Visvesvaraya Technological University, Belagavi

Project Report On

Face Detection and Tracking in Video Sequence

4GH12CS002	Akshaya H J
4GH12CS014	Diwakar
4GH12CS019	Karthik G N
4GH12CS036	Pooja G V

Under the Guidance of

Mr. Ranganatha S *B.E.,M.Tech.,MISTE.*
Assistant Professor
Dept. of Computer Science & Engineering
Government Engineering College, Hassan

Department of Computer Science & Engineering
Government Engineering College, Hassan
June-2016

Government Engineering College, Hassan-573 201
Visvesvaraya Technological University, Belagavi

Certificate

This is to certify that the project work entitled **"Face Detection and Tracking in Video Sequence"** is a bonafide work carried out by **Karthik G N (4GH12CS019)** in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science Engineering of Visvesvaraya Technological University, Belagavi, during the year 2015-16. It is certified that all corrections / suggestions indicated during internal evaluation have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Guide | *Head of the Department*

Mr.Ranganatha S
B.E.,M.Tech.,MISTE.
Asst. Professor
Dept of CS & E, GEC, Hassan

Dr.K.C.Ravishankar
B.E.,M.Tech.,Ph.D.
Professor and Head
Dept of CS & E, GEC, Hassan

Principal

Dr. Karisiddappa *B.E.,M.Tech.,Ph.D.*
Principal
GEC, Hassan-573 201

Date :
Place : Hassan

Examiners : 1._____

2._____

Acknowledgement

At the outset I express my most sincere grateful thanks to my **Guide, Mr. Ranganatha S, Assistant Professor, Department of CS & E**, for his continuous support and advice not only during the course of the project but also during the period of my stay in GECH.

I express my gratitude to **Mr. M T ThirtheGowda**, Project Coordinator and Assistant Professor, Department of CS & E, for his encouragement and support throughout the work.

I express my gratitude to **Dr. K C Ravishankar**, Professor and Head, Department of CS & E, for his encouragement and support throughout the work.

I wish to express thanks to our beloved **Principal**, **Dr. Karisiddappa**, for encouragement throughout my studies.

Finally, I express my gratitude to all teaching and non-teaching staff of Department of CS & E, my fellow classmates and my parents for their timely support and suggestions.

Karthik G N (4GH12CS019)

Table of Contents

List of Tables

List of Figures

Abstract

In Video Processing, Face Detection and Tracking has a wide scope for research. The demand for biometric security system has risen due to a wide range of surveillance, access control and law enforcement applications. Although many existing algorithms are present for detection and tracking in the video sequences, the rapid development in video technologies is inducing more complexity in processing these videos. It is creating the way for many researchers to search for new methodologies as old algorithms are gradually becoming obsolete.

"Face detection and tracking in video sequence" project focuses on developing a modified algorithm from existing algorithms to increase the accuracy. The increase in tracking accuracy is achieved by fusing two different algorithms that work based on similar concepts and similar point of interest. The new fused Face detection and tracking algorithm provide more accuracy due to the fact that it combines two algorithms, it is a simple logic that if one algorithm fails to track the facial region, other algorithm keeps track of it and gradually the accuracy will be improved. Accuracy being improved doesn't mean that it will track all videos efficiently and accurately, it just shows that it tracks that video better than or equal to that of existing algorithm. Even fused algorithm has certain constraints, they have been discussed in further chapters.

Chapter 1

Introduction

An **Image** is a 2-Dimensional representation of a real world objects. The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or a painting. In this wider sense, images can also be rendered manually, such as by drawing, the art of painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph. A volatile image is one that exists only for a short period of time, a mental image exists in an individual's mind as something one remembers or imagines and a still image is a single static image.

Figure 1.1: Image Example

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing.

Video processing is a particular case of signal processing, which often employs video filters where the input and output signals are video files or video streams. A

video is nothing but just the very fast movement of pictures. The quality of the video depends on the number of frames or pictures per minute and the quality of each frame being used. Video processing involves noise reduction, detail enhancement, motion detection, frame rate conversion, aspect ratio conversion, color space conversion etc.

1.1 Electromagnetic Spectrum

The **electromagnetic spectrum** is the range of all possible frequencies of electromagnetic radiation. The "electromagnetic spectrum" of an object has a different meaning and is instead the characteristics distribution of electromagnetic radiation emitted or absorbed by that particular object.

The electromagnetic spectrum extends from the low frequencies used for modern radio communication to gamma radiation at the short-wavelength end, thereby covering wavelengths from thousands of kilometers down to a fraction of the size of an atom. The limit for long wavelengths is the size of the universe itself, while it is thought that the short wavelength limit is in the vicinity of the Planck length. Until the middle of last century it was believed by most physicists that this spectrum was infinite and continuous.

Electromagnetic radiation interacts with matter in different ways across the spectrum. These types of interaction are so different that, historically different names have been applied to different parts of the spectrum, as though these were different types of radiation. Although these "different kinds" of electromagnetic radiation from a quantitatively continuous spectrum of frequencies and wavelengths, the spectrum remains divided for practical reasons related to these qualitative interaction differences.

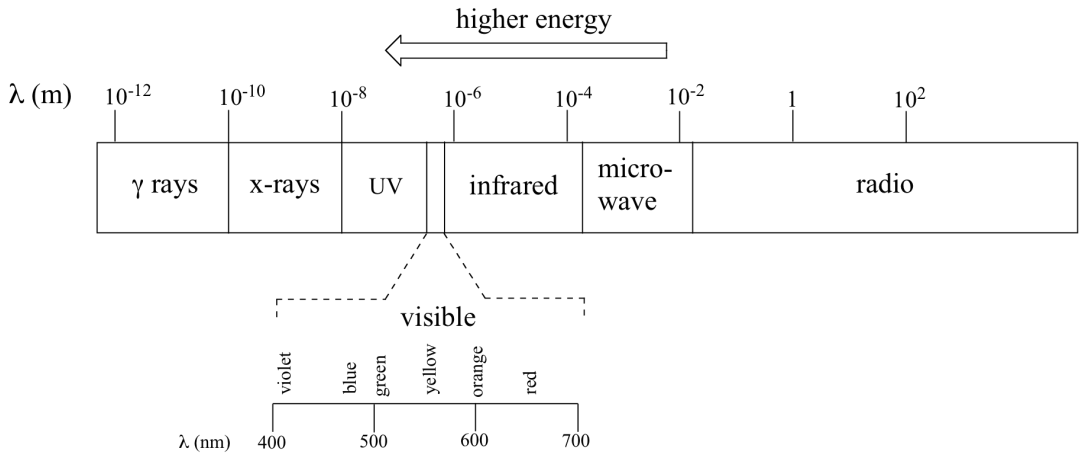

Figure 1.2: Electromagnetic Spectrum

Region of the spectrum	Main interactions with matter
Radio	Collective oscillation of charge carriers in bulk material.An example would be the oscillatory travels of the electrons in an antenna.
Microwave through far infrared	Plasma oscillation, molecular rotation.
Near infrared	Molecular vibration, plasma oscillation.
Visible	Molecular electron excitation, plasma oscillations.
Ultraviolet	Excitation of molecular and atomic valence electrons, including ejection of the electrons.
X-rays	Excitation and ejection of core atomic electrons, Compton scattering.
Gamma rays	Energetic ejection of core electrons in heavy elements, Compton scattering, excitation of atomic nuclei, including dissociation of nuclei.
High-energy gamma-rays	Creation of particle-antiparticle pairs. At very high energies a single photon can create a shower of high-energy particles and antiparticles.

