value at point (x,y) in Fig 3.6 *a*, only the sum of pixel values within the rectangular region that is top left to the point need to be added. The average pixel value can then be found by just dividing by the area.

Similarly in Fig 3.6 *b*, to find the integral value in rectangle *D*, the integral values at the four corner points are needed, requiring only four lookups as shown below:

$$D = \text{I(D)} + \text{I(A)} - \text{I(B)} - \text{I(C)}$$

where $I(x)$ is the integral value at the corner x. Each Haar-like feature may need more than four lookups, depending on how it was defined. Viola and Jones's 2-rectangle features need six lookups, 3-rectangle features need eight lookups, and 4-rectangle features need nine lookups only which can be computed quite efficiently.

3.5.3 Adaboost

To select the specific Haar features to use, and to set threshold levels, Viola and Jones use a machine-learning method called AdaBoost. Adaboost takes a number of features sets and training sets with positive and negative images. The machine is trained with these sets to create a set of weak

classifiers of Haar-like features. AdaBoost selects a set of weak classifiers to combine and assign a weight to each. Good features are assigned larger weights whereas poor features get lesser weight. This weighted combination forms the strong classifier.

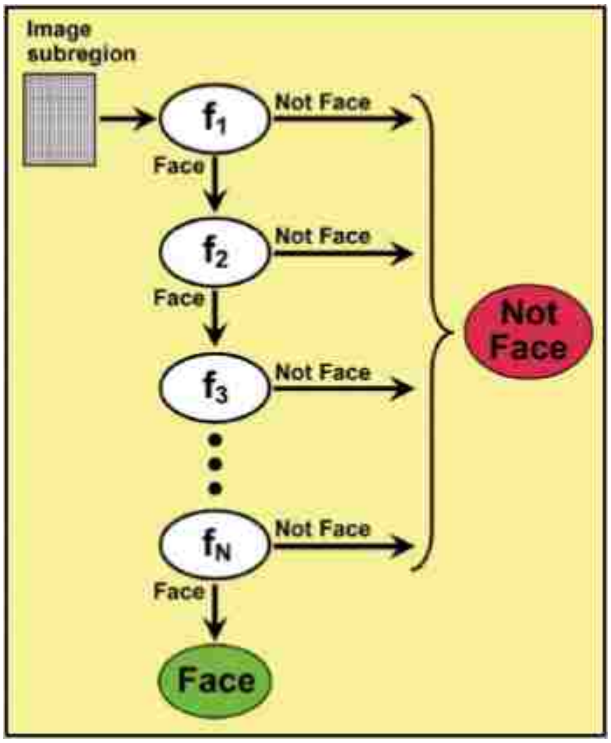

Figure 3.7: Filter chains in cascade

3.5.4 Cascade

The classifiers developed are then arranged in a filter chain as shown in Fig 3.7 with each filter chain consisting of a fairly small number of weak classifiers. These chains are then formed in a cascade to get the final filter. The order of filters in the cascade is based on the importance weighting that AdaBoost assigns. The first chain consists of the classifiers with the highest weights. In each filter chain the acceptance threshold is set low enough to pass nearly all face examples in the training set.

For example, the first chain consists of the classifier with two strong features described above

about the eyes and the nose. This filters out most of the non-face regions from the face regions. If the region passes the test of the first filter then only that region is tested for the next set of features. This helps to eliminate less likely regions quickly so that no more processing is needed in that region, speeding up the overall algorithm.

3.6 Head Segment Extraction

After the face has been detected in the image, the region above the face can be found as the head region. The width of the head region for analysis is taken as twice the width of the face, and the height is taken as the same as the face height. And the position of the top of the head is taken as an offset of 4/5 times the face height. The calculation for the region is shown below.

```
headRegion.rect.Height = face.rect.Height;
headRegion.rect.Y = face.rect.Y - (face.rect.Height *4) /5;
headRegion.rect.Width = 2 * face.rect.Width;
headRegion.rect.X = face.rect.X - (face.rect.Width * 1) / 2;
```

3.7 Color Filtering

The color of the hat is mostly distinct from the background. Mostly bright colors are used for the hard hat like red, green, white, blue, or yellow. So, the color filter can be implemented to distinguish the hat from the background. In this case, a red colored hat is used. Hence, the red color filter is applied to remove background from the image. This filtering gives just the image of the hard hat with some noise in the bitmap image format. If the hat is present, it returns the image; otherwise just some noise image is filtered out.

3.8 Contour testing

The edge detection technique is applied to the resultant image to get the frame of the hat. This frame can be checked with our own set of descriptions for the object. The contour of the image is created from the filtered image. The contour gives the image with the connected regions. This contour can be tested for the semicircular region which is of some particular size. If there is a presence of a semicircular region, we can say that there is a hat on the head of that person, otherwise an alert message to the site manager is generated informing him about the violations on the site.

CHAPTER 4

IMPLEMENTATION AND RESULTS

This chapter describes the implementation details of the video surveillance tool for the construction site and each step of the algorithm. The implementation is done in C#.NET with Microsoft Visual Studio 2010 as Integrated Development Environment (IDE). We have also used EmguCV [17] which is an open-source image processing library for C#.NET.

The project implementation can be divided into two parts which are image processing and video transmission which will be explained in detail in this chapter.

