MOVING OBJECT DETECTION AND TRACKING

A FINAL YEAR PROJECT REPORT

Submitted by

PRAVEEN KUMAR M (2017105574)

MOHAN KUMAR JC (2017105560)

RUPAN KUMAR K (2017105583)

RAHUL SOLAIAPPAN (2017105066)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

DEPARTMENT OF ELECTRONICS ANDCOMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY, CHENNAI-600025

ANNA UNIVERSITY
CHENNAI - 600025

BONAFIDE CERTIFICATE

Certified that this Report titled **"MOVING OBJECT DETECTION AND TRACKING"** is the bonafide work of **PRAVEEN KUMAR M (2017105574), MOHAN KUMAR JC (2017105560), RUPAN KUMAR K (2017105583), RAHUL SOLAIAPPAN (2017105066)** who carried out work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. M.A. BHAGYAVENI
Professor and Head
Department of Electronics and Communication Engineering
Anna University
Chennai - 600025

Dr. C.M. SUJATHA
Associate Professor
Department of Electronics and Communication Engineering
Anna University
Chennai - 600025

ACKNOWLEDGEMENT

We wish to express sincere gratitude to **Dr. M.A. BHAGYAVENI,** Professor and Head, Department of Electronics & Communication Engineering for her enthusiastic encouragement and support throughout the project.

We present our sincere thanks and gratitude to our project supervisor, **Dr. C.M. SUJATHA,** Associate Professor, Department of Electronics and Communication Engineering for her whole hearted support, patience, valuable guidance, technical expertise and encouragement in our project.

We thank all the teaching and non-teaching staff of Department of Electronics and Communication Engineering, for their kind help and co-operation during the course of our project.

Last but not the least, we would not have made this project without the support from our beloved family and friends.

PRAVEENKUMAR M
MOHANKUMAR JC
RUPANKUMAR K
RAHULSOLAIAPPAN

ABSTRACT

Detecting and tracking objects are among the most prevalent and challenging tasks that a surveillance system has to accomplish in order to determine meaningful events and suspicious activities, and automatically annotate and retrieve video content. Under the business intelligence notion, an object can be a face, a head, a human, a queue of people, a crowd as well as a product on an assembly line. Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Object tracking can be defined as the process of locking on to a moving object and being able to determine if the object is the same as the one present in the previous frame.

There are many approaches to detection and tracking using multiple algorithms each of which has their own unique advantages and disadvantages. MATLAB and OpenCV python are a commonly used as a tool, with the help of which image processing techniques like background subtraction, frame differencing, gray-scaling, edge detection, etc. are implemented. In this project tools such as SSD and MobileNet are used for detection and tracking process and the final results are obtained. In this project, object detection and tracking has been implemented using tools such as SSD and MobileNet. It is proposed that the output will encompass being able to identify certain classes (ex: teddy bear, ball, bed, etc.) from a set and track them as they move along a frame in real time. Along with this, a hand recognition system is planned to allow switching between different programs revolving around different unique detection and tracking applications.

திட்டபணி சுருக்கம

பொருள்களைக் கண்டறிதல் மற்றும் கண்காணித்தல் என்பது ஒரு கண்காணிப்பு அமைப்பு அர்த்தமுள்ள நிகழ்வுகள் மற்றும் சந்தேகத்திற்கிடமான செயல்பாடுகளைத் தீர்மானிக்க வேண்டும், மேலும் வீடியோ உள்ளடக்கத்தை தானாகவே சிறுகுறிப்பு செய்து மீட்டெடுக்க வேண்டும். வணிக நுண்ணறிவு கருத்தின் கீழ், ஒரு பொருள் ஒரு முகம், ஒரு தலை, ஒரு மனிதன், மக்கள் வரிசை, ஒரு கூட்டம் மற்றும் ஒரு வரிசையில் ஒரு தயாரிப்பு இருக்க முடியும். பொருள் கண்டறிதல் என்பது கணினி பார்வை மற்றும் பட செயலாக்கம் தொடர்பான கணினி தொழில்நுட்பமாகும், இது டிஜிட்டல் படங்கள் மற்றும் வீடியோக்களில் ஒரு குறிப்பிட்ட வகுப்பின் (மனிதர்கள், கட்டிடங்கள் அல்லது கார்கள் போன்றவை) சொற்பொருள் பொருள்களைக் கண்டறிவதைக் கையாள்கிறது. பொருள் கண்காணிப்பு என்பது நகரும் பொருளைப் பூட்டுதல் மற்றும் முந்தைய சட்டகத்தில் உள்ளதைப் போலவே பொருளைத் தீர்மானிக்க முடியுமா என்று வரையறுக்கலாம்.

பல வழிமுறைகளைப் பயன்படுத்தி கண்டறிதல் மற்றும் கண்காணிப்பதற்கான பல அணுகுமுறைகள் உள்ளன, அவை ஒவ்வொன்றும் அவற்றின் தனித்துவமான நன்மைகள் மற்றும் தீமைகள் உள்ளன. MATLAB மற்றும் OpenCV பைதான் பொதுவாக ஒரு கருவியாகப் பயன்படுத்தப்படுகின்றன, இதன் உதவியுடன் பின்னணி கழித்தல், பிரேம் வேறுபாடு, சாம்பல்-அளவிடுதல், விளிம்பில் கண்டறிதல் போன்ற பட செயலாக்க நுட்பங்கள் செயல்படுத்தப்படுகின்றன. இந்த திட்டத்தில் SSD மற்றும் MobileNet போன்ற கருவிகள் கண்டறிதல் மற்றும் கண்காணிப்பு செயல்முறைக்கு பயன்படுத்தப்படுகின்றன மற்றும் இறுதி முடிவுகள் பெறப்படுகின்றன. இந்த செயல்திட்டத்தில், SSD மற்றும் MobileNet போன்ற கருவிகளைப் பயன்படுத்தி பொருள் கண்டறிதல் மற்றும்

கண்காணிப்பு செயல்படுத்தப்பட்டுள்ளது. ஒரு தொகுப்பிலிருந்து சில வகுப்புகளை (எ.கா: டெடி பியர், பந்து, படுக்கை போன்றவை) அடையாளம் காணவும், அவை உண்மையான நேரத்தில் ஒரு சட்டத்துடன் செல்லும்போது அவற்றைக் கண்காணிக்கவும் வெளியீட்டை உள்ளடக்கும் என்று நாங்கள் முன்மொழிகிறோம்.

இதனுடன், வெவ்வேறு தனித்துவமான கண்டறிதல் மற்றும் கண்காணிப்பு பயன்பாடுகளைச் சுற்றியுள்ள வெவ்வேறு நிரல்களுக்கு இடையில் மாறுவதை அனுமதிக்க கை அங்கீகாரம் அமைப்பு திட்டமிடப்பட்டுள்ளது.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF FIGURES	VI
	LIST OF ABBREVIATIONS	VIII
1	**INTRODUCTION**	1
2	**LITERATURE SURVEY**	3
3	**METHODS**	
	3.1 Background Subtraction	6
	3.2 Frame Differencing	7
	3.3 Optical Flow	8
	3.4 Canny Edge Detection	8
	3.5 Face and Eye detection	10
	3.6 Finger Count Recognition	12
	3.7 Color Masking	14
	3.8 Color Tracking for Drawing	15
	3.9 Mouse Tracking	15
	3.10 Object Detection and Tracking	17

4	**SIMULATION AND RESULTS** 4.1 Background Subtraction 4.2 Frame Differencing 4.3 Optical flow 4.4 Canny Edge Detection 4.5 Face Eye Detection 4.6 Finger Count Recognition 4.7 Color Masking 4.8 Color Tracking 4.9 Mouse Tracking 4.10 Object Detection and Tracking	 19 20 21 22 22 23 23 24 25 26
5	**CONCLUSION**	28
	REFERENCES	29

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.0	MATLAB Methods	5
3.1	Background Subtraction Work Flow	6
3.2	Frame Difference Work Flow	7
3.3	OpenCV Work Flow	9
3.5	Face detection Work Flow	10
3.6	Finger Count Recognition Work Flow	12
3.7	Color Masking Work Flow	14
3.9	Mouse tracking Work Flow	16
3.10	Object Detection and tracking Work Flow	17
4.1.1	Background Subtraction Model	19
4.1.2	Simulated Output (Background Subtraction)	19
4.2.1	Frame Difference Model	20
4.2.2	Simulated Output (Frame Difference)	20
4.3.1	Optical Flow Model	21
4.3.2	Simulated Output (Optical Flow)	21
4.4	Simulated Output (Canny Model)	22
4.5	Face Eye Detection Model	22
4.6	Finger Count Model	23
4.7	Simulated Output (Blue color Mask)	23
4.8.1	Simulated Output (Color Tracking)	24
4.8.2	Simulated Output (With terminal information)	25
4.9	Simulated Output (Mouse Tracking)	25
4.10.1	Simulated Output (Multiple Object detection)	26

4.10.2	Simulated Output (Blurring important Credentials, e.g. ID card)	27

LIST OF ABBREVIATIONS

BGR	Blue, Green, Red
FPS	Frames per Second
HSV	Hue, Saturation, Value
IP WEBCAM	Internet Protocol Web Camera
OpenCV	Open Source Computer Vision Library
SSD	Single Shot Detector

CHAPTER I
INTRODUCTION

Visual surveillance in dynamic business environments attempts to detect, track, and recognize objects of interest from videos, and more generally to interpret object behaviors and actions. Visual surveillance is an increasingly significant method for organizations that seek to safe guard physical and capital assets. At the same time, the necessity to observe more people, places, and things coupled with a desire to pull out more useful information from video data is motivating new demands for scalability, capabilities, and capacity. These demands are exceeding the facilities of traditional analog visual surveillance approaches. Remote and mobile monitoring, operations monitoring, traffic monitoring, public safety, social distance tracking, human computer interaction etc. are some of the applications of these systems.

The conventional methods for object detection and tracking comprise of background subtraction, frame differencing, optical flow, kernel, silhouette tracking etc. The rationale in the background subtraction 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". In frame differencing, the difference between successive frames is taken and the subsequently displayed. Considering a live video stream, the difference between successive frames provides a lot of information. Optical flow is the distribution of the apparent velocities of objects in an image. By estimating optical flow between video frames, one can measure the velocities of objects in the video.

Unfortunately, there are many drawbacks to this conventional approach which include the inability to detect when there are sudden/ drastic lighting changes, relatively many extraction parameters which must be chosen intelligently, difficulty in avoiding detection of non-stationary background objects (like swinging leaves, rain, snow), internal background model lacking capacity to react quickly to changes in background (like starting and stopping of vehicles), and accuracy of frame differencing being highly dependent on object speed and frame rate. The above-mentioned conventional methods have been implemented in MATLAB and results were obtained. Many MATLAB inbuilt image processing functions and commands allow for ease of use in this process. Tools like MobileNet, SSD along with OpenCV were used for multiple tracking systems along with the corresponding documentations.

MobileNet is a lightweight deep neural network architecture designed for mobiles and embedded vision applications. In many real-world applications such as a self-driving car, the recognition tasks need to be carried out in a timely

fashion on a computationally limited device. To fulfill this requirement, MobileNet was developed in 2017. The core layers of MobileNet is built on depth-wise separable filters. The first layer, which is a full convolution, is an exception. Around 2016, single shot detector was developed by Google Research team to cater the need for models that can run real-time on embedded devices without a significant trade-off in accuracy. Single shot object detection or SSD takes one single shot to detect multiple objects within the image. The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes. It's composed of two parts: extracting feature maps and applying convolution filter to detect objects. SSD is designed to be independent of the base network, and so it can run on top of any base networks such as VGG, YOLO, MobileNet.