Table 1.1: Spectrum table

In this electromagnetic spectrum, we are only able to see the visible spectrum. Visible spectrum mainly includes seven different colors that are commonly termed as (VIBGOYR). VIBGOYR stands for violet , indigo , blue , green , orange , yellow and red.

But that does not nullify the existence of other stuff in the spectrum. Our human eye can only see the visible portion, in which we saw all the objects. But a camera can see the other things that a naked eye is unable to see. For example: X-rays, gamma rays so on. Hence the analysis of all that stuff too is done in digital image processing.

This discussion leads to another question which is; why do we need to analyze all the other regions of the spectrum in Electromagnetic spectrum?

The answer to this question lies in the fact, because that other stuff such as X-Ray has been widely used in the field of medical. The analysis of Gamma ray is

necessary because it is used widely in nuclear medicine and astronomical observation. Same goes with the rest of the things in EM spectrum.

1.2 Applications of DIP

Some of the major fields in which digital image processing is widely used are mentioned as follows:

1. **Image sharpening and restoration** refers to process images that have been captured from the modern camera to make them a better image or to manipulate those images in way to achieve desired result. It refers to do what Photoshop usually does. This includes Zooming, blurring, sharpening, gray scale to color conversion, detecting edges, image retrieval and image recognition.

2. **Medical field** includes Gamma ray imaging, PET (Positron emission tomography) scan, X-ray imaging, CT (computerized axial tomography) scan, UV (Ultraviolet) imaging.

3. **Transmission and encoding** mainly deals with transmission of the images from one location to another via network. Today we are able to see live video feed or live CCTV footage from one continent to another with just a delay of seconds. It means that a lot of work has been done in this field too. This field does not only focus on transmission, but also on encoding. Many different formats have been developed for high or low bandwidth to encode photos and then stream it over the internet so on.

4. **Machine/Robot vision** deals with many challenges that a robot face today, one of the biggest challenge still is to increase the vision of the robot. Make robot able to see things, identify them, identify the hurdles and so on. Much work has been contributed by this field and a complete other field of computer vision has been introduced to work on it.

5. **Color processing** includes processing of colored images and different color spaces that are used. For example RGB color model, YCbCr, HSV. It also involves studying transmission, storage and encoding of these color images.

6. **Pattern recognition** involves study from image processing and from various other fields that includes machine learning (a branch of artificial intelligence). In pattern recognition, image processing is used for identifying the objects in an images and then machine learning is used to train the system for the change in pattern. Pattern recognition is used in computer aided diagnosis, recognition of handwriting, recognition of images and so on.

7. **Video processing** A video is nothing but just the very fast movement of pictures. The quality of the video depends on the number of frames/pictures per minute and the quality of each frame being used. Video processing involves noise reduction, detail enhancement, motion detection, frame rate conversion, aspect ratio conversion, color space conversion and so on.

8. **Microscopic Imaging.**

1.3 Video Processing

As discussed above video processing is a case of signal processing, which often employs video filters where the input and output signals are video files or video streams. Video processors are often combined with video scalars to create a video processor that improves the apparent definition of video signals. They perform the following tasks:

- Deinterlacing
- Aspect ratio control
- Digital zoom and pan
- Brightness/contrast/hue/saturation/sharpness/gamma adjustments
- Frame rate conversion and inverse-telecine
- Color space conversion
- Mosquito noise reduction
- Block noise reduction
- Detail enhancement
- Edge enhancement
- Motion compensation

Video processing techniques can be employed in various areas of computer vision such as Face detection and tracking, line follower robots, object detection, Feature extraction and so on. Video processing techniques for our project Face Detection and Tracking has various algorithms such as Viola-Jones Face Detection algorithm, KLT Algorithm, Mean shift Algorithm, CAMSHIFT Algorithm and various other customized algorithms.

Even though we have many algorithms and techniques for Face Detection and Tracking, there are various challenges that have to be taken care beforehand developing the new techniques, few of the challenges faced are:

- **Illumination**
  It is the appearance of the face changes along with the variation in illumination. Skin reflectance properties and the internal camera control are the major causes for variations. Under moderate illumination variation, 2D methods do well but when both illumination and pose variations occur, performances drop noticeably.

- **Pose variation**
  Pose variation is a serious problem for the identification of the input image, if the database includes only the frontal view then the authentication process is severely affected. Most of the face recognition systems assume the availability of frontal faces of similar sizes. Due to the varied nature of face appearance and environment conditions, this assumption may not hold in reality.

- **Skin texture**
  Problems are also imposed by facial expressions and skin tone in face identification. Most of the existing algorithms till date work robust, except for extreme expressions such as scream and various skin tone.

- **Ageing**
  Face detection and tracking accuracy also depends on the age variation of the humans. As the person becomes older, the features of the face also changes which will affect the accuracy of recognition. The aging modeling technique is quite robust against illumination and pose variation problems.

- **Surrounding environment**
  A face could occur in a complex background and in many different positions. Standard face images based recognition systems are likely to mistake some areas of the background as a face. In order to rectify the problem, the face region has to be localized and extracted.

- **Occlusion**
  Face detection is severely affected when the face in the input image lacks some parts possibly due to scarf, glasses, beard and mustache.

Video processing is widely used in the areas like Entertainment, Smart cards, Information security, Law enforcement and Surveillance and so on.

1.4 MATLAB Toolbox

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, FORTRAN and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

The MATLAB application is built around the MATLAB scripting language. Common usage of the MATLAB application involves using the Command Window as an interactive mathematical shell or executing text files containing MATLAB code. Like other scripting languages MATLAB also provides facilities for variables, vectors, matrices, structures, functions, classes and objects. MATLAB supports developing applications with graphical user interface (GUI) features. MATLAB includes GUIDE (GUI development environment) for graphically designing GUIs. It also has tightly integrated graph-plotting features.

MATLAB is a proprietary product of MathWorks, so users are subjected to vendor lock-in. Although MATLAB Builder products can deploy MATLAB functions as library files which can be used with .NET or Java application building environment, future development will still be tied to the MATLAB language. Each toolbox is purchased separately. If an evaluation license is requested, the MathWorks sales department requires detailed information about the project for which MATLAB is to be evaluated. If granted (which it often is), the evaluation license is valid for two to four weeks. A student version of MATLAB is available as is a home-use license for MATLAB, Simulink, and a subset of Mathwork's Toolboxes at substantially reduced prices.

The various tools are provided by MATLAB for Video Processing and Computer vision. Computer Vision System Toolbox provides algorithms, functions, and apps for designing and simulating computer vision and video processing systems. We can perform feature detection, extraction, and matching; object detection and tracking; motion estimation; video processing and so on. For 3-D computer vision, the system toolbox supports camera calibration, stereo vision, 3-D reconstruction, and 3-D point cloud processing. With machine learning based frameworks, we can train object detection, object recognition, and image retrieval systems. Algorithms are available as

MATLAB functions, System objects and Simulink blocks.

Key features provided by MATLAB Computer Vision toolbox are:

- Object detection and tracking, including the Viola-Jones, Kanade-Lucas-Tomasi (KLT), and Kalman filtering methods.
- Training of object detection, object recognition and image retrieval systems including cascade object detection and bag-of-features methods.
- Camera calibration for single and stereo cameras, including automatic checkerboard detection and an app for workflow automation.
- Stereo vision, including rectification, disparity calculation and 3-D reconstruction.
- 3-D point cloud processing, including I/O, visualization, registration, denoising and geometric shape fitting.
- Feature detection, extraction and matching.
- Support for C-code generation and fixed-point arithmetic with code generation products.