4.1 Image Processing

In image processing, the input image is processed to get the required output. An image is given as input which is then processed through different stages to get the final result. The input can be uploaded directly using the open file dialog or each frame of the video can be taken as the image input. For example, an image with a person in the site with a hard hat. The image is processed through different stages to determine if the person is wearing a hard hat or not, and is returned as the output. The project is developed as a C#.NET Windows form application. The project has a Windows Form named Thesis with the following components as shown in the Fig 4.1.

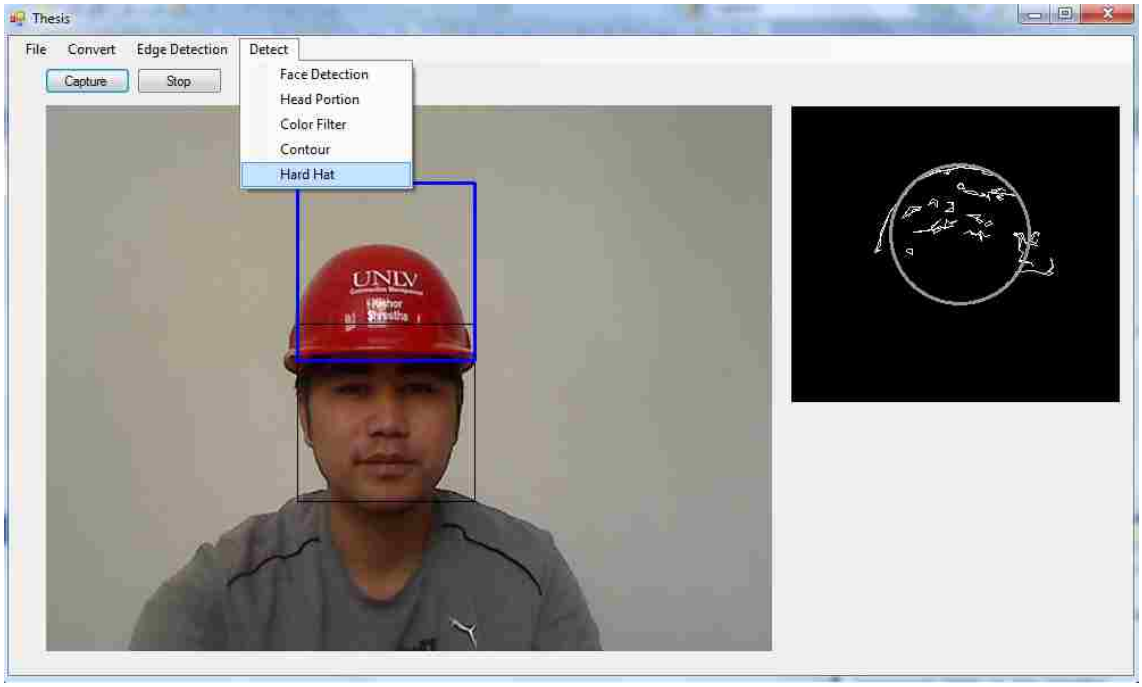

Figure 4.1: MainForm Thesis

There are mainly three types of components in the implementation which are described in detail in the Tables 4.1, 4.2, 4.3.

Table 4.1: Menu Components

S.N.	Menu Tab	Menu Item	Functionality
1	File	Open	Opens the open file dialog to select the image file which needs to be processed and loads it to the image pictureBox.
2		Exit	Close the form and exit.
3	Convert	Gray Scale	Converts the image to GrayScale.
4		Canny Edge	Converts the image to edges using Canny Edge detection.
5	Edge Detection	Sobel	Converts the image to edges using Sobel Edge detection.
6		Marr-Hildreth	Converts the image to edges using Marr-Hildreth Edge detection.
7		Canny	Converts the image to edges using Canny Edge detection.
8	Detect	Face Detection	Detects the face in the image and marks it with a black rectangular box.
9		Head Portion	Detects the head region and displays in the headRegion pictureBox.
10		Color Filter	Applies the color filter to the head region and displays in the headRegion pictureBox.
11		Contour	Finds the contour of the color filtered head region and displays in the headRegion pictureBox.
12		Hard Hat	Finds the presence of the hard hat and then marks it with a blue rectangular box.

Table 4.2: Button Components

S.N.	Button	Functionality
1	Capture	Starts capturing the video using the camera and detects the hard hat in the image frame and displays it to pictureBox.
2	Stop	Stop capturing the video.

Table 4.3: PictureBox Components

S.N.	PictureBox	Functionality
1	imageBox	Displays the image that has been loaded from file or captured from video and also displays the processed image after any menu item has been selected.
2	headRegionBox	Displays the processed head region in the box.

The different stages of the algorithm and their results are shown below.

4.1.1 Image Input

The input of an image can be taken as a directly uploaded image from the openfileDialog as shown in Fig 4.2 or each frame of the video captured from the camera.

Figure 4.2: File uploaded from the openfileDialog

4.1.2 Grayscale

The image is converted to the grayscale as shown in Fig 4.3 for further processing. As the grayscale contains only one value of color, it makes the processing faster and less complex. An image is converted to grayscale by combining all three color intensities using the formula

$$Y = 0.299R + 0.587G + 0.114B$$

Figure 4.3: Grayscale Image

4.1.3 Edge Detection

The edges from the image are computed using the Canny Edge Detection algorithm as shown in Fig 4.4. An upper threshold of 100 and a lower threshold of 50 is used for this case.