These tools help to assist human operators by implementing intelligent visual surveillance systems which help in detecting and tracking suspicious or unusual events in the video sequence. Especially for the increasing interest in building autonomous mobile systems, the detection and tracking of moving objects in natural scenes is a very important and challenging task. In the context of mobile systems, such algorithms cannot be limited to static cameras. In addition, real-time constraints must be satisfied. The task must be solved in a closed loop between image acquisition and reaction (for example, camera movement to pursuit the moving object).

The general motive of object detection in this project is to recognize and locate (localize) all known objects in a scene. This project aims to automatically compute the class of objects detected in a frame and track them live as they move around while maintaining accuracy and speed. Certain classes containing sensitive details like identity cards should be identified and automatically blurred for user's privacy and safety. MobileNet, SSD, OpenCV etc. are important tools that will help in the working of the project to build the necessary implementations described above.

CHAPTER 2
LITERATURE SURVEY

As visual recognition and computer vision have matured as a science, the industry has created an ever-increasing demand for such products and services. From traditional areas of machine inspection, commercial vision applications have expanded into video surveillance, medical image analysis, face detection and recognition and many others. Most researchers show greater interest in building different adaptive background models in order to reduce the influence of dynamic scene changes on motion segmentation.

From the early studies given by Karmann et al., 1990 [6] and Kilger, 1992 [7], it can be concluded that an adaptive background model based on kalman filtering was proposed to adapt temporal changes of weather and lighting. Rowley et al., 1997 [15] focused on the segmentation of optical flow fields of articulated objects. From their work, it can be observed that major contributions were to add kinematic motion constraints to each pixel, and to combine motion segmentation with estimation in expectation maximization (EM) computation. In Bregler, 1997 [3], it can be observed that each pixel was represented by its optical flow. These flow vectors were grouped into blobs having coherent motion and characterized by a mixture of multivariate Gaussians.

Liang et al., 2003 [8] in their works on Optical flow used to detect moving objects even in the presence of camera motion. Most optical flow computation methods are computationally complex and very sensitive to noise, and cannot be applied to video streams in real-time without specialized hardware. Viola et al., 2004 [20], presented a multistage detection method namely the Haar cascade classifier. The idea of Haar cascade involves extracting features from images using a kind of ‘filter’, similar to the concept of the convolutional kernel. Inspired by Viola et al., 2004 works, many people tried to construct a cascade classifier [12],[25],[13],[10],[17],[16],[11]. From these works, it can be concluded that the classifiers in the early days are simpler than their later versions.

The authors Wei et al., 2016 [22] have used an approach, named SSD, which discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. It has been observed that a key feature of their SSD model is the use of multi-scale convolutional bounding box outputs attached to multiple feature maps at the top

of the network. They have suggested that their representation allows them to efficiently model the space of possible box shapes. Apart from its standalone utility, the relatively simple SSD model provides a useful building block for larger systems that employ an object detection component.

Ben Ayed et al., 2015 [2] proposed a method for detection of text data based on a texture in video frames by big data analytics. The video frames are decomposed into various fixed size blocks and these blocks are analyzed using har wavelet transform technique. Further, a neural network was used to classify the text and non-text blocks. However, this study needs to concentrate on extracting the regions towards remove the noisy regions as well as exclude the text like sections.

Soundrapandiyan et al., 2015 [18] suggested a novel and adaptive method for pedestrian detection. Further, the foreground separated objects from the background by image pixel intensities. Subsequently, they used high boost filter for enhancing the foreground edges. The efficacy of the proposed method is evident from the subject evaluation results as well as objective evaluation with around 90% of pedestrian's detection rate compared to the other single image existing methods. In future, they planned to improve the performance of the method with higher detection rate and low false positives on par with sequence image methods.

Ramya et al., 2016 [14] suggested a modified frame difference method which uses the correlation between blocks of current image and background image to categorize the pixels as foreground and background. The blocks in the current image that are highly correlated with the background image are considered as background. For the other block, the pixel-wise comparison is made to categorize it as foreground or background. The experiments conducted proved this approach improves the frame difference method particularly as finding accuracy with speed. However, this study needs to focus towards other information available in the blocks such as shape and edge can be used to improve the detection accuracy.

CHAPTER 3

METHODS

Two methods of detection and tracking have been implemented: MATLAB and OpenCV. In this project, object detection was implemented using MATLAB techniques. Frame differencing is an important algorithmic technique that observes the motion in an image scene and detects the moving objects using a fixed surveillance camera. The outputs obtained provide conclusive evidence for detection and tracking. The difficulties experienced using MATLAB for accessing the live webcam were overcome in due course of the project with the use of OpenCV as a tool for further implementation of the project. In using this tool and its documentation, the two major components involving the detection and tracking system were found to be: target representation and localisation AND filtering and data association. The basic techniques being implemented using MATLAB include background subtraction, frame differencing, optical flow, canny edge detection.

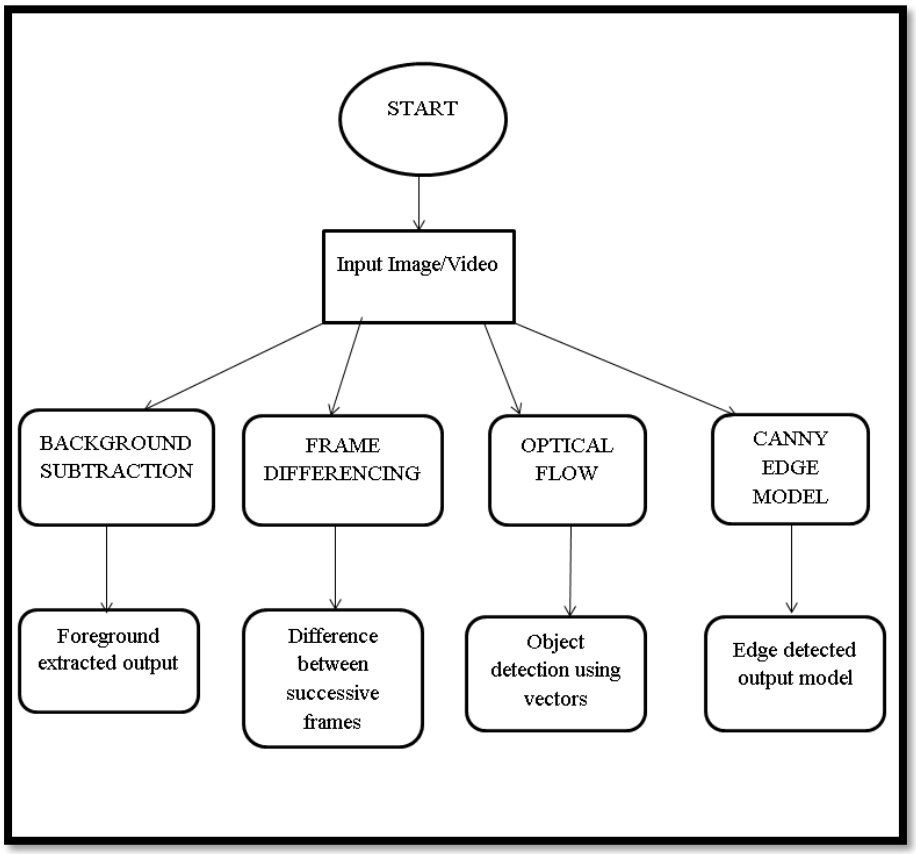

Figure 3.0 MATLAB Methods

3.1 BACKGROUND SUBTRACTION

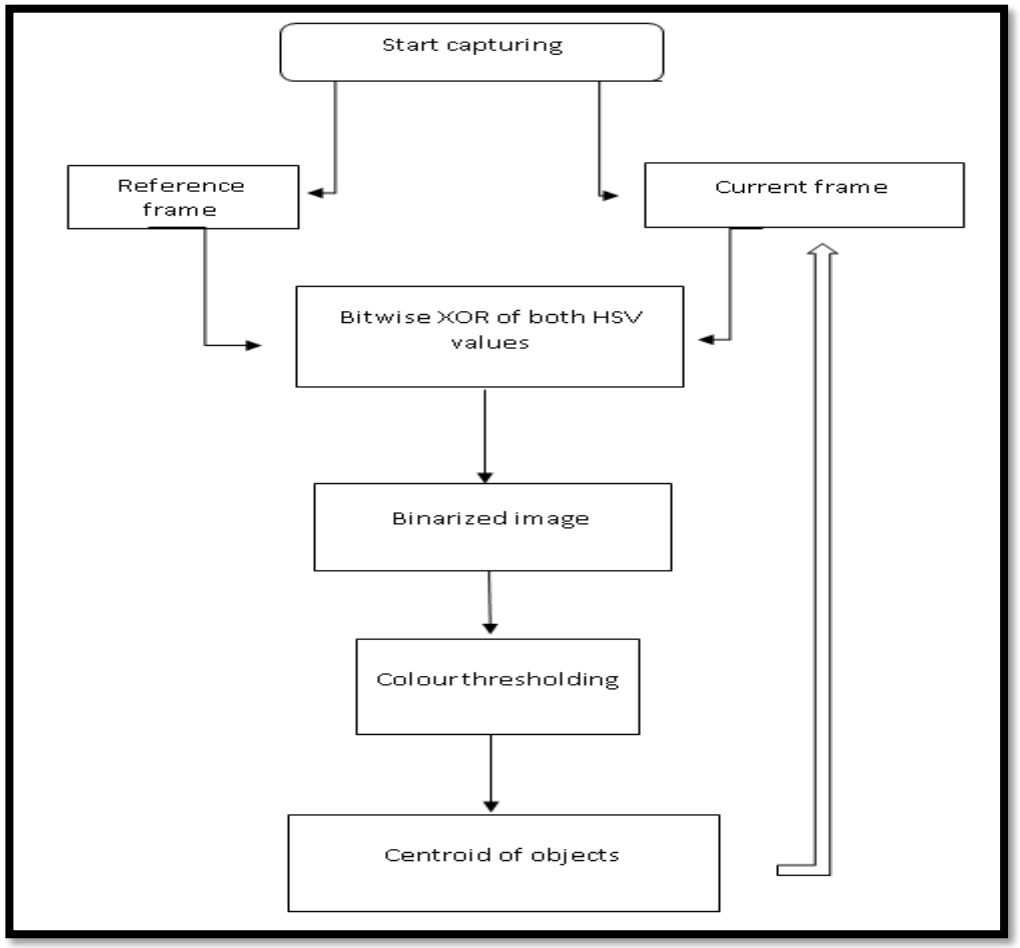

Figure 3.1 Background Subtraction Work Flow

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 pose estimation. Background subtraction is generally based on a static background hypothesis which is often not applicable in real environments.