Chapter 2

Literature Survey

According to the survey presented by **Ranganatha S, Dr.Y P Gowramma** [1] in recent days the demand for biometric security system has risen due to a wide range of surveillance, access control and law enforcement applications. Among the various biometric security systems based on finger print, iris, voice or speech and signature, face recognition seems to be easily accessible, non-intrusive, and most universal system. Face recognition techniques can be used efficiently for mass scanning which is quite difficult in case of other biometrics and also increases user friendliness in human computer interaction. Several interesting research attempts are made in the field of face recognition. There are three main divisions of face recognitions based on the face data acquisition type: methods that deal with intensity images, those that operate on video sequences and those that are based on other sensor inputs. As per this survey the challenges that has to be faced here are illumination, pose, expressions, ageing, occlusion, background conditions and so on. Improvements in computer hardware, an algorithm capable of tracking and recognizing human faces in different background video data with accurate rate, face tracking with high resolution improves the accuracy rate for face recognition, parallel algorithms which speeds up the computation and so on can be adopted to overcome the challenges that are going to be encountered while carrying out our experiment in this area.

R. Jagathishwaran, K.S. Ravichandran and Premaladha Jayaraman [2] made a survey on Face Detection and Tracking. Face detection and Tracking are important research areas in the field of computer vision and image processing. Face detection is a computer technology that helps to determine the locations and size of the human faces. Face detection techniques are used in cameras for auto focus. Face detection and tracking are the two processes done by using various approaches. It is applied mainly in surveillance. The main purpose of these processes are detect and track the face even in poor viewing conditions in surveillance application. In this

survey various techniques are used for human detection and tracking such as adaptive color based particle filter, fuzzy based particle filter algorithm and so on are discussed. Comparisons between the various approaches are illustrated, performance measures in terms of number of particles used, root mean square error values etc have been reported. Drawbacks for the techniques like tracker facing the problems while detection and tracking has been explained, reasons why fuzzy based particle filter is best among all the approaches have been produced. This survey presents the discussion about various face detection and tracking approaches, performance measures and drawbacks. Tracking faces is the main process used in different fields like surveillance application, computer vision and image processing. Particle filter algorithm is frequently used for tracking process even though it performs well, it faces some difficulties like, if both objects are very close to each other with similar color then the tracker fails. So, fuzzy based particle filter algorithm is suggested for solving this type of problem. Fuzzy system is constructed by hierarchical structures. It provides good performance than particle filters and requires more processing time.

WU Lushen, WU Peimin, MENG Fanwen [3] proposed a fast face detection algorithm which integrates differential algorithm and mosaic rules for videos. The comprehensive algorithm with temporal difference and background difference was used for acquiring and tracking the moving objects, which can accurately locate the target region. In the region, the gray rules based on mosaic image were used for detecting face, in which the intergral image was adopted to compute the sum gray value. This algorithm uses a new method based on the frequency histogram to combine the detection area, and finally the face position was obtained. The experiment shows that the average detection time is 30 ms/frame and the accuracy rate is high, which is suitable for real-time face detection for different kinds of video images. The drawback of this algorithm is that even though it is less complex the number of processing stages are more, computing and analysis of frequency histogram and the high hardware requirements to process them.

One of the important method in the field of face detection is the **Viola Jones method** [6].This system was trained to detect only front facing images. So the system works well for front facing images and it is the fastest algorithm in its category. No color information was used in this algorithm. The detection rate was mentioned to be equal to that of the Rowley-Baluja-Kanade detector [8]. The main drawback of this method was that it was trained to detect only frontal faces from images. This algorithm can be trained to detect faces having different orientations. For detecting faces in other orientations additional training is required. The paper by Jones et

al. [9] is an extension to the paper published by Viola et al. [6]. This method was trained to detect rotated faces and profile faces. A decision tree approach was used to determine the viewpoint from the image. Twelve rotation classes each of angle 60 (12x60=360)was used for the detector.

Dorin Comaniciu and Peter Meer [7] proposed a non-parametric feature-space analysis technique for analysis of a complex multimodal feature space and to delineate arbitrarily shaped clusters in it. Mean shift is a technique for locating the maxima of a density function, a so-called mode-seeking algorithm [11]. Application domains include cluster analysis in computer vision and image processing [6]. The mean shift algorithm can be used for visual tracking. The simplest such algorithm would create a confidence map in the new image based on the color histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position. The confidence map is a probability density function on the new image, assigning each pixel of the new image a probability, which is the probability of the pixel color occurring in the object in the previous image. A few algorithms, such as ensemble tracking, CAMSHIFT [10] expand on this idea. The drawbacks of this technique are the selection of a window size is not trivial, inappropriate window size can cause modes to be merged, or generate additional "shallow" modes, often requires using adaptive window size.

The problem with Mean shift technique is that it does not give appropriate window size while tracking, which is not good. So the Mean shift algorithm has been modified by **G.R Bradski** [10] came up with an the idea of CAMSHIFT (Continuous Adaptive Mean shift) technique which solved the drawback of Mean shift and the window is varied according location of the object being tracked and also this technique emphasized more on the facial region, CAMSHIFT is considered as one of the robust technique till date for tracking faces in the video sequences. CAMSHIFT technique is often used along with Viola Jones technique [6][8] for detection of the faces in the video.

Han-Pang Huang and Chun-Ting Lin [5] developed a system for multiple objects tracking and multi-view faces detection and recognition. A novel method (Multi-CAMSHIFT), which is based on the characteristics of color and shape probability distribution, to solve the tracking problems for multiple objects. The tracker is used to get the candidate regions by outlining the interested probability distribution. The system performance is further improved by using multi-resolution framework.

The principal component analysis (PCA) and support vector machine (SVM) are integrated to form the multi-view faces detection and recognition module for classifying different face poses and identities. Beside color information, the gray background image is used to locate the human head in the region of tracking pedestrian based on probability distribution rule. The rule can also be used for skin color face tracking to remove background region (non-face region). Since this proposed Multi-CAMSHIFT (MCAMSHIFT) is computationally efficient, it can work in complex background and track in real-time. The slowly changing lighting condition is effectively resolved using probability model update. From experiments, the proposed MCAMSHIFT was successfully applied to multi-view faces tracking and recognition. It can also be applied to surveillance system, pedestrian tracking and face guard systems. As the system involves multi-view faces to be detected and recognized the complexity of the algorithm increases and hence the time complexity will be very large for larger length videos.

In computer vision, the **Kanade Lucas Tomasi (KLT) feature tracker** is an approach to feature extraction. It is proposed mainly for the purpose of dealing with the problem that traditional image registration techniques are generally costly. KLT makes use of spatial intensity information to direct the search for the position that yields the best match. It is faster than traditional techniques for examining far fewer potential matches between the images. The KLT feature tracker is based on two papers, in the first paper, Lucas and Kanade [14] developed the idea of a local search using gradients weighted by an approximation to the second derivative of the image and in the second paper Tomasi and Kanade [13] used the same basic method for finding the registration due to the translation but improved the technique by tracking features that are suitable for the tracking algorithm. The proposed features would be selected if both the eigenvalues of the gradient matrix were larger than some threshold. In a third paper, Shi and Tomasi [12] proposed an additional stage of verifying that features were tracked correctly. Most important point to observe in this technique is that for face detection and tracking, this technique is one of the best and optimized method till date.

From the literature analysis we can come to the following conclusions:

- There are various fast algorithms for detection and tracking.
- There are various algorithms of higher bound time complexity.
- There are various predefined and robust algorithms existing for each of detection, extraction and tracking.

- Lower bound complexity algorithms require more number of processing steps.
- There are various challenges that has to be taken care while designing new algorithms.
- There are many modified algorithms to the original algorithms to increase its efficiency such as Mean shift, CAMSHIFT, Multi-CAMSHIFT.

By inferring from the above conclusion; we draw the idea for our project which takes care of few challenges, lesser complexity, lesser number of processing steps, new algorithm that would comprise of many algorithms to increase the efficiency and reduce the time complexity for the computation and so on.