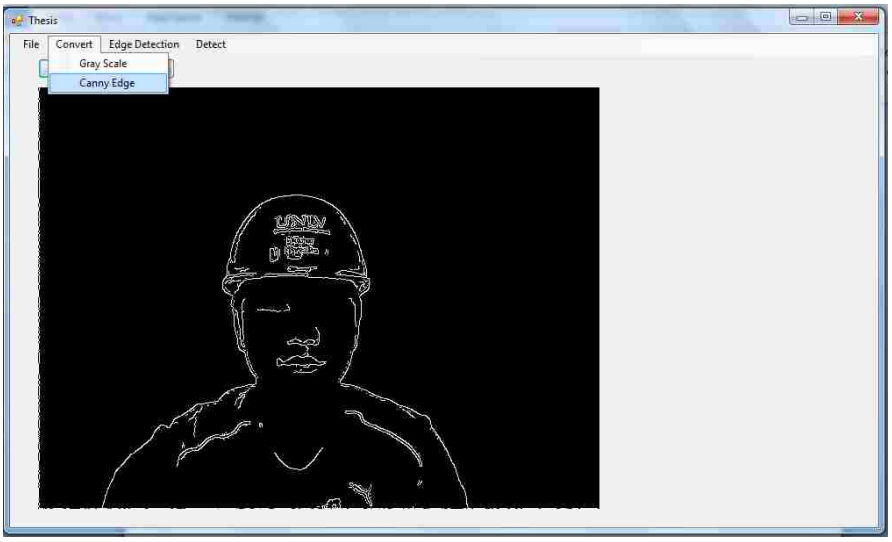

Figure 4.4: Edges found after applying Canny Edge Detection

4.1.4 Face Detection

Face detection is done by using the Viola and Jones Haar-like feature. EmguCV has already created the haarcascade XML file for detecting the face features. This cascade is created with the help of a number of positive and negative samples. The cascade can be created by using the haartraining with our own samples, but right now our purpose is just detection of the face, so the sample given by EmguCV is used.

Canny edges are used for pruning the non-face and face regions to reduce the complexity of the algorithm. The non-faces regions are rejected in the first cascade to make it work faster. The varying region size is taken to check for the face like regions. These regions are grouped into rectangles and whenever more than the specified overlapping regions are found, the algorithm returns that region as the face. The result of the face detection in the image is as shown in Fig 4.5.

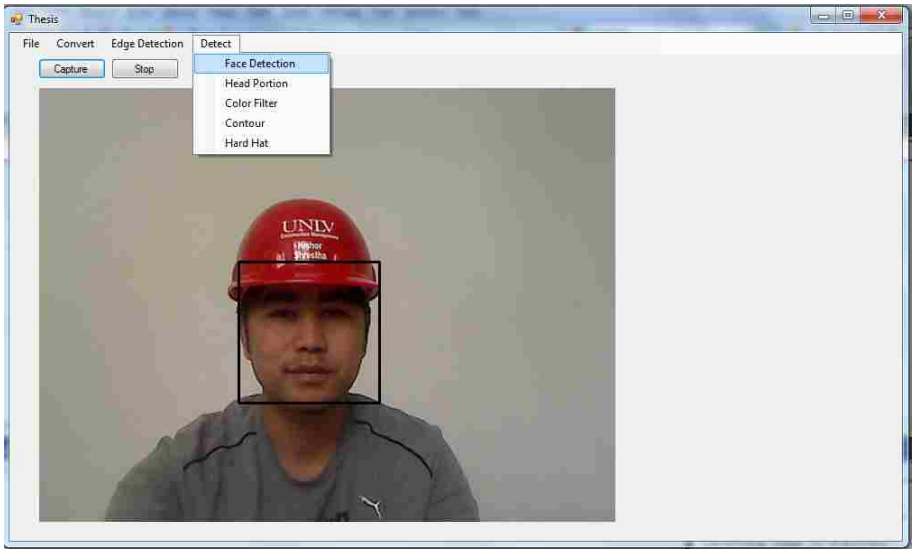

Figure 4.5: Face Detected in the image

4.1.5 Head Region Analysis

After detecting the face, the head region is computed. Then color filtering is applied to get the filtered binary image. Then, the contour is determined from the filtered image. The contour may contain some noise due to the variance in color of the image. The results for head region analysis

are shown in Fig 4.6 .

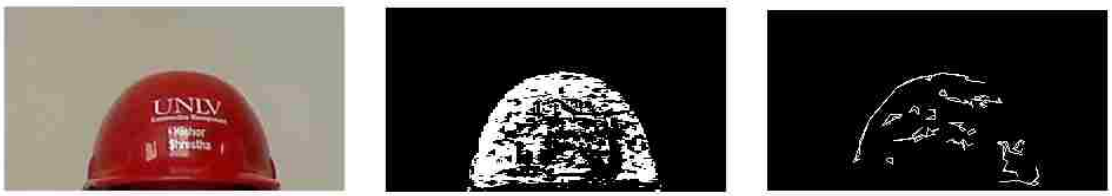

Figure 4.6: Head Region Analysis

4.1.6 Hard Hat Detection

The contour formed from the head region analysis is tested for the presence of a semicircular region. If there is a presence of the region then it returns the image with the hard hat as shown in Fig 4.7. The hard hat is marked with a blue rectangular box in the head region.