3.2 FRAME DIFFERENCING

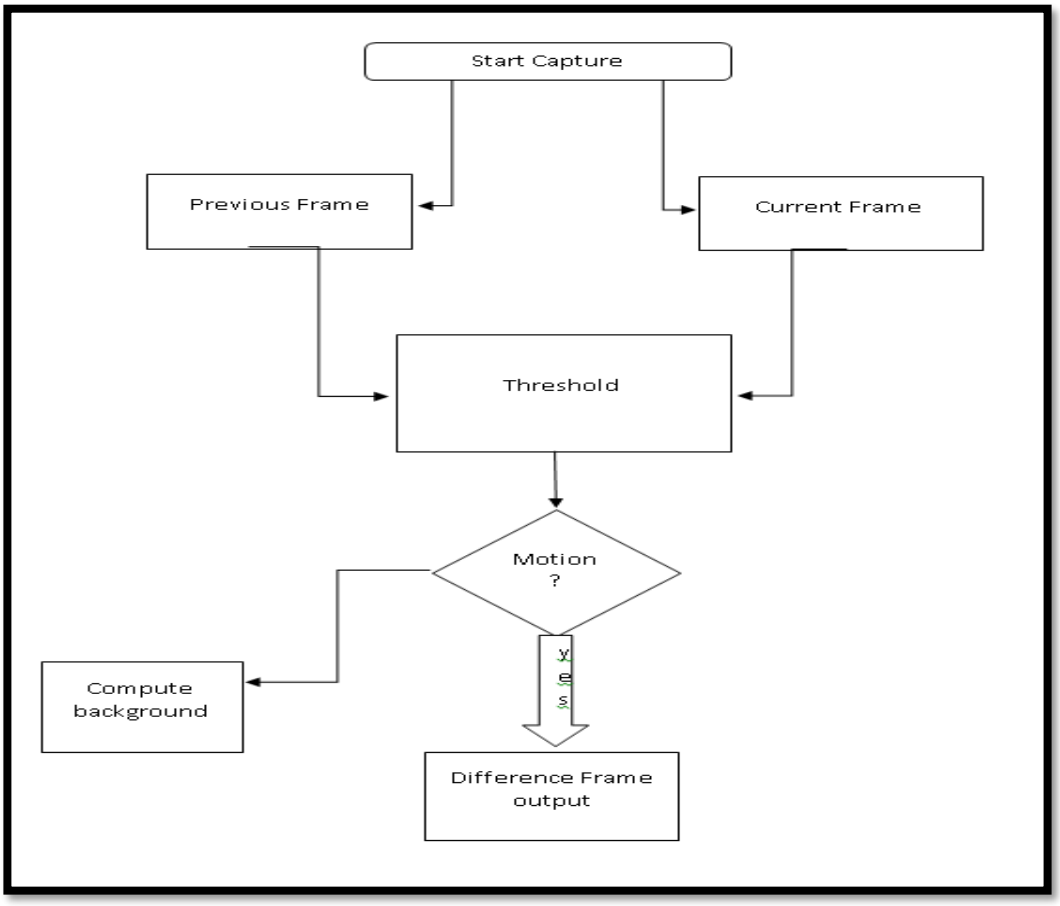

Figure 3.2 Frame Difference Work Flow

It is known that a static background image cannot be used to detect objects. So, one of the ways to fix this would be to use frame differencing. It is one of the simplest techniques that is used to see what parts of the video are moving. Considering a live video stream, the difference between successive frames gives a lot of information. The concept is fairly straightforward. The difference between successive frames is taken and the difference is displayed. Most techniques work with some blur and threshold, to distinct real movement from noise. Because frame could differ too when light conditions in a room change (and camera auto focus, brightness correction etc.).

3.3 OPTICAL FLOW

Optical flow is the distribution of the apparent velocities of objects in an image. By estimating optical flow between video frames, one can measure the velocities of objects in the video. In general, moving objects that are closer to the camera will display more apparent motion than distant objects that are moving at the same speed. Optical flow estimation is used in computer vision to characterize and quantify the motion of objects in a video stream, often for motion-based object detection and tracking systems.

3.4 CANNY EDGE DETECTION

In an image, an edge is a curve that follows a path of rapid change in image intensity. Edges are often associated with the boundaries of objects in a scene. Edge detection is used to identify the edges in an image.
To find edges, the edge function is used. This function looks for places in the image where the intensity changes rapidly, using one of these two criteria:

- Places where the first derivative of the intensity is larger in magnitude than some threshold
- Places where the second derivative of the intensity has a zero crossing

Edge provides several derivative estimators, each of which implements one of these definitions. For some of these estimators, one can specify whether the operation should be sensitive to horizontal edges, vertical edges, or both. Edge returns a binary image containing 1's where edges are found and 0's elsewhere.

The most powerful edge-detection method that edge provides is the Canny method. The Canny method differs from the other edge-detection methods in that it uses two different thresholds (to detect strong and weak edges), and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be affected by noise, and more likely to detect true weak edges.

For MATLAB: The above shown works were done using MATLAB. There are some demerits of implementing the object detection and tracking using MATLAB. MATLAB is more compatible with avi files (.avi – Audio Video Interleave format) which isn't real-time video and thus accessing a live webcam becomes difficult. Moreover, accessing the live webcam through MATLAB is a tedious process, which reduces efficiency and as a consequence increases the overall run-time of the program. Hence, OpenCV Python was opted for further implementation of the project.

The basic techniques of object detection and tracking with OpenCV are done and using the following concept ideas and the final tracking system was implemented on this. In OpenCV python, **fps** is a term (expanded as frames per second) that is often used to describe the total number of frames that the webcam receives per second. When this number is under 30, the ability to identify and track objects is much more efficient.

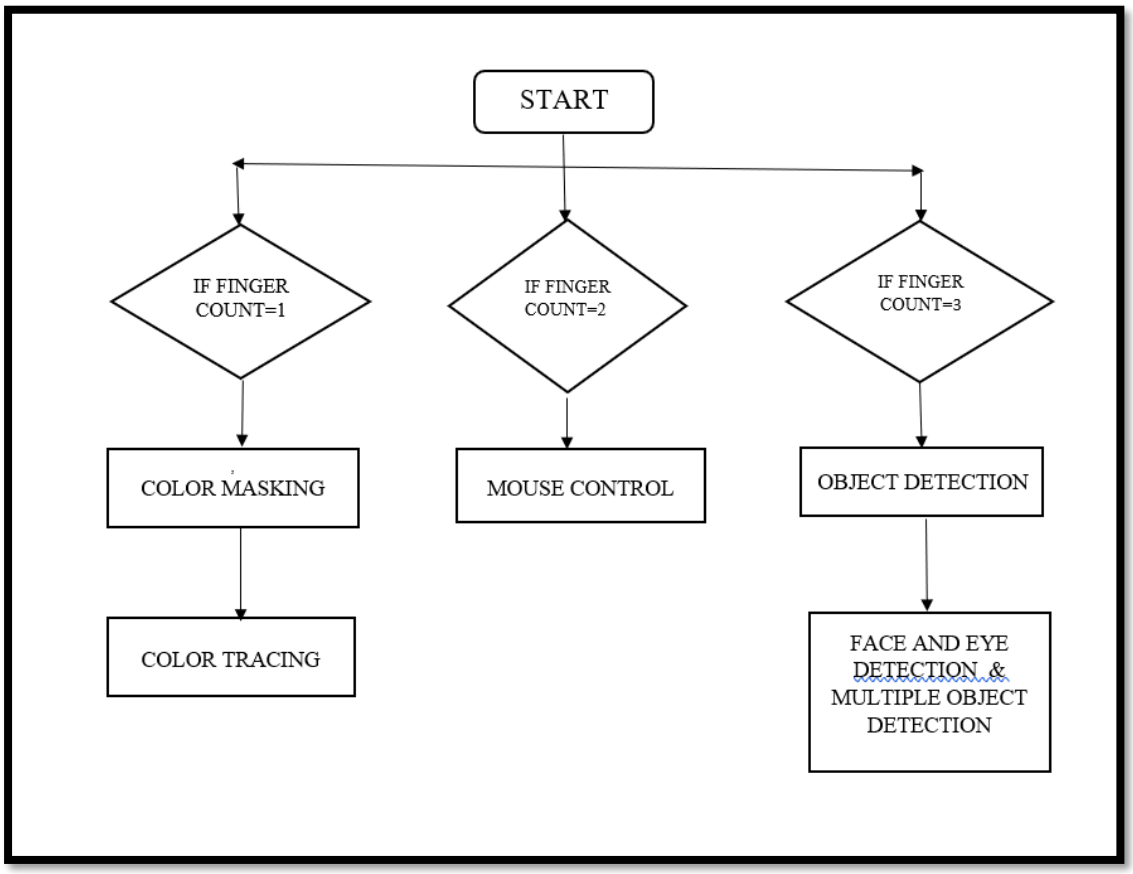

Fig 3.3 OpenCV Work Flow

This above diagram depicts the overall flow of the project. In the main program, depending on the finger count recognized, the program is redirected to a corresponding operation.

3.5 FACE AND EYE DETECTION

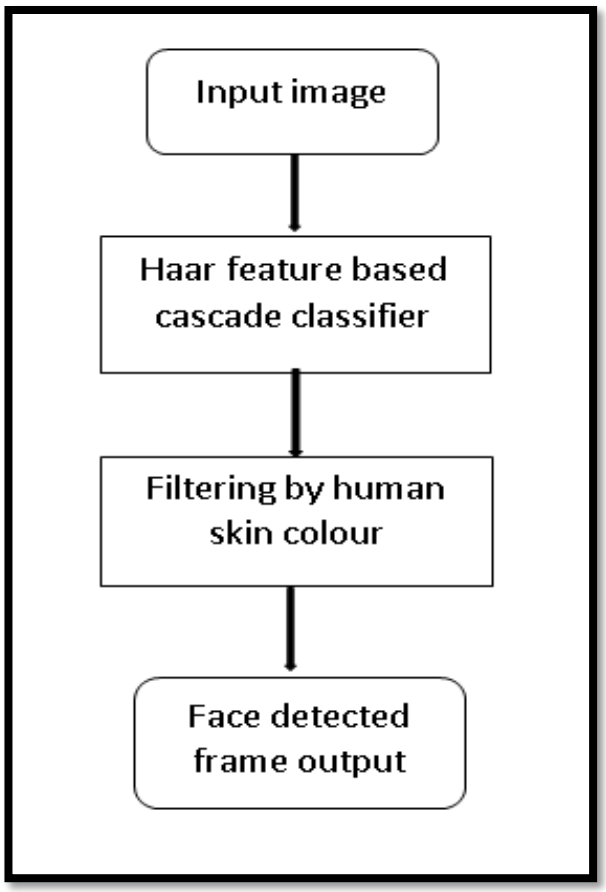

Figure 3.5 Face Detection Work Flow

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face-detection algorithms focus on the detection of frontal facial features of humans. It is analogous to image detection in which the image of a person is matched bit by bit.