Chapter 3

Requirements & Constraints

This chapter deals with the functional and non-functional requirements of the system. In general functional requirements define what the system is suppose to do, whereas non-functional requirements defines how a system is supposed to be.

3.1 Functional Requirements

This defines the function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing or other specific functionality.

3.1.1 Input

Video extensions are of many types such as .wmv (windows media video), .avi (audio video interleave), .3gp (Third Generation Partnership Project), .flv (Flash video), .mpeg (Moving Picture Experts Group), .mp4 (Moving Picture Experts Group Phase-4) and so on, among these video file types user can use any one of them as input. User must provide high resolution videos to have a better chances of face detection and tracking.

3.1.2 Method

Firstly, a decent quality video file is taken as input and using Viola-Jones algorithm face region is detected. Next, the the frame undergoes masking, then uses Sobel's edge detection algorithm to detect all the possible edges inside the ROI and then uses Gaussian filter to remove noise and then using Harris corner measure corner points are detected. Once corner points are detected using that the face region is tracked throughout the further frames.

3.1.3 Output

The output is dependent on the option that user selects. If he goes for detection then only face in the first frame is detected, if he goes for detection and tracking, then the face is detected and the tracking of the video sequence is displayed to the user and if he selects option for edge detection then user gets to view all the processing stages that has taken place to detect the edges and corner points in the given video sequence.

3.2 Non-Functional Requirements

Non-functional requirement is a requirement that specifies the criteria that can be used to judge the operation of a system, rather than specific behaviors.

3.2.1 Software Requirements

Platform : Cross-platform (64-bit system).
Environment : Matlab R2015a or above (64-bit version).
Libraries : Matlab Runtime Libraries (To make .exe work windows only).

3.2.2 Hardware Requirements

Processor : Intel Core 2 duo or higher.
RAM : 512 MB minimum/1 GB or higher recommended.
Hard Disk : 20 GB or higher recommended.

3.3 Constraints on Requirements

All the requirements as mentioned above are subjected to some constraints in our project.

3.3.1 Constraints on Functional Requirements

1. The face must be present in the first frame of the input video sequence.
2. The video must be recorded only by fixing the camera in one particular location or fixing the person location and varying the camera.
3. Variation in camera position must be negligible, failure in this leads to increase in complexity while detecting and tracking the faces in video sequence.

4. To get more accuracy in result, the input video must be of high quality.

5. The input video format must be one among the standard formats used world-wide, change in format may lead to false results.

6. As the project fuses various algorithms to increase its efficiency, output binds with the few of the limitations of each algorithms even after overcoming most of their drawbacks.

7. The resulting system must have only one face detected in the first frame, in case there are multiple faces detected then the Sobel's algorithm detect edges but computation of centroid fails; leading to failure in tracking of face(s) in further frames of the video sequence.

3.3.2 Constraints on Non-Functional Requirements

1. The output is directly proportional to the hardware on which this project works on.

2. Since this technique needs more computational efficiency, the target system must have above average CPU and GPU for optimal performance.

3. MATLAB is a cross-platform tool, so any operating system with MATLAB, symulink and computer vision toolbox present in it can use the technique described in the project for detection and tracking.

4. Higher the RAM, faster will be the processing and HDD must be variable to store more video sequences.

Chapter 4

System Design

4.1 Architectural Design

The main objective of the FDT (Face detection and tracking) in video sequence is to detect the face present in the video sequences and track it throughout the further frames in that video sequence. The FDT system collects input from the GUI (Graphical User Interface), further steps of processing is based on the user's choice selection from the GUI. Once the user selects the option the function associated with that option is executed.

In FDT system we have used MVC (Model View Controller) architectural pattern. Model-view-controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. The MVC consists of following components:

1. The **model** directly manages the data, logic and rules of the application.
2. A **view** can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar-chart, pie-chart for management and a tabular view for accountants.
3. The **controller** accepts input and converts it to commands for the model or view.

In the architecture shown in Figure 4.1, user interacts with the computer system which takes input from user and sends it to controller component and also displays output sent by the view component.

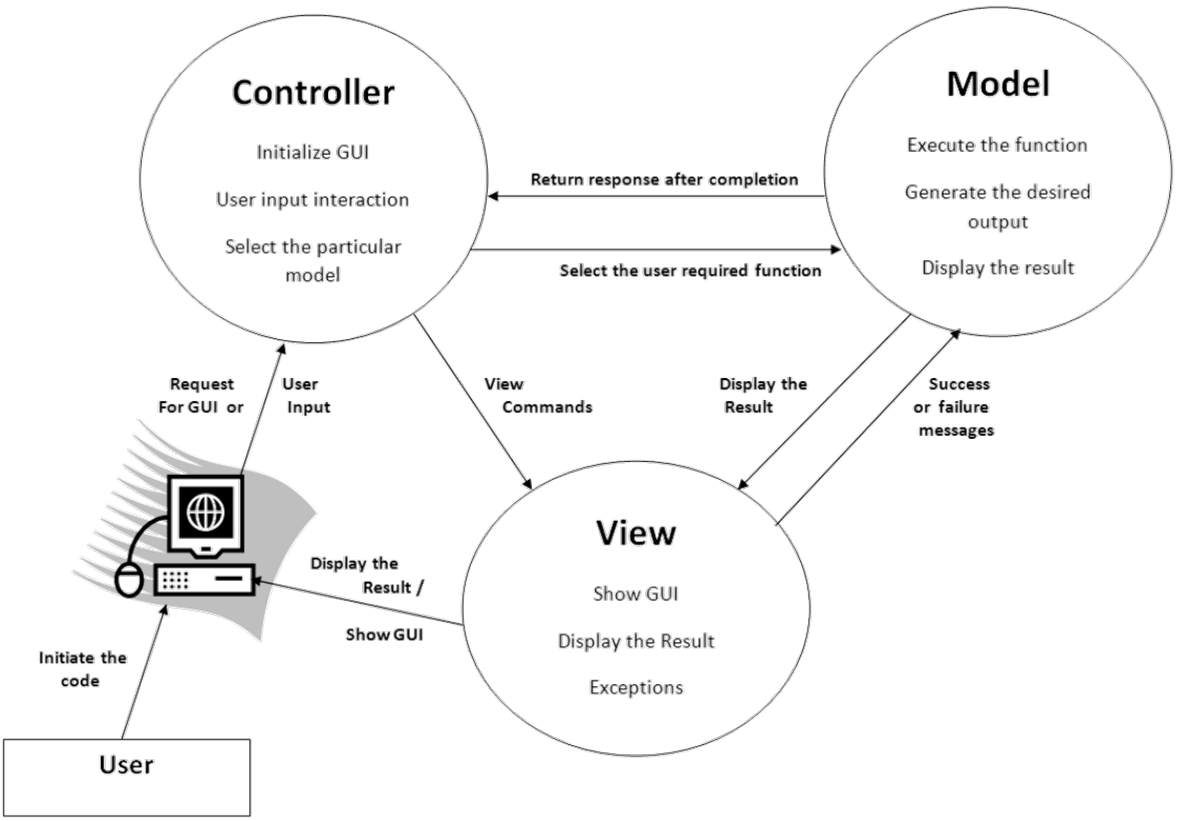

Figure 4.1: Architectural Design

The working of the architecture is as follows:

1. In the beginning stage the user initiates the code present in the computer system, the request is forwarded to controller and it initiates view command, the view component displays GUI on the computer system.

2. Then the user can choose the desired option present in the GUI.

3. The user input generated from option selection makes the component to call the user required function.

4. Model components executes the function requested and sends the response back to the controller after completion, also it forwards the result(s) to the view component, which in turn displays the result on the computer system.