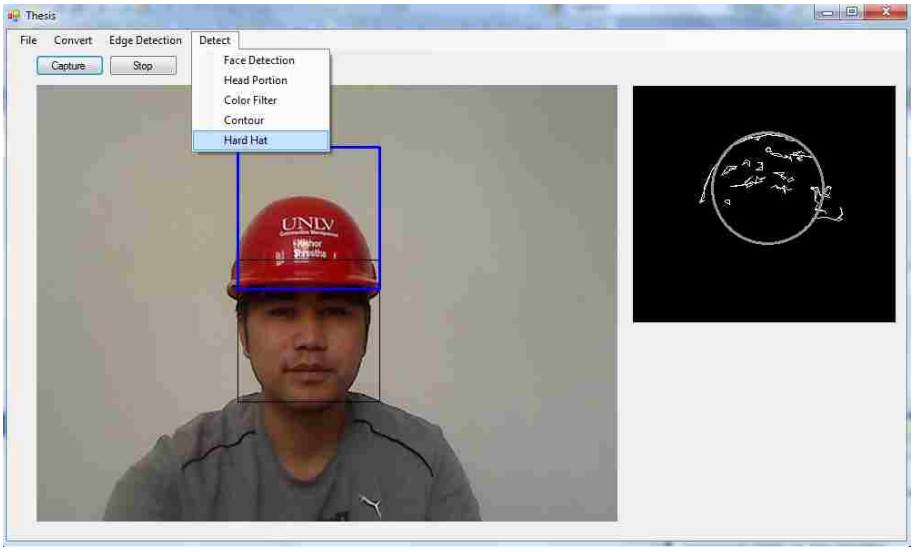

Figure 4.7: Hard Hat detected in the head

For the case with no hat on the head, the color filtering filters the image, and the contour test gives no semicircular region in the head region which confirms the head with no hat. The result for the image containing no hat is shown in Fig 4.8.

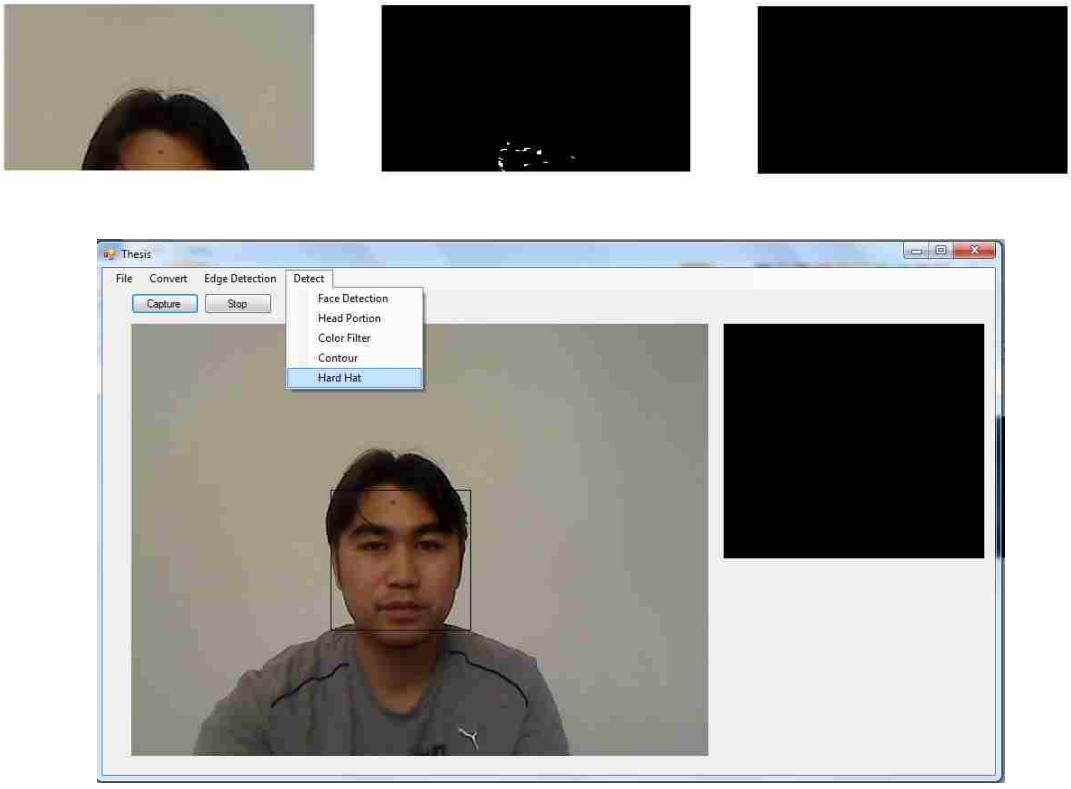

Figure 4.8: No hat detected

4.2 Video Transmission

Real-time video is captured by means of a camera and then transferred to the server. The server processes the video and also saves data for future references in the database. The client can request data from the server at any time. The client can either request live video or past data from the database. Both data are sent to the client using UDP and buffering. The details of the video transmission and results are described below.

4.2.1 Server

This is the place where all the processing takes place. The real time video is captured and stored in the server. Video is broken down to the frames and then is analyzed by using the image processing technique for the presence of hard hats. The interface for the server is as shown in Fig 4.10.

Figure 4.9: Server Form

This form consists of two buttons to start and stop the video and the imageBox that displays the image after processing is done. Each and every frame of the image that has been processed is stored into the mysql database. The database design for mysql is as shown in the Fig 4.10.