Here, using OpenCV library functions, (inbuilt face detection algorithm (Haar Classifier)) the facial and eye features (both moving & stationary) were detected and the outputs were recorded. The detection process lags during changing of profile or varying of line of sight. The detectMultiScale() function is a general function that detects objects in a varied scalable window. Since face cascade is called, the program detects the relevant facial features.

```
#Detect faces in the image

faces = faceCascade.detectMultiScale (
   gray,
   scaleFactor=1.1,
   minNeighbors=5,
   minSize=(30, 30),
   flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

This function detects the actual face and is a key part of the code.

- The detectMultiScale function is a general function that detects objects.
- The first option is the grayscale image.
- The second is the scale Factor. Since some faces may be closer to the camera, they would appear bigger than the faces in the back. The scaling factor compensates for this.
- The detection algorithm uses a moving window to detect objects. minNeighbors defines how many objects are detected near the current one before it declares the face found.
- minSize gives the size of each window.

3.6 FINGER COUNT RECOGNITION

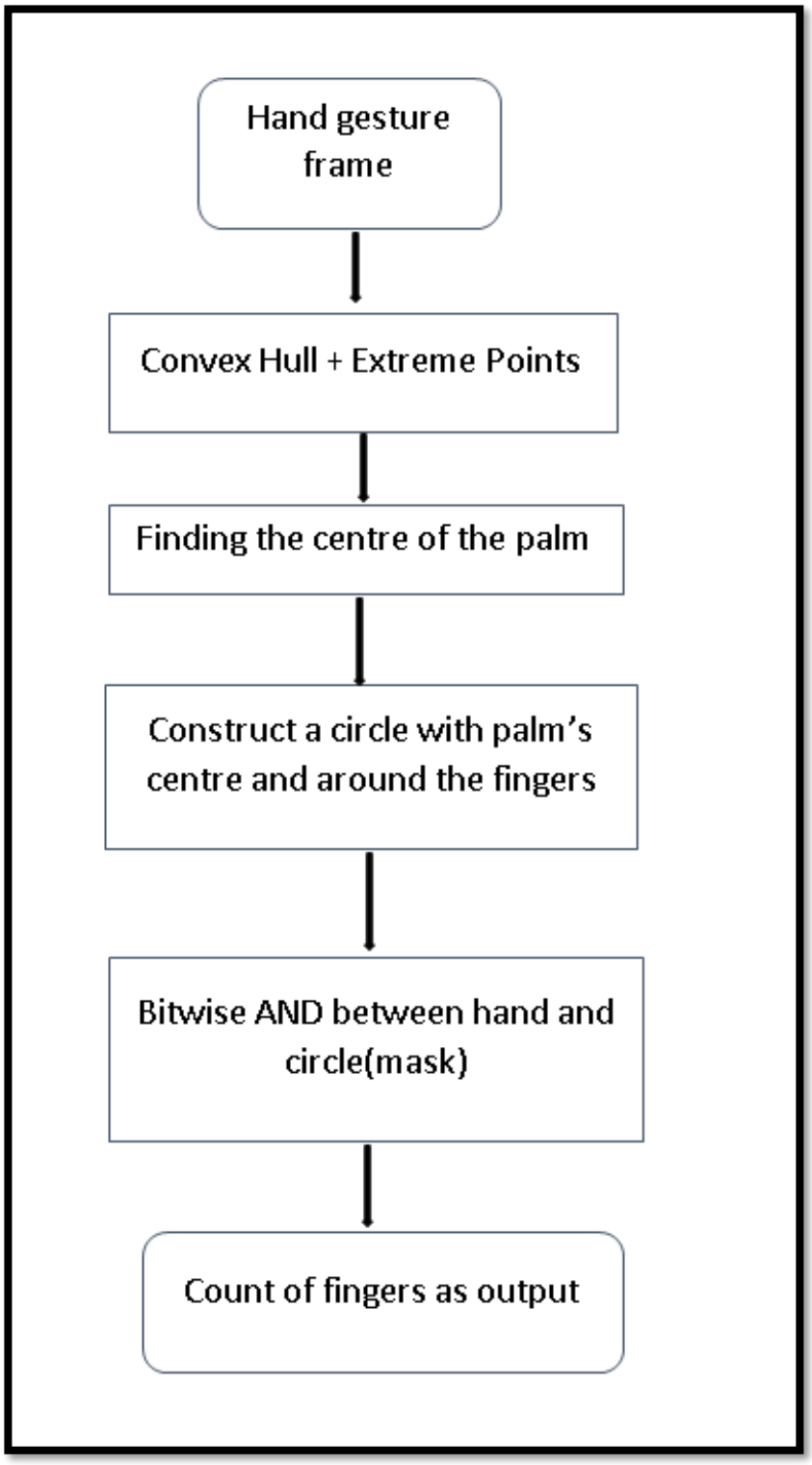

Figure 3.6 Finger Count Recognition Work Flow

This is a system which detects a human hand, segments the hand using

thresholding, counts the number of fingers being held up and displays the finger count, all from a live video input.

- Begin by creating a region of interest in a live video frame, where the hand is to be inserted for counting.
- Once the hand is detected, it is isolated by applying thresholding techniques, Binary Thresholding in this case using OpenCV.
- To the thresholded hand segment, a polygon is drawn around the hand to identify the extreme points, using Convex Hull technique.
- Using the intersection of the extreme points in the polygon (Top, Bottom, Left, Right), the center of the hand is calculated.
- The point furthest away from the center of the hand is identified and a ratio of that distance say 80% is used as the radius for a circle, which is drawn around the center of the hand (For visualization purposes, we may think of this as the palm region).
- Any points outside the circle and far away from the bottom are concluded to be the extended fingers.
- Return the total count of these identified points.

Using the knowledge acquired from the above shown implementation of Finger Count Recognition (Hand Gesture Recognition), a system which redirects to different codes based on the number of fingers shown was developed.

For example, showing one finger in front of the webcam will redirect to color masking section and the same program will be run. Showing two fingers will redirect to object detection section, which enables a person to perform object detection and tracking using a remote mobile phone camera. This process allows the entire system to be performed in one motion.

This implementation paves a whole new way for human-computer interaction which is conventionally done using input devices such as keyboard, mouse, etc.

The section below deals with some implementations which aim to overcome the mouse, keyboard etc. functionalities'.

3.7 COLOR MASKING

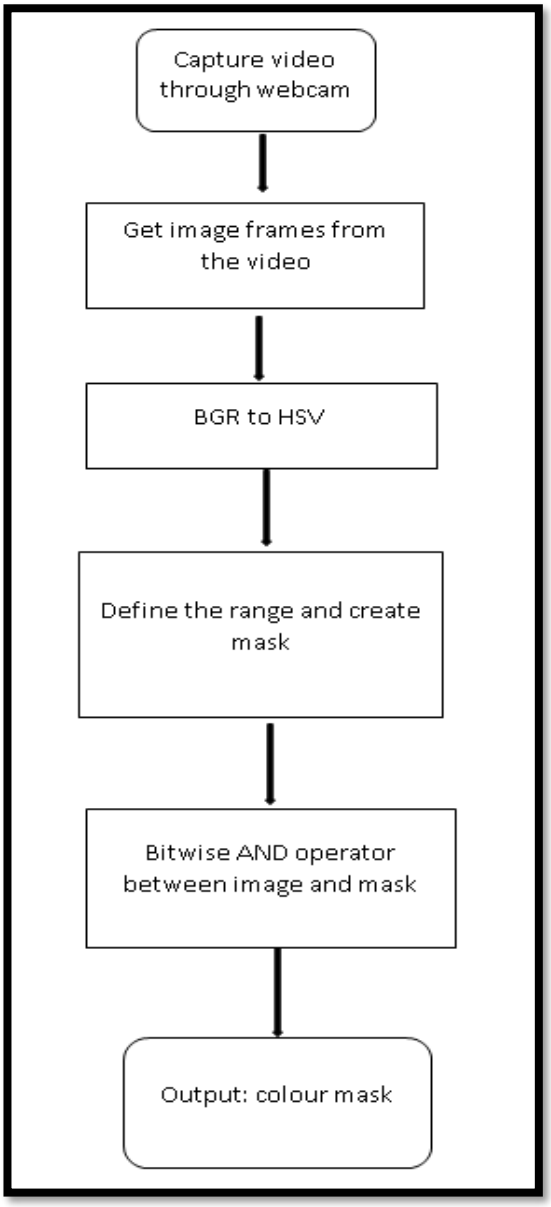

Figure 3.7 Color Masking Work Flow

Color masking allows for finer control of the color values of any given input frame. By limiting the color channels using code, different segmentations can be made that each store a different grayscale image. This technique helps to edit random colors by applying masking techniques and thus limiting the range of displayed colors to specific shades.