5. If any exception occurs or after successful result the view component sends the appropriate message to the model component.

6. This process continues until user terminates the GUI.

4.2 Sequence Diagram

A sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It shows object interaction arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows a parallel vertical lines(lifelines), different processes or objects that live simultaneously and horizontal arrows as the messages exchanged between them in the order in which they occur.

Face detection scenario

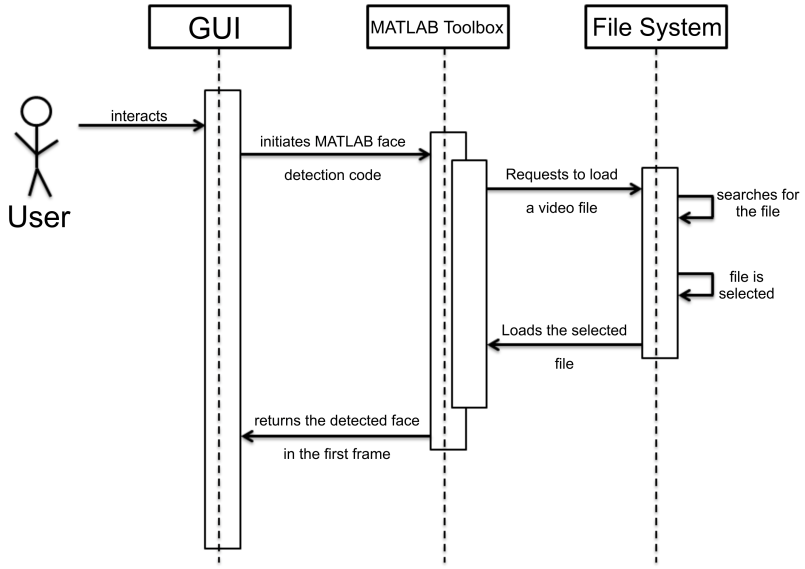

Figure 4.2: Face detection Scenario

In the face detection scenario as shown above, the interaction between user, GUI, MATLAB Toolbox, File system can be seen. All these objects interact simultaneously to accomplish the task of detecting the facial region in the first frame of the video sequence.

Edge & Corner detection scenario

In the Edge & Corner detection scenario as shown in Figure 4.3, the interaction between user, GUI, MATLAB Toolbox which comprise of face detection function and edge detection function can be seen, all these objects interact simultaneously to accomplish the task of detecting the facial region in the first frame of the video sequence

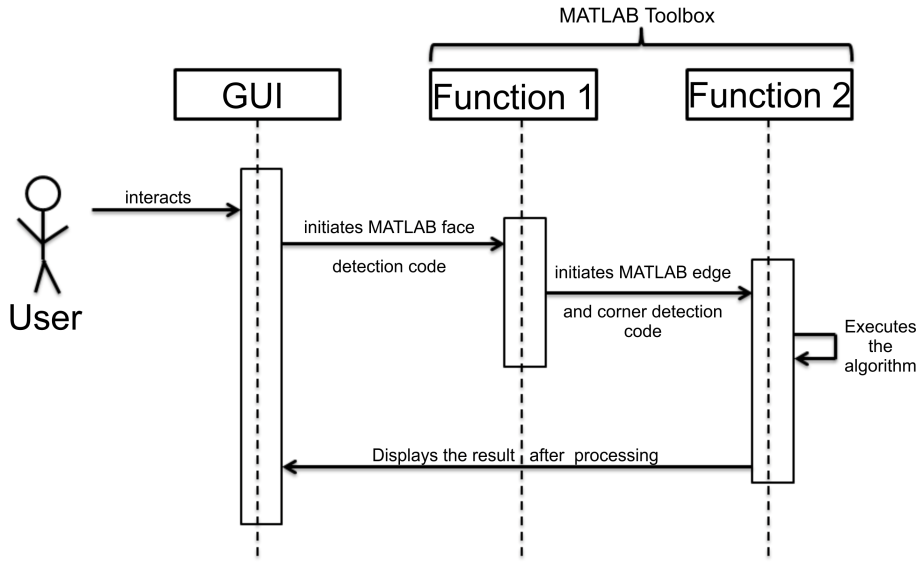

Figure 4.3: Edge and corner detection Scenario

and after that the corner points array is obtained using which we compute the centroid of the facial region.

Face tracking scenario

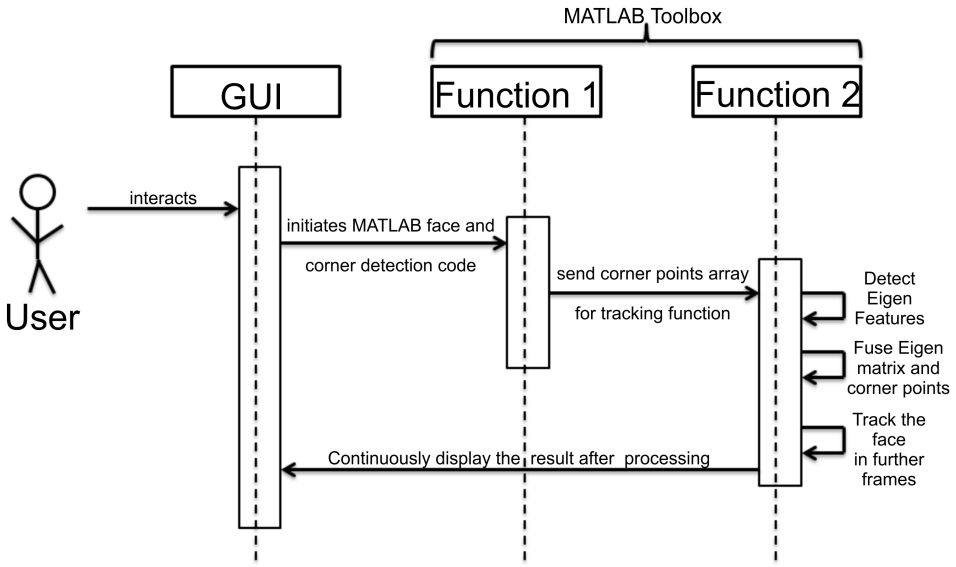

Figure 4.4: Face tracking Scenario

In the face tracking scenario as shown above, the interaction between user, GUI, MATLAB Toolbox which comprise of face and edge detection function and face tracking function can be seen, all these objects interact simultaneously to accomplish the task. The face after successful detection is subjected to corner points detection in it, the detected corner points array is fused with the Eigen points matrix and all the points are then tracked using KLT point tracker algorithm.

The detailed module wise explanation is done in the next section, it gives more clearer information regarding each module and function that has been used in the FDT system.

4.3 Modules Design

The system consists of 3 modules: Face detection module, Edge & Corner detection module and Face tracking module. Each module can be interacted by user using GUI.