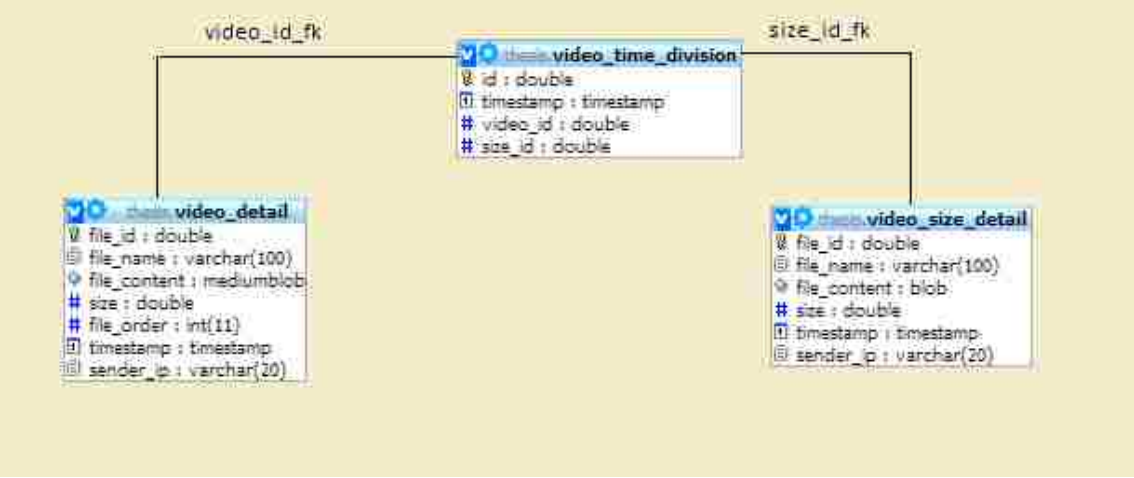

Figure 4.10: Database Design

4.2.2 Client

A client can request the server to get the real-time video as well as the video from history. The form for a client is as shown in the Fig 4.11.

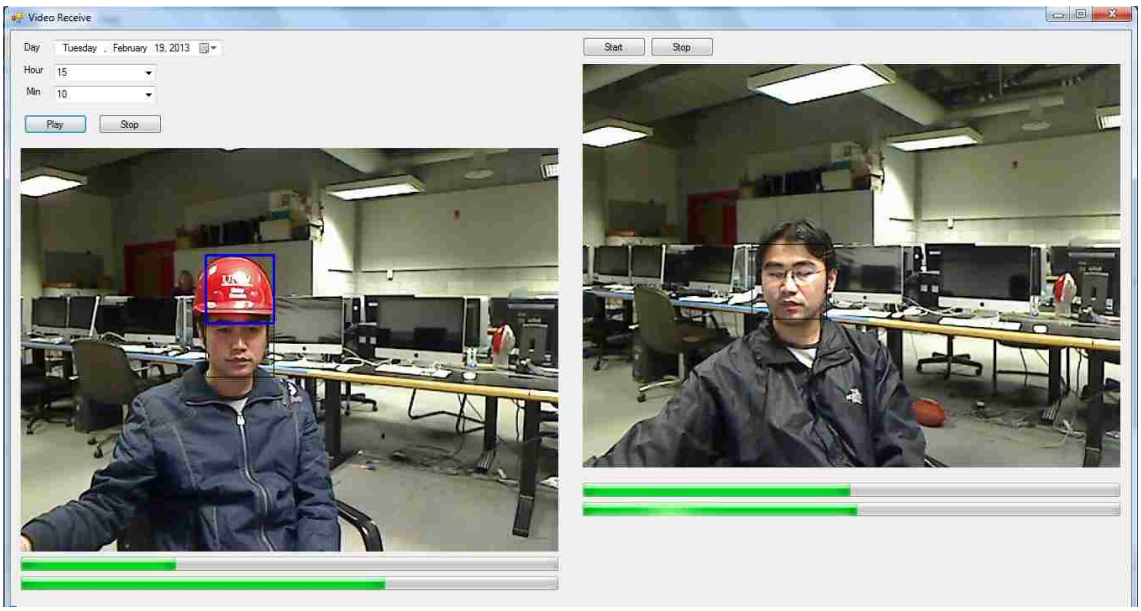

Figure 4.11: Client Form

The form consists of two parts, one to show the live video and the other to show the video from the history, which are both results after image processing. Two videos can be played in parallel simultaneously. Different UDP ports are used for the parallel transfer. There are start and stop buttons to play the live video in the right part of the form. The video is buffered in files and then shown in the interface as continuous streaming. The video from history can also be selected by selecting the date, hour and min from the dropdown and then clicking on the play button. The stop button can be used to stop the history video. History can be fetched as a slot of 10 minute intervals.

CHAPTER 5

CONCLUSION

In this thesis, a method was successfully developed to perform image processing of a video to analyze the presence of safety equipment. Our own classification algorithm was built to classify the equipment from the non-equipment. This algorithm can be utilized to uncover any kind of safety violations taking place in construction sites.

Using the algorithm, the video was split into frames of images. Each frame was first converted into edges and then was checked for the presence of a face region using a face detection algorithm. Each segmented face region was analyzed to get the region of the head where a hard hat could reside. The head region was then checked for the presence of a hat which was displayed in the image.