For color segmentation, the requirement is the threshold values or the knowledge of the lower bound and upper bound range of colors in one of the color spaces. It works best in the Hue-Saturation-Value color space. After specifying the range of color to be segmented, a mask is generated and a

particular region of interest can be separated out. This can then be used further to analyze specific details within the image at a greater scope than was previously possible.

3.8 COLOR TRACKING FOR DRAWING

To detect colors in images, the upper and lower HSV bounds must be defined. Once upper and lower limits are defined, a call to the cv2.inRange can be made which returns a mask, specifying which pixels fall into the specified upper and lower range. Then the mask is obtained and it can be applied to area of interest using the cv2.bitwise_and function.

To impose the mask on top of the original image, the function cv2.bitwise_and() is used which keeps every pixel in the given image if the corresponding value in the mask is 1. The color is then followed as it moves across the frame to create a smooth colorful motion and the output can be screenshot with a spacebar command to capture the drawing at that instant.

Color image segmentation that is based on the color feature of image pixels assumes that homogeneous colors in the image correspond to separate clusters and hence meaningful objects in the image. In other words, each cluster defines a class of pixels that share similar color properties. As the segmentation results depend on the used color space, there is no single color space that can provide acceptable results for all kinds of images.

3.9 MOUSE TRACKING

It is a mouse simulation system that performs all the normal functions of a handheld mouse corresponding to regular hand movements and gestures. A camera captures frames and depending on distance between the two blue markers, one can move the cursor and perform click/drag functions. This includes left click, right click, drag, select and scroll up and down. The predefined gestures make use of color identification, tracking and distance capture.

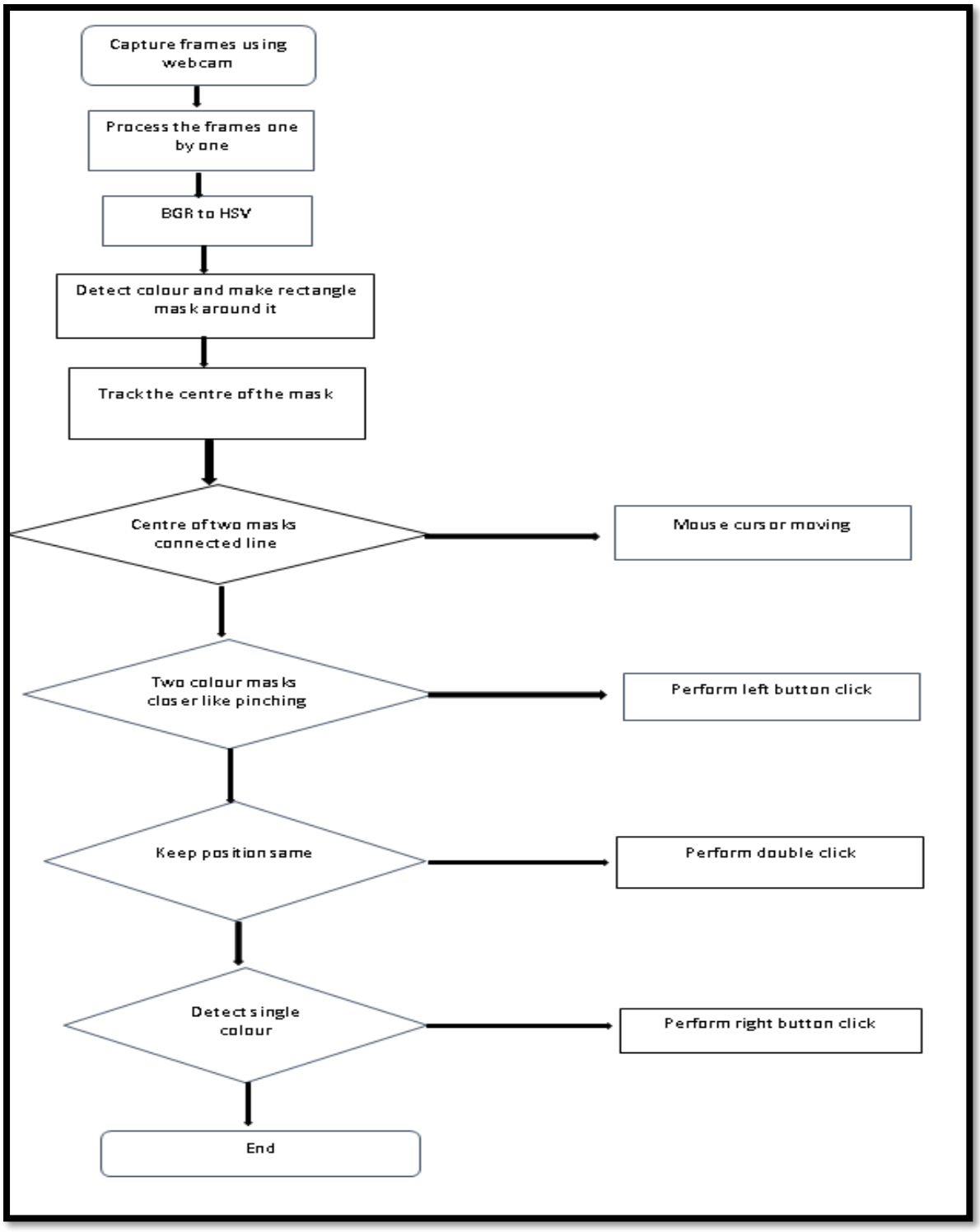

Figure 3.9 Mouse Tracking Work Flow

The variables (camx, camy) as well as other objects are essential. The Mouse object is useful for general mouse movements wx app allows the fucntion wx.GetDisplaySize() to be used which gets the screen resolution. The variables camx, camy are used to set the captured image resolution. These variables will be used later in image resizing as well.

FINAL IMPLEMENTATION

3.10 OBJECT DETECTION AND TRACKING

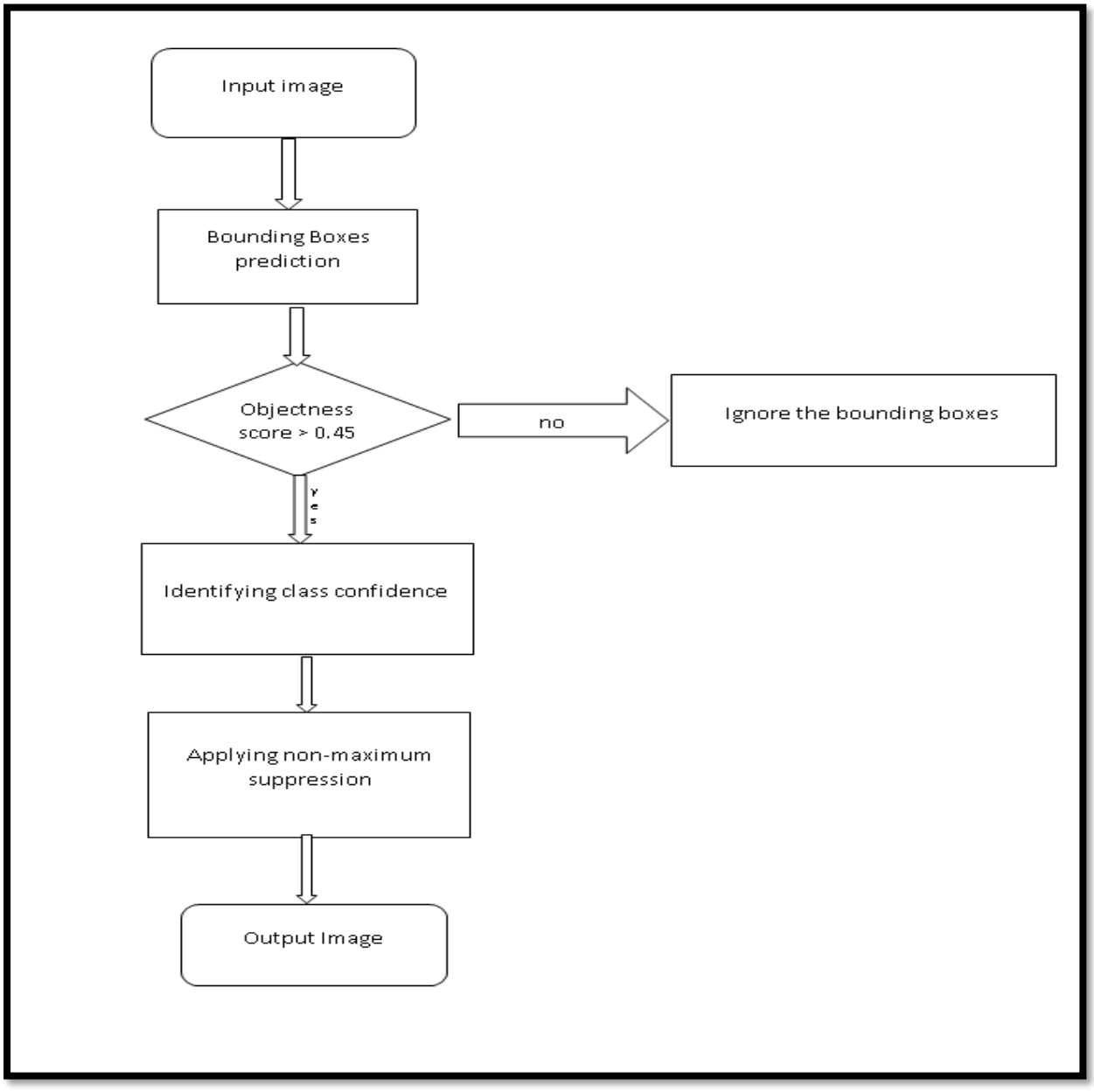

Figure 3.10 Object Detection and Tracking Work Flow

By combining the above basic techniques, the final detection and tracking system is implemented. With the help of an IP mobile webcam, live tracking becomes efficient and simple. Additionally, the blurring occurs when any

important credentials are shown, thereby increasing security and safety.

The ultimate aim of this system is:

- To identify the location of objects that are present in the image, their class and constantly track them across the frame.

This system is able to identify different objects present in the frame with incredible accuracy using caffe models. This implementation can be used in helping self-driving cars navigate through traffic, spot violent behaviour in crowded place, ensure proper quality control of parts in manufacturing, etc.

The Object Detection and Tracking was done with the help of caffe models. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. Caffe supports many different types of deep learning architectures geared towards classification and image segmentation. It supports CNN, RCNN, LSTM and fully connected neural network designs.

CHAPTER IV
SIMULATION AND RESULTS

4.1 BACKGROUND SUBTRACTION:

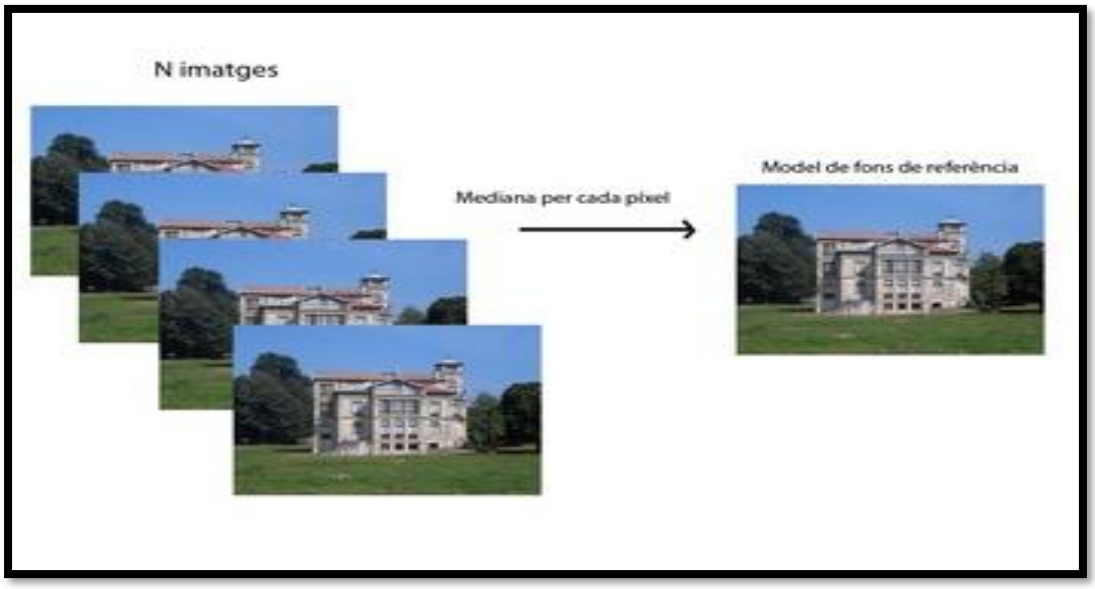

Figure 4.1.1 Background Subtraction Model