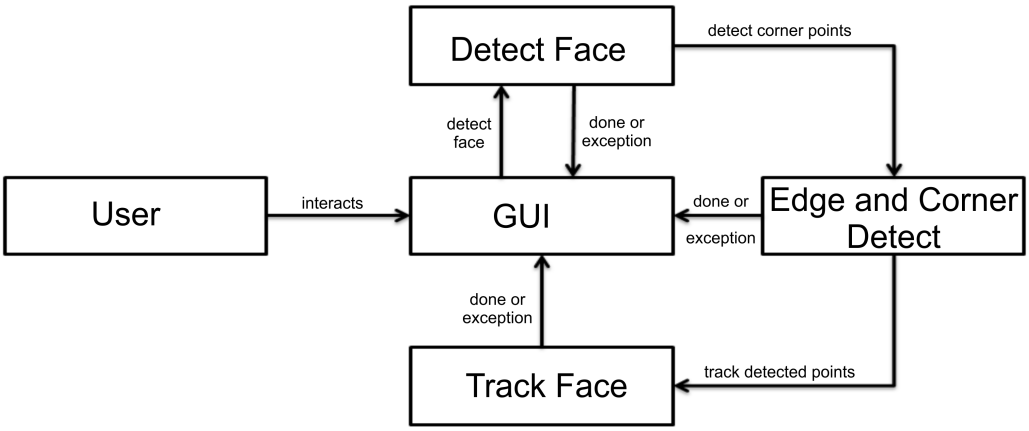

Figure 4.5: FDT system modules design

4.3.1 Data Flow Diagram

A Data Flow Diagram(DFD) is a graphical representation of the flow of data through an information system. Data flow diagram can also be used for the visualization of data processing or structured design. Data flow diagrams can be used in both Analysis and Design phase of the SDLC.

Face detection module

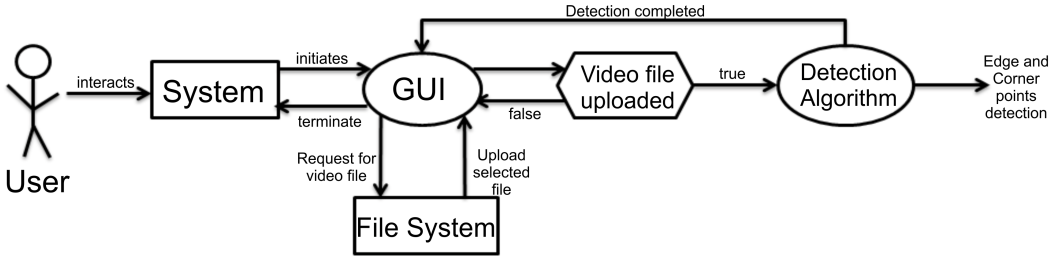

Figure 4.6: Face detection DFD

This module mainly deals with the detection of facial region in the video sequence, which user has given as input to the FDT system. In this module face is detected using Viola Jones detector. As shown in Figure 4.6, the user interacts with system and initiates GUI. Then using GUI user loads the video file from file system on to the FDT system. Then that video file is sent to Face detection algorithm for further computation and detection of face. Once the face is detected; the control returns to GUI for further user inputs or the output is sent to Edge & Corner detection module depending on the option selected by user in GUI.

Edge & Corner detection module

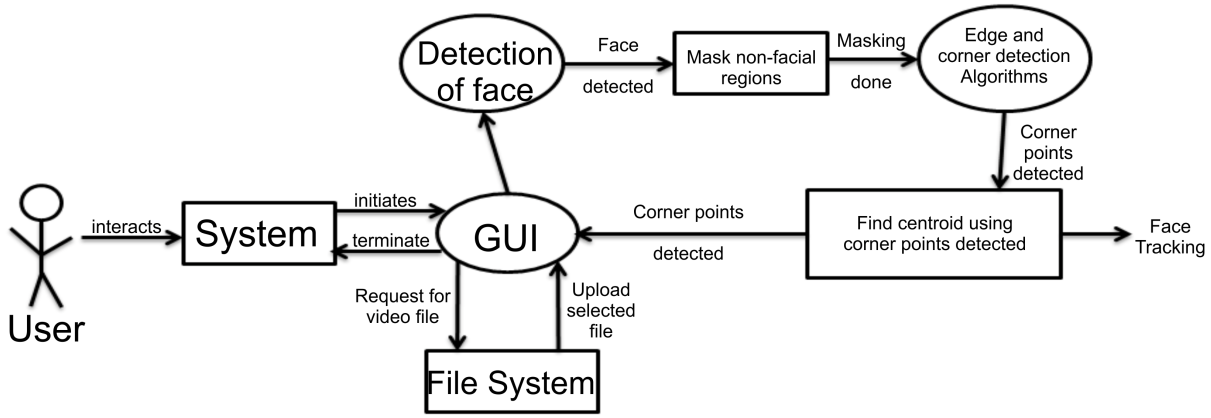

Figure 4.7: Edge and Corner Detection DFD

This module deals with detecting of the edges and corner points in the required region of a video frame. As shown in Figure 4.7, this module follows the face detection module. After detecting face region from the previous module, the non-facial

region is masked, using Sobel's Edge detection algorithm find the computer distinguishable edges and using these edges, corner points are detected. Using these points the centroid of the facial region is calculated. Similar to Face detection module if the user wants to detect only edges and corner points on the face region then it displays them and returns the control to GUI, else control is proceeded to the tracking module.

Face tracking module

This module deals with tracking of the face region that is being detected in the first frame. After detection of the face region, the corner points are detected which gives corner points array and also the same detected region is used to find Eigen point matrix in it. Then both these corner points and Eigen points are fused to form a

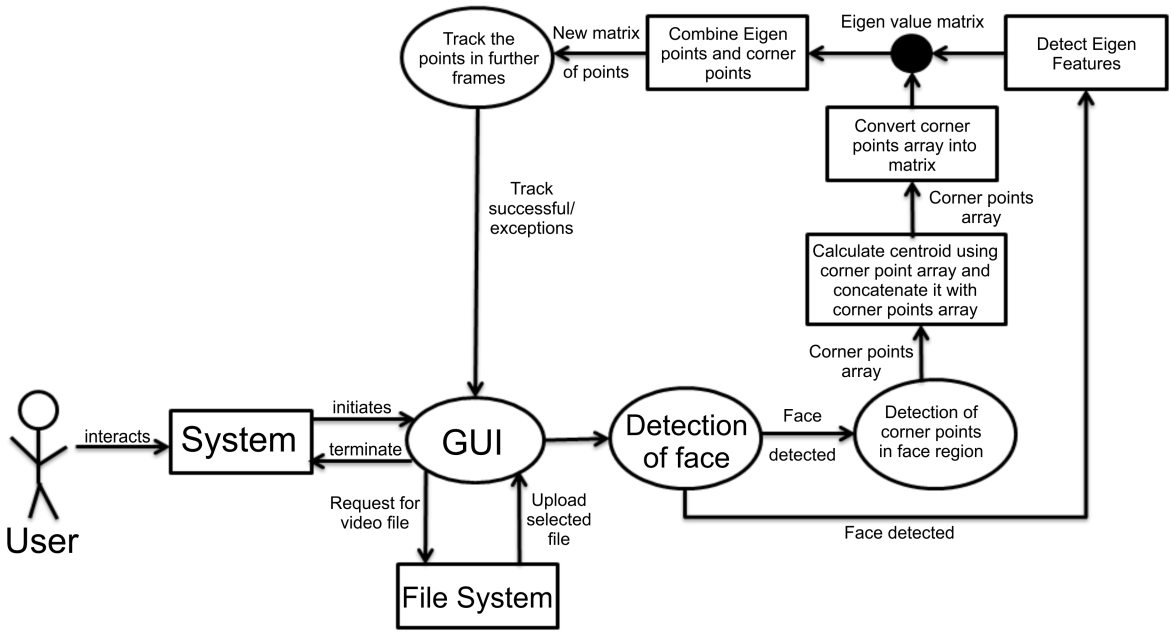

Figure 4.8: Tracking face DFD

single matrix of points. These points are then used along with point tracker of KLT algorithm.

The theoretical aspect of the outcome states that tracking accuracy will be atleast few frames greater than or equal to that of KLT algorithm. It is a general prediction because we are using 4 methods that are fused together for tracking the facial regions i.e., Eigen Matrix, Edges, Corner points and Centroid point all of them work together to track the facial region in the further frames. The practical feasibility of the fusion concept as discussed above theoretically is the main goal of this project work on the developed FDT system.