Different techniques and algorithms were reviewed for development of a framework for finding edges, segmentation of objects from the image, face detection and object recognition. Canny edge detection for edge detection and Haar-like feature for face detection were found to be best in the case of real-time video monitoring as it gave more accurate results in a very efficient manner. These techniques were implemented along with some of our own for detecting a hard hat in the head region of a person.

A framework for saving the history of video was also developed alongwith the image processing technique. A server client configuration was created for efficient transfer of video in the network. Clients can continuously get live video as well as past video. This is quite useful in video surveillance and can act as evidence in the case of a violation. UDP protocol with buffering was used for the efficient transferring of data, and data was stored into a MySQL database for future analysis.

This thesis can be utilized to develop an efficient video surveillance tool. But still there are a few improvements that need to be done. Right now it can detect hard hats only by detecting the frontal face. This might not be the case always. So further research should be done to detect

hats in any position of the head and to alert in the event of a violation. Similarly, research needs to be done to detect other equipment like safety jackets, safety shoes, etc., using their own properties.

In conclusion, this approach can be quite helpful in building an efficient video surveillance tool to reduce fatalities and injuries in construction sites, which can also be used in other fields needing intellegent video monitoring.

APPENDIX A

CODE SNIPPETS

```
//Detect Hard Hat in the image
//Input: Image, Stage-> To set each stage of hard hat detection like head region
//extract, color filtering etc.
//Output: Image -> Processed image with face and hard hat detection
private Image<Bgr, Byte> detectHardHat(Image<Bgr,Byte> image, int stage) {
    Console.Out.WriteLine("Detecting Hard Hat...");
    try
    {
        MCvAvgComp[] faces = faceDetection(image);
        Image<Bgr, Byte> newImage = image.CopyBlank();
        foreach (MCvAvgComp face in faces)
        {
            //Counter for the cirles in the processed image
            int counter = 0;

            //Object to store the region of the head above face for analysis
            var headRegion = face;
            headRegion.rect.Height = (face.rect.Height) ;
            headRegion.rect.Y = face.rect.Y - (face.rect.Height * 8) / 10;
            headRegion.rect.Width = face.rect.Width + face.rect.Width;
            headRegion.rect.X = face.rect.X - (face.rect.Width * 2) / 4;

            //Set the region of the image to headRegion for analysis
            image.ROI = headRegion.rect;

            //Object to store the region of the hard hat for display
```

```
var hardHat = face;
hardHat.rect.Height = face.rect.Height;
hardHat.rect.Y = face.rect.Y - (face.rect.Height * 8) / 10;
hardHat.rect.Width = face.rect.Width;
hardHat.rect.X = face.rect.X;

//Convert Image to HSV format for filtering
Image<Hsv, Byte> hsv = image.Convert<Hsv, Byte>();
if (stage == 1) headRegionBox.Image = image.ToBitmap();
//Apply the red color filter in the HSV model
grayScale_Image = hsv.InRange(new Hsv(0, 80, 80), new Hsv(8, 240, 240));

if (stage == 2) headRegionBox.Image = grayScale_Image.Convert<Bgr, Byte>
    ().ToBitmap();
//List to store the contour region
List<Contour<Point>> list = new List<Contour<Point>>();

//Find the canny of the filtered image
grayScale_Image = cannyEdgeDetect(grayScale_Image.Convert<Bgr, Byte>(),
    100, 50);

//Find the contours
Contour<Point> ctrs = grayScale_Image.FindContours();

while (ctrs != null)
{
    Contour<Point> ctr = ctrs.ApproxPoly(1);
    if (ctr.Area > 4) { list.Add(ctr); }
    ctrs = ctrs.HNext;
}
//Draw the contours to newImage
foreach (Contour<Point> lis in list)
{
    newImage.Draw(lis, new Bgr(Color.White), 1);
```

```
        }

        if(stage!=5) grayScale_Image = newImage.Convert<Gray, Byte>();
        if (stage == 3) headRegionBox.Image = grayScale_Image.Convert<Bgr, Byte>
            ().ToBitmap();
        //Find circles in the new contour image
        CircleF[] circles = grayScale_Image.Convert<Gray, Byte>().HoughCircles
            (new Gray(220), new Gray(160), 5, 40, (headRegion.rect.Height * 7)
            / 20, headRegion.rect.Height / 2)[0];
        foreach (CircleF circle in circles)
        {
            counter++;
            if(counter==1) grayScale_Image.Draw(circle, new Gray(150), 2);
        }

        //Reset the region of image back to whole image
        image.ROI = Rectangle.Empty;

        //Draw rectangle for each face detected
        image.Draw(face.rect, new Bgr(0, 0, 0), 1);

        //Draw hard hat that has been detected
        if (counter > 0 && (stage==4 || stage==5))
        {

            image.Draw(hardHat.rect, new Bgr(255, 0, 0), 2);
        }
        if (stage == 4 || stage == 5) headRegionBox.Image = (grayScale_Image.
          Convert<Bgr, Byte>()).ToBitmap();

    }
    return image;
}
catch(Exception ex){
```

```
        Console.Out.WriteLine("Error in detectHardHar(): " + ex.Message);
    }
    return myImage;
}
//detectHardHat ends


//Capture frames from video and save it to file buffer
//Input: Object, Event
//Output: void
private void FillBuffer(object sender, EventArgs e)
{
    int inx = 0;
    String frameFile;
    {
        try
        {
            //Filename to save the captured image
            frameFile = "captureFrame" + inx.ToString() + ".jpg";

            //Get current frame from video.
            Image<Bgr, Byte> currentFrame = camera.QueryFrame();
            //If current frame is not null detect hard hat and show
            if (currentFrame != null)
            {
                myImage = currentFrame;