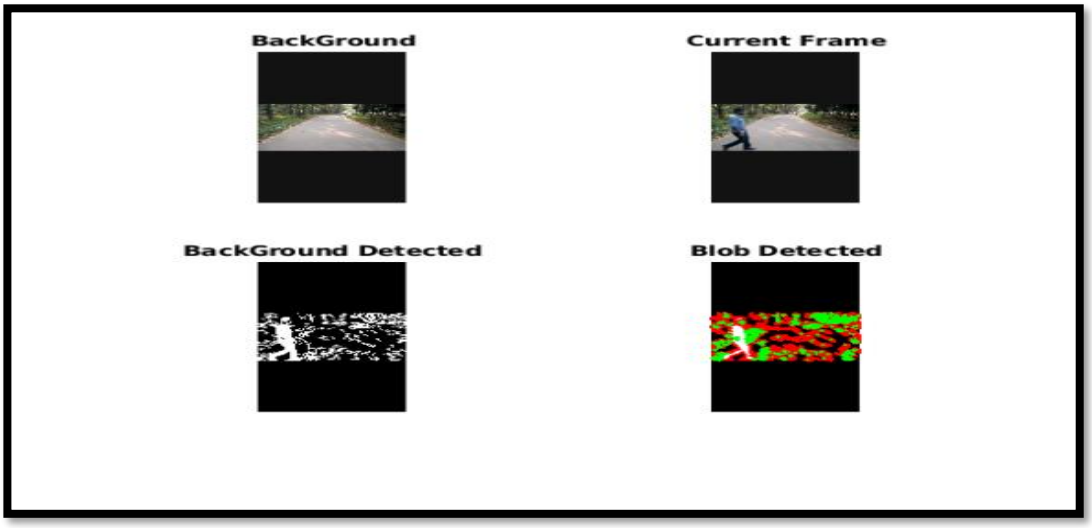

Figure 4.1.2 Simulated Output (Background Subtraction)

The output picture above shows the background, current frame, detected background and detected blob. The detected blob clearly distinguishes the background with the person in the frame i.e., the object of interest.

4.2 FRAME DIFFERENCING:

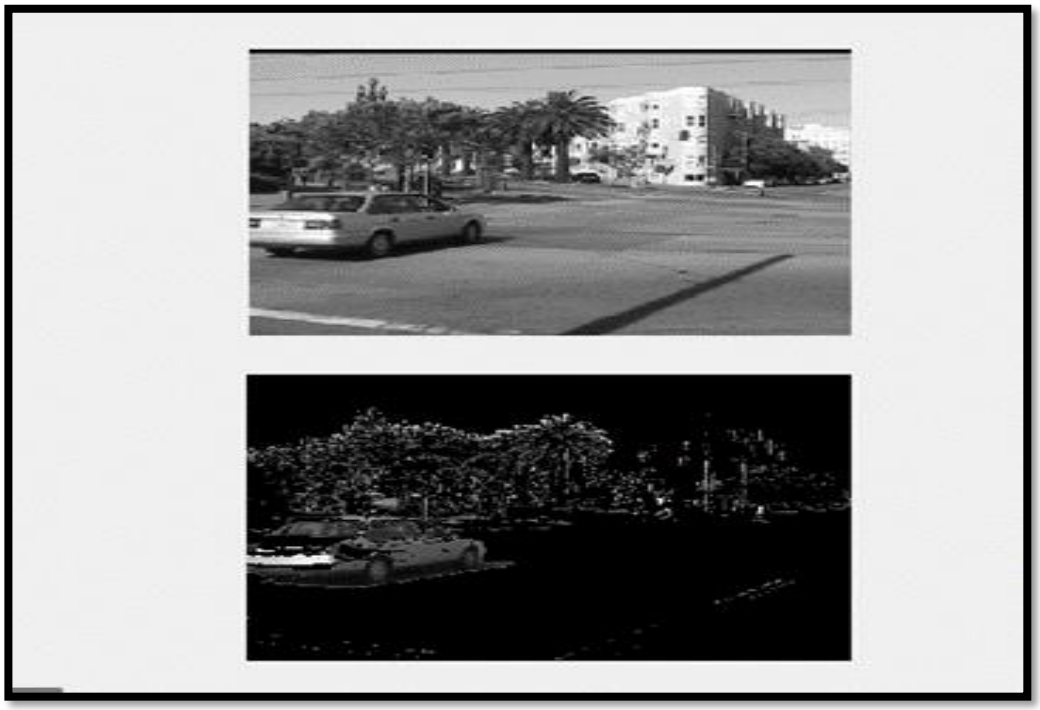

Figure 4.2.1 Frame Difference Model

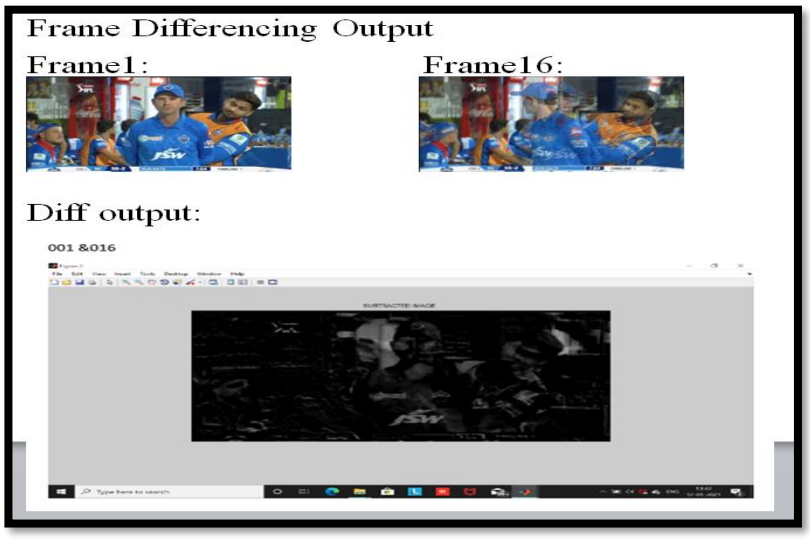

Figure 4.2.2 Simulated Output (Frame Difference Model)

The output picture above shows the difference in positions of objects that happened between frames 1 and 16 as an example. The frame differenced output gives an insight of the movements that take place in a given video stream.

4.3 OPTICAL FLOW:

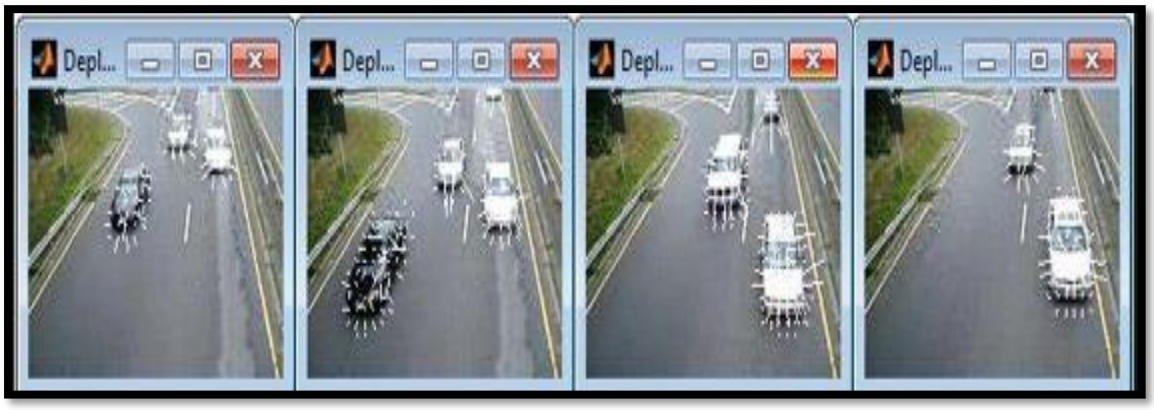

Figure 4.3.1 Optical Flow Model

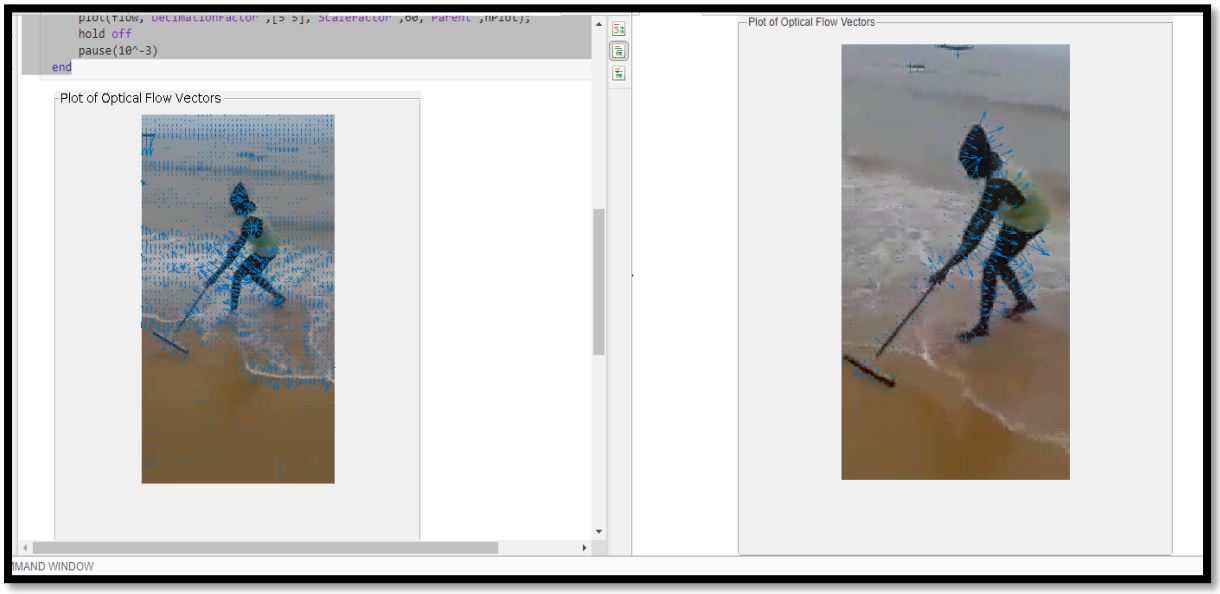

Figure 4.3.2 Simulated Output (Optical Flow)

The picture above has two consecutive frames showing difference in movements highlighted by blue vectors. This output gives the difference in movements taking place by highlighting the points that are moved in successive frames i.e., from first fame to second frame.

4.4 CANNY EDGE DETECTION:

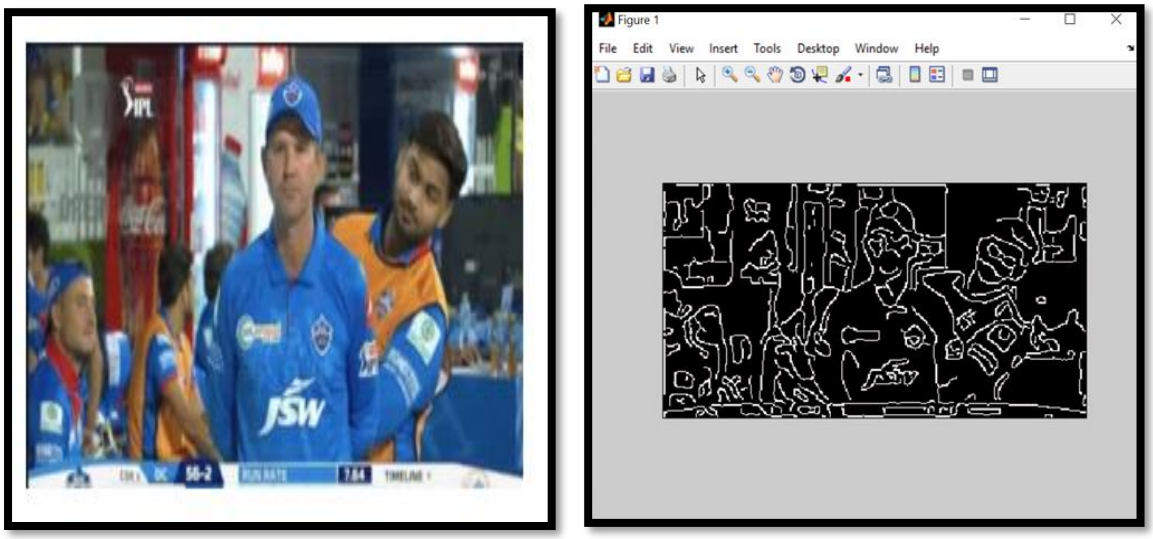

Figure 4.4.1 Simulated Output (Canny Model)

The picture on the left is the given input image and the one on the right is its corresponding edge detected image. From this output picture, it can be seen that the edges are distinguished by color gradient changes (boundaries) between two objects.

4.5 FACE EYE DETECTION:

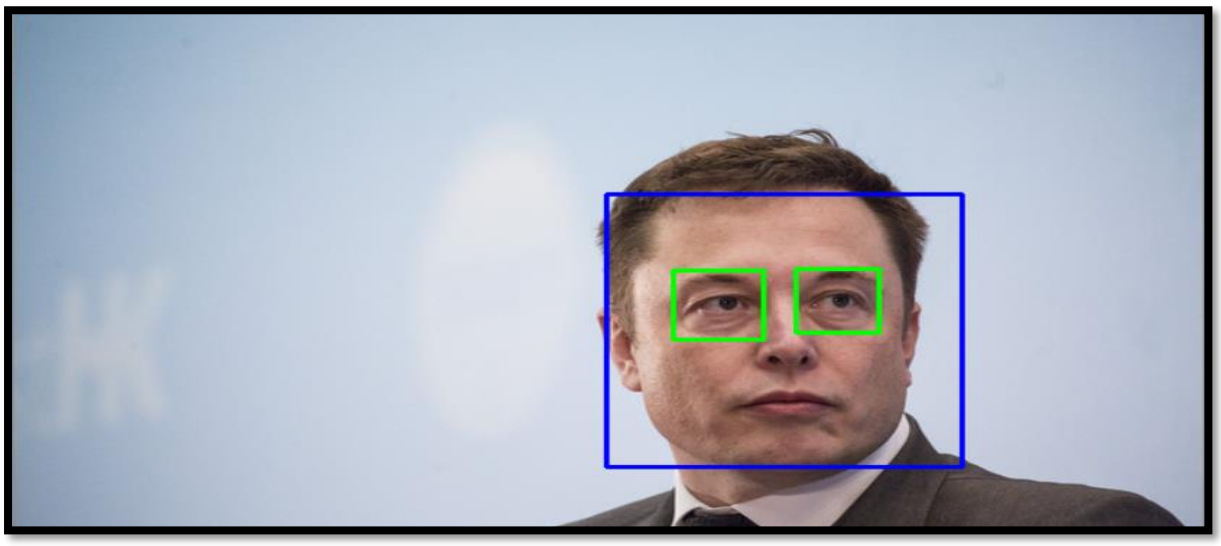

Figure 4.5 Face Eye Detection Model

The model takes an input frame or webcam/video (series of frames) and outputs the bounding box of both eyes and the face. Haar feature-based cascade

classifiers proposed by Paul Viola and Michael Jones is implemented. The eyes are in green boxes while face appears in a blue box. In live video, the bounding boxes track the moving face in real-time maintaining an accurate reading of the facial features.

4.6 FINGER COUNT RECOGNITION:

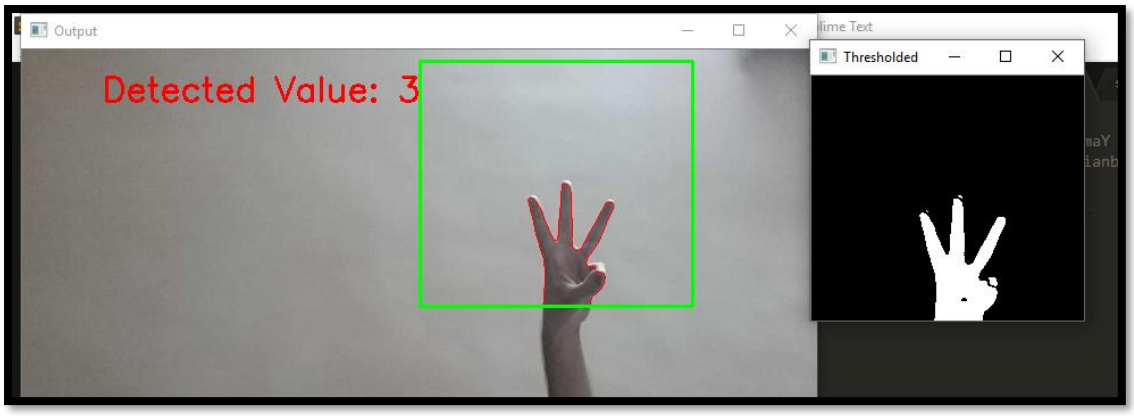

Figure 4.6 Finger Count Model

Firstly, the hand region is segmented from the live video. The segmentation is taken strictly from within the confines of the green box depicted in the window. Then convex hull algorithm is applied to the segmented region. The gray-scaled image appears in the threshold window. It outlines the hand in white while keeping the background dark. Once the code detects the number of fingers, it displays the number in a window in text form. Depending on the number recognized, the code redirects to a specific part of the program.

4.7 COLOR MASKING:

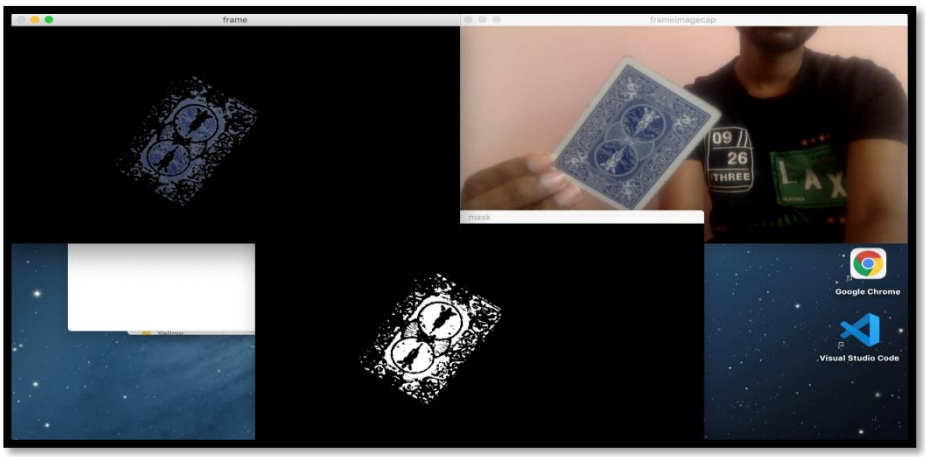

Figure 4.7 Simulated Output (Blue color Mask)

To extract colors in an image, the first thing is to convert from RGB to HSV Scale. Then one must define the upper and lower regions for the blue color to extract the playing card details. Using this, various inbuilt functions can be used to trace or follow the identified color across the frame while segmenting out the unnecessary parts.

4.8 COLOR TRACKING:

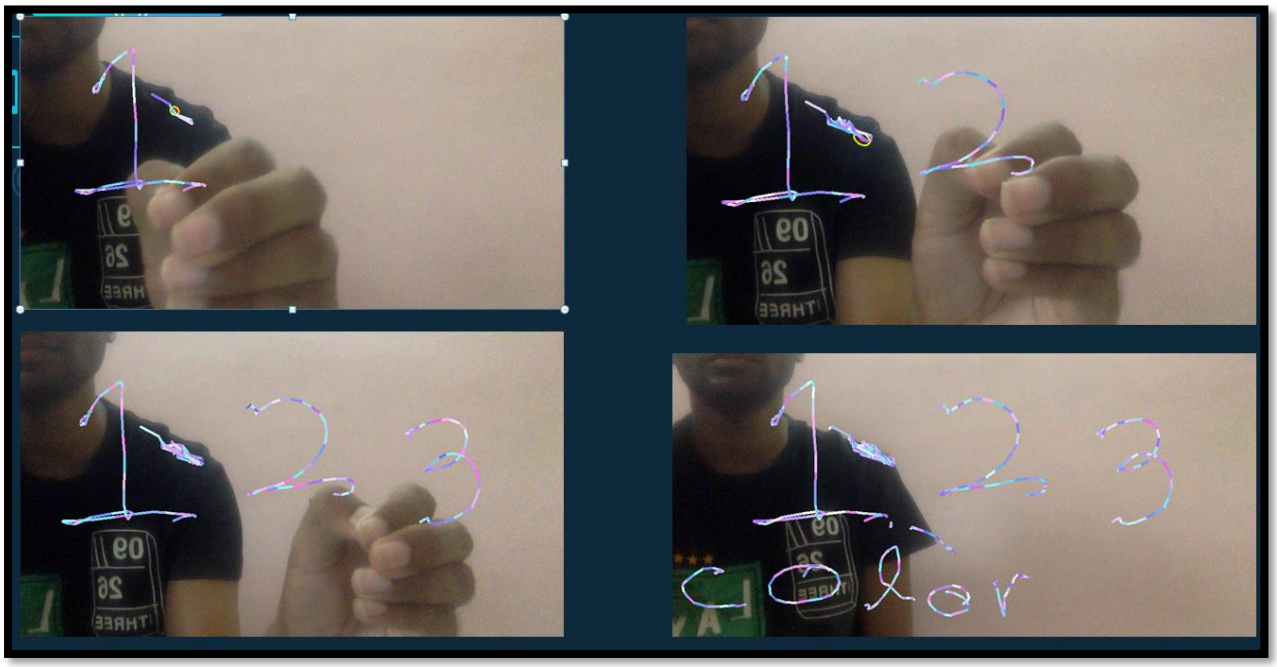

Figure 4.8.1 Simulated Output (Color Tracking)

Colour tracking involves identifying the point in the frame that matches the given specified HSV range provided in the code (in this case, colour shades of blue). The blue tip, i.e. cap of a blue pen, is followed around the frame and a colourful trace is produced as output. This appears whenever the blue point appears in the frame.

The required frames can then be screenshotted using certain commands to capture particular moments.

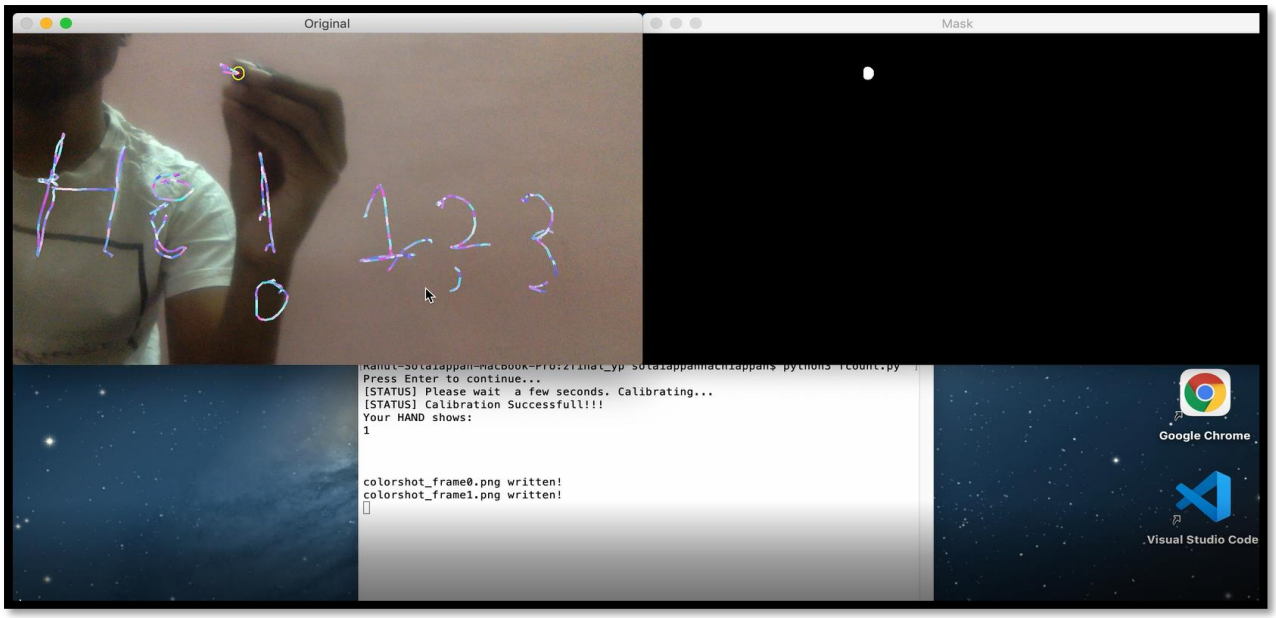

Figure 4.8.2 Simulated Output (With terminal information)

Also by pressing spacebar in the keyboard, we can store the required frame as a screenshot (.png) as shown above.