Chapter 5

Implementation

In computer science, implementation is the practical application of a methodology or algorithm to full a desired purpose. In software architectural design, the top-level structure and organization of the system is described and various components are identified (how the system is decomposed and organized into components). It must describe the interfaces between these components.

The FDT system design as discussed in previous chapter includes various modules which interact with each other simultaneously to achieve the expected result i.e., in our case, tracking of the facial region in the frames of the video sequence which is given as an input to the system by the user.

5.1 Methodology

5.1.1 Face Detection Function

- Get the video file from the user using the GUI.
- Create a cascade detector object by using vision toolbox as

  vision.CascadeObjectDetector();

- Read the file and store its handle in a variable.
- Extract first frame from the video sequence and detect facial region to put bounding box using Viola-Jones face detector created above

  step(videoFileReader);
  step(faceDetector, videoFrame);

- Draw bounding box in the detected region.

5.1.2 Edge & Corner Detection Function

- Detect face as described above.
- Mask out non-facial region using the command

createMask(e,x);

- Apply Gaussian low-pass filter to reduce the noise in the frame.
- With Sobel's Edge detector algorithm detect computer distinguishable edges in the unmasked region of the first frame.
- After obtaining edges, apply Harris corner measure to find corner points and find the centroid of the facial region.
- Plot and display the corner points detected as well as centroid of the facial region of the first frame.

5.1.3 Face Tracking Function

- Concatenate the centroid point with the corner points array and form a new corner points array.
- Convert the modified coordinate points array into a matrix.
- After converting points to matrix concatenate it with the Eigen points matrix and obtain a new set of points.
- These new set of points matrix is tracked throughout the next frames in the video sequence.
- These new points are tracked using point tracker of KLT point tracker algorithm.

Overall after successful satisfaction of above methodology, the theoretical assumption done in the system design chapter is practically achieved. The FDT system designed by fusing KLT algorithm along with Sobel's Edge detection and Harris Corner measure algorithms gives more accuracy i.e., face is tracked in few more number of frames or at least in equal number of frames when the fused algorithm is used over KLT algorithm due the fact that fused FDT system in modified version of KLT algorithm which not only works based on Eigen values, but also on Corner points as well as on the calculated centroid point.

Next chapter include results of various tests done on the number of different video sequences to verify the credibility of the FDT system developed, the videos have been captured under different conditions such as varying illumination, pose variation, video quality and many other constraints to prove the effectiveness of the fused FDT system.

5.2 Pseudo code

5.2.1 Face Detection

```
function detect()
    x ← Read the video file
    Create vision.CascadeObjectDetector()
    y ← read video file(x)
    z ← step(y)
    bbox ← Find bounding box points
    a ← Draw bounding using bbox points
    Display a
end function()
```

5.2.2 Edge & Corner Detection

```
function edge()
    Detect Facial region
    Detect face as described above
    b ← createMask() from the bounding box
    c ← Augment the mask to three channels
    ROI ← y
    ROI(c==0) ← 0
    Display extracted portion
    ROI ← rgb2gray(ROI)
    Ix ← filter2(fx,ROI)
    Iy ← filter2(fy,ROI)
    Apply Gaussian filter on computed values
    height ← size(ROI,1)
    width ← size(ROI,2)
    result ← zeros(height,width)
    R ← zeros(height,width)
    set Rmax ← 0
```

```
        for i ← 1:height
            for j ← 1:width
            Compute Rmax
            end
        end
        cnt ← 0
        i ← 2:height-1
            j ← 2:width-1
                if R(i,j) is corner point
                    result(i,j) ← 1
                end
            end
        end
        [posc, posr] ← find(result == 1)
        Display ROI
        minr ← posr(1)
        minc ← posc(1)
        maxr ← 0
        maxc ← 0
        for i ← 1:size(posc)
            if posc(i) ≥ maxc
                maxc ← posc(i)
            end
            if posc(i) < minc
                minc ← posc(i)
            end
            if posr(i) ≥ maxr
                maxr ← posr(i)
            end
            if posr(i) < minr
                minr ← posr(i)
            end
        end
        tr ← (maxr-minr)/2
        tc ← (maxc-minc)/2
        Display y and plot tr and tc
        Display Detected face and all corner points and centroid point
    end function()
```

5.2.3 Track Face

```
function track()
    Detect Facial region
    Generate corner point array as described above
    points ← detectMinEigenFeatures()
    Concatenate points matrix and valrc array
    imshow(y) and plot(points)
    Create vision.PointTracker('MaxBidirectionalError', 2)
    points ← points.Location
    initialize(pointTracker, points, y)
    initialize video player to display the results
    while !isDone(x)
        y ← step(x) %Read next frame
        Track the points by [points, isFound] ← step(pointTracker,
                                                videoFrame)
        if size(visiblePoints, 1) ≥ 2 % need at least 2 points
            Estimate the geometric transformation between the old points
            and the new points
            bboxPoints ← transformPointsForward(xform, bboxPoints)
            Insert a bounding box around the object being tracked
            Reset the points
        end
    step(videoPlayer, y)
    end
end function()
```

Chapter 6

Test report

6.1 Comparison Testing

Comparison testing is one of the testing approaches in Software Testing done between two or more products. Comparison of product strengths and weaknesses with previous versions or other similar products.

In this project comparison test is carried out between KLT algorithm and newly developed FDT system which is also a modified version of KLT system. To compare between these two approaches of tracking, let us evaluate both systems frame by frame for various video sequences. The presence of bounding box around the the face is counted frame by frame in each video sequence by introducing a new count variable into the tracking function of both systems, calculating the number of frames that contain bounding box around the face gives more accuracy.
The results are tabulated in the below table after carrying out comparison testing:

Video No.	No. of Frames	KLT	Fused FDT	Fused without centroid
1	412	412	412	412
2	170	170	170	170
3	103	71	71	71
4	30	30	30	30
5	157	157	157	157
6	64	64	64	64
7	246	246	246	246
8	88	88	88	88
9	225	225	225	225
10	180	57	90	90
11	112	112	112	112
12	100	100	100	100

Video No.	No. of Frames	KLT	Fused FDT	Fused without centroid
13	185	185	185	185
14	153	153	153	153
15	255	255	255	255
16	135	135	135	135
17	316	316	316	316
18	82	82	82	82
19	62	3	62	62
20	192	192	192	192
21	41	41	41	41
22	150	150	150	150
23	248	242	248	242
25	257	257	257	257
26	95	95	95	95
27	110	110	110	110

Table 6.1: Comparison testing

From the Table 6.1, on closer observation we can notice that the Fused FDT system tracks few more frames than KLT algorithm alone tracks. The result may not be 100% success in tracking with the new fused FDT algorithm, but it does track more number of frames or atleast equal number of frames as that of KLT algorithm. The theoretical assumption that was made in the beginning of the project stating that the new algorithm that is going to be developed will have more accuracy than that of KLT algorithm, this assumption has been practically achieved and the evidence to it can be seen in the above tabulation.

The major accuracy is achieved in video number 19, KLT algorithm just tracks for 3 frames whereas the new FDT system tacks throughout all frames (62 frames) in that video sequence.

6.2 Manual Testing

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user and use most of all features of the application to ensure correct behavior. To ensure completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases. Manual testing can be done through:

- In **white-box testing** the tester is concerned with the execution of the statements through the source code.
- In **black-box testing** the software is run to check for the defects and is less concerned with how the processing of the input is done. Black-box testers do not have access to the source code.
- **Grey-box testing** is concerned with running the software while having an understanding of the source code and algorithms.