                Image<Bgr, Byte> detectImage = detectHardHat(myImage);
                showImage(detectImage.ToBitmap());
                image = detectImage.ToBitmap();
            }

            imgDataStream = new MemoryStream();
            //Save the image to the memory stream
            image.Save(imgDataStream, ImageFormat.Jpeg);
```

```
            image.Dispose();
            //Convert the image to array of element and set to buffer
            sendBuffer = imgDataStream.ToArray();
            //Save the buffer to database for future use
            try
            {
                saveToBuffer(sendBuffer);
            }
            catch(Exception ex) { Console.Out.WriteLine("Error in sending-> "+
                ex.Message); }
            imgDataStream.Close();

        }
        catch (Exception ex){
            Console.Out.WriteLine("Error in FillBuffer-> " +ex.Message);
        }
    }
}
//FillBuffer ends


//Save the data in buffer to database
//Input: Array of byte
//Output: void
private void saveToBuffer(byte[] sendBuffer)
{
    try
    {
        Console.Out.WriteLine("Writing output to database");
        DateTime currenttime = System.DateTime.Now;
        if (saveByteArray.Count==0)
            currenttime=System.DateTime.Now;
        saveByteArray.Add(sendBuffer);
        saveFrameSize.Add(sendBuffer.Length);
```

```
int maxFiles = AppSettingsController.GetAppSetting("BufferFilesLimit", 10);
int maxFrames = AppSettingsController.GetAppSetting("FramesInFileLimit", 40);
//Save the data only if certain threshold of frames are present
if (saveByteArray.Count >= maxFrames)
{
    fileCount++;
    //Loop through the buffer using filecount
    if (fileCount > maxFiles)
        fileCount = 1;

    //Filename to store video
    String vidFilename = "Video" + fileCount.ToString() + ".dat";
    //Filename to store size
    String sizeFilename = "Siz" + fileCount.ToString() + ".dat";
    //Create video file
    FileStream vidFileStream = new FileStream(vidFilename, FileMode.Create,
        FileAccess.Write);
    //Create size file
    StreamWriter sizeFileStream = new StreamWriter(sizeFilename);

    //Write all bufferdata to the file
    foreach (byte[] byteArrayElement in saveByteArray)
        vidFileStream.Write(byteArrayElement, 0, byteArrayElement.Length);
    //Close video file
    vidFileStream.Close();

    //Write the size of data to file. Used later for retreiving data
    foreach (int sizeArrayElement in saveFrameSize)
        sizeFileStream.WriteLine(sizeArrayElement);

    //Close size file
    sizeFileStream.Close();
```

```
//Save to database
//Create conncection
connection = new MySqlConnection(ConnectionString);
connection.Open();
try
{
    //Read content from the video file
    byte[] content=System.IO.File.ReadAllBytes(vidFilename);

    MySqlCommand command = new MySqlCommand("", connection);
    //Insert into the video_detail table
    command.CommandText = "insert into video_detail(file_name, "+
        "file_content, size, file_order, timestamp, sender_ip) "+
        "values (@name, @content, @size, @order, @time, @ip);";
    command.Parameters.AddWithValue("@name", vidFilename);
    command.Parameters.AddWithValue("@content", content);
    command.Parameters.AddWithValue("@size", 1024);
    command.Parameters.AddWithValue("@order", 2);
    command.Parameters.AddWithValue("@time", currenttime);
    command.Parameters.AddWithValue("@ip", "127.0.0.1");
    //Execute insert
    command.ExecuteNonQuery();

    //Select the last insertedid from video table and set to videoID
    MySqlCommand idcmd = new MySqlCommand("SELECT last_insert_id()",
        connection);
    MySqlDataReader dataReader = idcmd.ExecuteReader(CommandBehavior.
        SequentialAccess);
    dataReader.Read();
    double videoID = dataReader.GetDouble(0);
    Console.WriteLine(videoID);
    dataReader.Close();
```

```
MySqlCommand sizecommand = new MySqlCommand("", connection);
//Insert into the video_size_detail table
sizecommand.CommandText = "insert into video_size_detail("+
    "file_name, file_content, size, timestamp, sender_ip) "+
    "values (@name, @content, @order, @time, @ip);";
sizecommand.Parameters.AddWithValue("@name", sizeFilename);
sizecommand.Parameters.AddWithValue("@content",
    System.IO.File.ReadAllBytes(sizeFilename));
sizecommand.Parameters.AddWithValue("@size", 1024);
sizecommand.Parameters.AddWithValue("@order", 2);
sizecommand.Parameters.AddWithValue("@time", currenttime);
sizecommand.Parameters.AddWithValue("@ip", "127.0.0.1");
//Execute insert
sizecommand.ExecuteNonQuery();

//Select the last insertedid from size table and set to sizeID
MySqlCommand sizeidcmd = new MySqlCommand("SELECT last_insert_id()",
    connection);
MySqlDataReader sizedataReader = sizeidcmd.ExecuteReader(
    CommandBehavior.SequentialAccess);
sizedataReader.Read();
double sizeID = sizedataReader.GetDouble(0);
Console.WriteLine(sizeID);
sizedataReader.Close();


MySqlCommand timedivcommand = new MySqlCommand("", connection);
//Insert the sizeID and videoID to video_time_division
timedivcommand.CommandText = "insert into video_time_division("+
    "timestamp, video_id, size_id) values "+
    "(@time, @videoid, @sizeid);";
timedivcommand.Parameters.AddWithValue("@time", currenttime);
timedivcommand.Parameters.AddWithValue("@videoid", videoID);
timedivcommand.Parameters.AddWithValue("@sizeid", sizeID);
```

```
                timedivcommand.ExecuteNonQuery();

                connection.Close();

            }
            catch (Exception ex)
            {

                Console.WriteLine("Error in database" + ex.Message);
            }

            //Clear buffer and array
            saveByteArray.Clear();
            saveFrameSize.Clear();
            filesBuffered = filesBuffered > 5 ? 5 : filesBuffered+1;

        }

    }
    catch (Exception ex)
    {
        Console.WriteLine("save to Buffer: "+ex.Message);
    }
}
//saveToBuffer ends
```