4.9 MOUSE TRACKING:

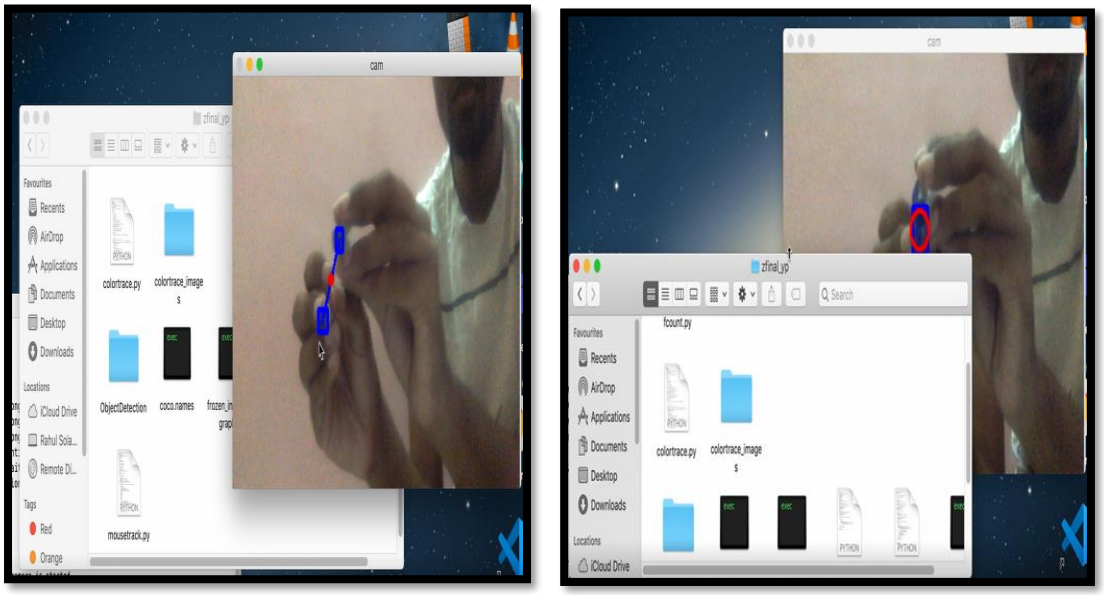

Figure 4.9 Simulated Output (Mouse Tracking)

The movements of a mouse on the laptop screen is replicated using the blue markers (2 blue pen caps) spaced at a small distance from each other. The centerpoint of these two markers forms the positioning of the mouse and when moved in conjunction, the mouse moves as well. Bringing the two blue markers closer to each other indicates a click/press which then enables movement of windows or files and allows for selection options.

4.10 OBJECT DETECTION AND TRACKING:

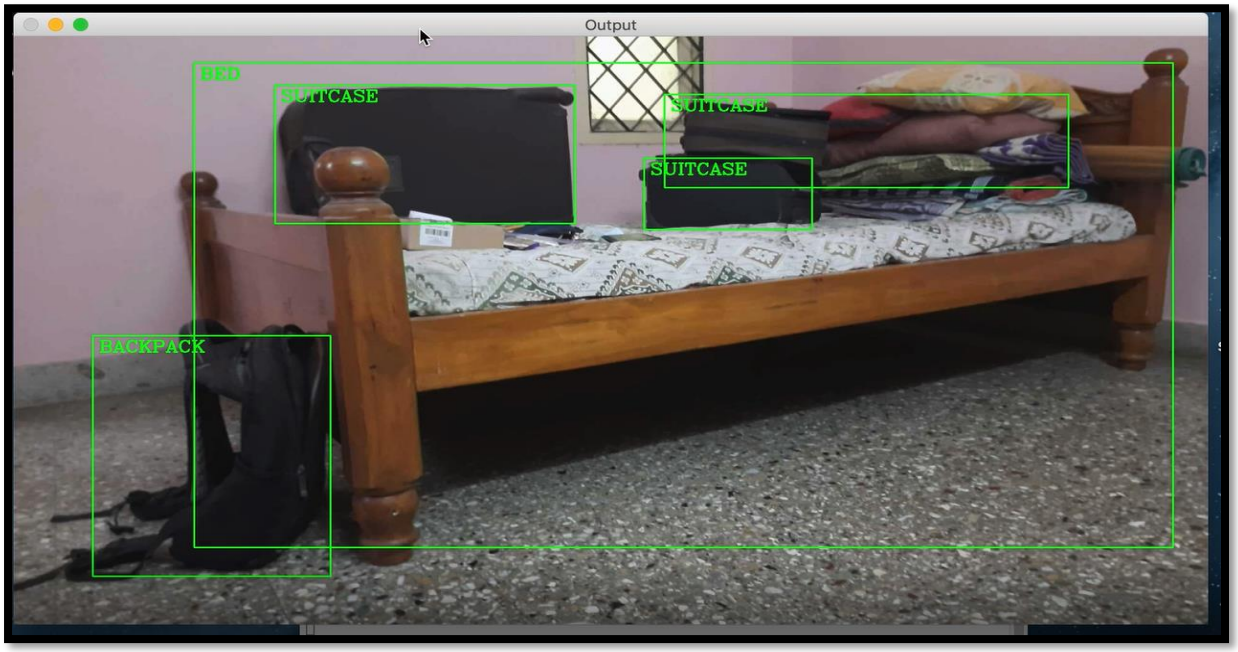

Figure 4.10.1 Simulated Output (Multiple Object detection)

Object tracking and identification is implemented using a live ip webcam on a phone which then connects to the computer that identifies distinct objects. Each unique object identified is placed in a green box and identified with a class noted on the top left side.

The picture shows the green box identifying a suitcase, bed and a backpack.

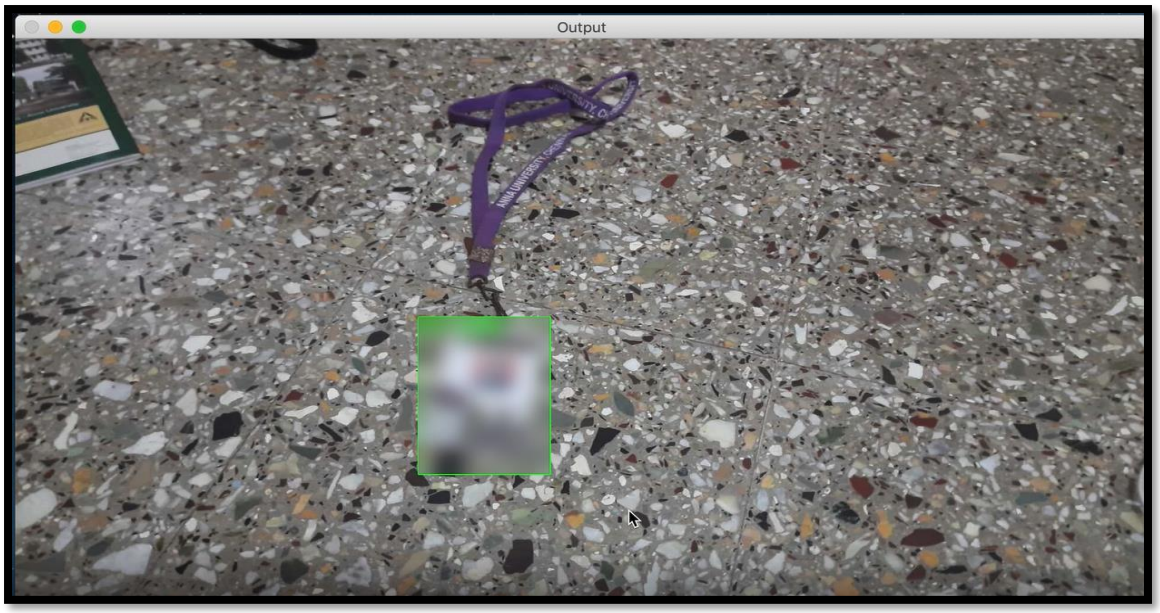

Figure 4.10.2 Simulated Output (Blurring important Credentials, e.g. ID card)

.

It also identifies cards that may contain sensitive data and automatically blurs them for user privacy and protection. In the picture shown, the college id card is detected and automatically blurred.

CHAPTER 5
CONCLUSION

Computer vision is a branch of computer science and software systems that can recognize as well as understand images and scenes. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. In this project, highly accurate object detection-algorithms were used using the help of MobileNet, SSD tools and OpenCV framework. Each and every object in a frame is detected, placed in highlighted rectangular boxes and identified by assigning it a specific tag. These processes were implemented keeping in mind a low memory capacity, efficiency and accuracy.

The project being presented has dealt with the real-time application of object tracking, i.e., live video. The colour tracking and mouse tracking systems shown proves to be a sufficient communication tool for people with disabilities. The tracking and detection system is useful for safety, security and identification systems. In our modern day world, it should be noted that many areas of object tracking is yet to be explored. With the knowledge acquired from the works shown in this report, many innovative projects and ideas are to be implemented in the view that the latter will be useful for our world.

Future implementations that could be expanded from this project based on the learned skills and techniques are playing certain types of music based on facial reactions shown or giving a signal to phone camera so it automatically takes picture after a delay of few seconds. Additionally, facial recognition could be used to print relevant details of identified subject or tracking could be used to count exercise repetitions to track daily fitness activity. A more advanced project could involve predicting future movement based on previously studied movement analysis. The capability of machines to identify suspicious objects and further identify their activities in a specific environment is an important part of permitting a machine to interact with humans in effective and easy manner.

Computer vision algorithms extract important features such as shapes, illumination, and color distributions from images and video sequences which can be used to better understand a scene automatically as done by the human visual system of eyes and brain. In other words, computer vision provides the real-time interpretation of the scene under observation, and warns if the system requires an immediate response. The project and further implementations hope to further this cause and strive towards advancing this area of science.