BIBLIOGRAPHY

[1] Wikipedia, "Construction site safety." http://en.wikipedia.org/wiki/Construction site safety.

[2] P. P. Shrestha, E. A. Yfantis, and K. Shrestha, "Construction safety visualization," 4th Int. Multi-Conf. on Eng. and Tech. Inno.(IMETI), Orlando, Florida, U.S.A., 2011.

[3] OSHA, "OSHA Pocket Guide." http://www.osha.gov/Publications/osha3252.pdf.

[4] BLS, "Bureau of Labor Statistics," *Injuries, Illnesses, and Fatalities: Census of Fatal Occupational Injuries (CFOI)*, 2011. http://www.bls.gov/iif/oshwc/cfoi/cftb0259.pdf.

[5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New Jersey: Prentice-Hall, second ed., 2002.

[6] Wikipedia, "Image Segmentation." http://en.wikipedia.org/wiki/Image segmentation.

[7] Q. Li, S. Yang, and S. Zhu, "Image segmentation and major approaches," in *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, vol. 2, pp. 465–468, IEEE, 2011.

[8] Wikipedia, "Edge Detection." http://en.wikipedia.org/wiki/Edge detection.

[9] M. Sharifi, M. Fathy, and M. T. Mahmoudi, "A classified and comparative study of edge detection algorithms," in *Information Technology: Coding and Computing, 2002. Proceedings. International Conference on*, pp. 117–120, IEEE, 2002.

[10] Wikipedia, "Canny Edge Detector." http://en.wikipedia.org/wiki/Canny edge detector.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[12] L. Zou and S.-i. Kamata, "Face detection in color images based on skin color models," in *TENCON 2010-2010 IEEE Region 10 Conference*, pp. 681–686, IEEE, 2010.

[13] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 679–698, Nov 1986.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[15] P. Viola and M. Jones, "Robust real-time face detection," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, p. 747, 2001.

[16] Wikipedia, "Haar Like Features." http://en.wikipedia.org/wiki/Haar-like features.

[17] EmguCV, ".NET Wrapper for OpenCV image processing library." http://www.emgu.com.

[18] S. Du, M. Shehata, and W. Badawy, "Hard hat detection in video sequences based on face features, motion and color information," in *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, vol. 4, pp. 25–29, IEEE, 2011.

[19] Y. Liu, W. Zhou, H. Yin, and N. Yu, "Tracking Based on SURF and Superpixel," in *Image and Graphics (ICIG), 2011 Sixth International Conference on*, pp. 714–719, IEEE, 2011.

[20] G. Xu, X. Wu, L. Liu, and Z. Wu, "Real-time pedestrian detection based on edge factor and Histogram of Oriented Gradient," in *Information and Automation (ICIA), 2011 IEEE International Conference on*, pp. 384–389, IEEE, 2011.

[21] Hu, Weiming and Zhou, Xue and Li, Wei and Luo, Wenhan and Zhang, Xiaoqin and Maybank, Stephen, "Active Contour-Based Visual Tracking by Integrating Colors, Shapes and Motions," in *Image Processing, IEEE Transactions on*, p. 1, IEEE, 2012.

[22] Z. Li, B. Qiao, and S. Deng, "Color-based visual object tracking with prediction and error judgment," in *Image and Signal Processing, 2009. CISP'09. 2nd International Congress on*, pp. 1–4, IEEE, 2009.

[23] A. Mohan and N. Sudha, "Fast face detection using boosted eigenfaces," in *Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium on*, vol. 2, pp. 1002–1006, IEEE, 2009.

VITA

Graduate College
University of Nevada, Las Vegas

Dinesh Bajracharya

Degrees:

Bachelor's Degree in Computer Engineering, Pulchowk Campus, Institute of Engineering, Tribhuvan University, Nepal, 2007

Thesis Title: Real Time Pattern Recognition In Digital Video With Applications To Safety In Construction Sites

Thesis Examination Committee:

Chairperson, Dr. Evangelos A. Yfantis
Committee Member, Dr. Laxmi P. Gewali
Committee Member, Dr. John T. Minor
Graduate College Representative, Dr. Peter A. Stubberud