REFERENCES

1] Aiyun Yan, Jingjiao Li, Beibei Sun. “Research on Moving Target Tracking System Based on FPGA”, Yi Wang College of Information Science and Engineering, Northeastern University, Shenyang, China.

2] Ben Ayed A., Ben Halima M., & Alimi A.M. “MapReduce-based text detection in big data natural scene videos”. Procedia Comput. Sci.,2016, 53, 216–223.

3] Bregler C. “Learning and recognizing human dynamics in video sequences”, Proceedings of IEEE CS Conference on Computer Vision and Pattern Recognition, 1997, 568-574.

4] Feng L., Liu Z.Y., Qi J. “Study of target detection methods in video tracking system [J]”, Microcomputer & Its Applications, 2014,12:34-36+39.

5] Geng Y.Y. “Research on target recognition and tracking system based on FPGA”, Changchun University of Technology, 2016.

6] Karmann K.P., Brandt A. “Moving object recognition using an adaptive background memory”, in Time-Varying Image Processing and Moving Object Recognition. V.Cappellini (ed.) Vol.2 Elsevier, Amsterdam, The Netherlands, 1990.

7] Kilger M. “A shadow handler in a video-based real-time traffic monitoring system”, Proceedings of the IEEE workshop on Application of computer Vision, 1992, pp. 1060-1066.

8] Liang Wang, Hu and Tan. “Recent developments in human motion analysis”, The journal of the Pattern Recognition, Elsevier publication, 2003, 585-601.

9] Li H.J. “Target tracking based on adaptive weighting of local features”, Network Security Technology & Application, 2019(04):41-43.

10] Li J., Wang T. and Zhang Y. “Face detection using SURF cascade”, In: IEEE international conference on computer vision workshops, Barcelona, 2011.

11] Madhuranath B. and Subrahmanya S.V. “Modified Adaboost method for efficient face detection”, In: 12th international conference on hybrid intelligent systems (HIS), Pune, 2012.

12] Mohan A.R. and Sudha N. “Fast face detection using boosted eigenfaces”, In: IEEE symposium on industrial electronics & applications, Kuala Lumpur, Malaysia, 2009.

13] Oro D., Fern'ndez C. and Segura C. “Accelerating boosting-based face detection on GPUs”, In: 41st international conference on parallel processing, Pittsburgh, 2012.

14] Ramya P. & Rajeswari R. “A Modified Frame Difference Method Using Correlation Coefficient for Background Subtraction”. Procedia Comput. Sci.,2016, 93, 478–485.

15] Rowley H. A., Rehg J. M. “Analyzing articulated motion using expectation-maximization”, Proceedings of the International Conference on Patten Recognition, 1997, 935-941.

16] Segundo P. and Silva B. “Real-time scale-invariant face detection on range images”, In: IEEE international conference on systems, man, and cybernetics, 2011.

17] Shen C., Paisitkriangkrai S., Jian Z. “Efficiently learning a detection cascade with sparse eigenvectors”, IEEE Trans Image Process, 2011.

18] Soundrapandiyan R. & Mouli P.V.S.S.R.C. “Adaptive Pedestrian Detection in Infrared Images Using Background Subtraction and Local Thresholding”. Procedia Comput. Sci.,2015, 58, 706–713.

19] Sung K, Poggio, T. “Example-based learning for view-based human face detection”, IEEE Trans Pattern Anal Mach Intell, 1998.

20] Viola P., Jones M.J. “Robust real-time face detection”, Int J Comp Vision, 2004.

21] Wang Y.H., Xu Z.Y., Ye D.M. “A target tracking algorithm based on local feature segmentation”, Laser Technology, 2019(43): 133-137.

22] Wei Liu, Dragomir Anguelov. “SSD: Single Shot MultiBox Detector”, University of Michigan, 2016.

23] Yang M.H., Kriegman D.J., Ahuja N. “Detecting faces in images: a survey”, IEEE Trans Pattern Anal Mach Intell, 2002.

24] Zhang C. and Zhang Z. “A survey of recent advances in face detection”, Microsoft Research Technical Report, 2010.

25] Zhang Z., Yi D., Lei Z. “Regularized transfer boosting for face detection across spectrum”, Signal Process Lett, 2012.