6.2.1 Black-box testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

To perform black-box test on the fused FDT system, we consider the location of bounding box on the face region only and have less variations in the bounding box side.The higher-level observation is done on both KLT and fused algorithms providing them different video sequences as an input, the results are tabulated below:

	Bounding Box		All Frames		Variations in Bounding Box	
Video No.	KLT	Fused	KLT	Fused	KLT	Fused
1	yes	yes	yes	yes	no	no
2	yes	yes	yes	yes	no	no
3	yes	yes	yes	yes	no	no
4	yes	yes	yes	yes	no	no
5	yes	yes	yes	yes	little	no
6	yes	yes	yes	yes	no	no
7	yes	yes	yes	yes	little	little
8	yes	yes	yes	yes	no	no
9	yes	yes	yes	yes	no	no
10	yes	yes	no	no	moderate	moderate
11	yes	yes	yes	yes	no	no
12	yes	yes	yes	yes	no	no
13	yes	yes	yes	yes	no	no
14	yes	yes	yes	yes	no	no

	Bounding Box		All Frames		Variations in Bounding Box	
Video No.	KLT	Fused	KLT	Fused	KLT	Fused
15	yes	yes	yes	yes	little	little
16	yes	yes	yes	yes	out of bounds	moderate
17	yes	yes	yes	yes	large	lesser than KLT
18	yes	yes	yes	yes	little	little
19	yes	yes	no	yes	NA	little
20	yes	yes	yes	yes	no	no
21	yes	yes	yes	yes	no	no
22	yes	yes	yes	yes	out of bounds	moderate
23	yes	yes	no	no	large	large
24	yes	yes	no	no	large	large
25	yes	yes	yes	yes	no	no
26	yes	yes	yes	yes	No proper box	No proper box
27	yes	yes	yes	yes	large	lesser than KLT

Table 6.2: Black-Box Testing

Table 6.2 depicts black-box test results, this test is carried out to check the bounding box location and size around the face. We used the term Black-box testing because we are doing this test without changing or messing with the source code of the FDT system. The results of this test clearly depicts that the variations in the bounding box placement in fused FDT will be lesser than that of KLT algorithm.

This will be another criteria to justify the statement that the fused FDT system is more accurate and more precise in plotting the bounding box region around the face. There are many other tests that can be carried out on the system to check its efficiency and reliability, those tests can be done as another research work to rigorously test the new FDT system. The tests carried out in this chapter is to support the efficiency of the project.

We chose to fuse with KLT because it works based on point tracking. CAMSHIFT could have been used as an alternative but it works on hue value which is uncertain when illumination and pose varies.

Chapter 7

Functions and Terminologies

7.1 Functions used

- **addpath**: Adds directory to the search path.
- **assignin**: Assigns Variable in workspace i.e., assignin(WS,'name',V) assigns the variable 'name' in the workspace WS the value V. WS can be one of 'caller' or 'base'.
- **bbox2points**: Convert a rectangle into a list of points.
- **cornerPoints**: Object describes corner points, POINTS = cornerPoints(LOCATION) constructs a cornerPoints object from LOCATION, an M-by-2 array of [x y] coordinates.
- **createMask**: Returns a mask, or binary image, that is the same size as the input image with 1s inside the ROI object h and 0s everywhere else. The input image must be contained within the same axes as the ROI.
- **det**: det(A) returns the determinant of square matrix A.
- **detectMinEigenFeatures**: Returns a cornerPoints object, points. The object contains information about the feature points detected in a 2-D grayscale input image, I. The detectMinEigenFeatures function uses the minimum eigenvalue algorithm developed by Shi and Tomasi to find feature points.
- **estimateGeometricTransform**: Estimate geometric transform from matching point pairs.
- **figure**: Figure creates a new figure window using default property values. This new figure window becomes the current figure, and it displays on top of all other figures on the screen. The title of the figure is an integer value that is

not already used by an existing figure. MATLAB saves this integer value in the figure's Number property.

- **filter2**: filter2(h,X) filters the data in X with the two-dimensional FIR filter in the matrix h. It computes the result, Y, using two-dimensional correlation, and returns the central part of the correlation that is the same size as X.
- **find**: find(X) returns a vector containing the linear indices of each nonzero element in array X.
- **fspecial**: fspecial(type) creates a two-dimensional filter h of the specified type. fspecial returns h as a correlation kernel, which is the appropriate form to use with imfilter. type is a string having one of these values. fspecial('gaussian', hsize, sigma) returns a rotationally symmetric Gaussian lowpass filter of size hsize with standard deviation sigma (positive). hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3]; the default value for sigma is 0.5.
- **function**: Declares a function named myfun that accepts inputs x1,...,xM and returns outputs y1,...,yN. This declaration statement must be the first executable line of the function.
- **hold**: Retain current plot when adding new plots.
- **image**: image(C) displays the data in array C as an image. Each element of C specifies the color for 1 pixel of the image. The resulting image is an m-by-n grid of pixels where m is the number of columns and n is the number of rows in C. The row and column indices of the elements determine the centers of the corresponding pixels.
- **imshow**: Used to display image on the default output device.
- **initialize**: initialize(H,POINTS,I) initializes points to track and sets the initial video frame. The method sets the M-by-2 POINTS array of [x y] coordinates with the points to track, and sets the initial video frame, I. The input, POINTS, must be an M-by-2 array of [x y] coordinates. The input, I, must be a 2-D grayscale or RGB image, and must be the same size as the images passed to the step method.
- **insertObjectAnnotation**: Annotate truecolor or grayscale image or video stream.

- **insertShape**: Returns a truecolor image with shape inserted. The input image, I, can be either a truecolor or grayscale image. You draw the shapes by overwriting pixel values.

- **plot**: plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

- **release**: release(h) releases the interface and all resources used by the interface. Other interfaces on that object might still be active.

- **reshape**: reshape(A,sz) reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).

- **rgb2gray**: rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

- **size**: Size of triangulation connectivity list.

- **step**: step calculates the step response of a dynamic system. For the state-space case, zero initial state is assumed. When it is invoked with no output arguments, this function plots the step response on the screen.

- **trace**: trace(A) is the sum of the diagonal elements of the matrix A.

- **transformPointsForward**: transformPointsForward(tform,u,v) applies the forward transformation of tform to the input 2-D point arrays u and v and returns the point arrays x and y. The input point arrays u and v must be of the same size.

- **uigetfile**: uigetfile displays a modal dialog box that lists files in the current folder and enables you to select or enter the name of a file. If the file name is valid (and the file exists), uigetfile returns the file name as a string when you click Open. If you click Cancel (or the window's close box), uigetfile returns 0.

- **vision.CascadeObjectDetector**: vision.CascadeObjectDetector creates a System object, detector, that detects objects using the Viola-Jones algorithm. The ClassificationModel property controls the type of object to detect. By default, the detector is configured to detect faces.

- **vision.PointTracker**: The point tracker object tracks a set of points using the Kanade-Lucas-Tomasi (KLT), feature-tracking algorithm. You can use the point tracker for video stabilization, camera motion estimation, and object tracking.

It works particularly well for tracking objects that do not change shape and for those that exhibit visual texture. The point tracker is often used for short-term tracking as part of a larger tracking framework. As the point tracker algorithm progresses over time, points can be lost due to lighting variation, out of plane rotation, or articulated motion. To track an object over a long period of time, you may need to reacquire points periodically.

- **vision.VideoFileReader**: The VideoFileReader object reads video frames, images, and audio samples from a video file. The object can also read image files.
- **vision.VideoPlayer**: The VideoPlayer object can play a video or display image sequences.
- **which**: Which item displays the full path for item.
- **zeros**: Creates array of all zeros.

7.2 Major terms

- **cnt**: Variable that keeps the count of number of frames that contains the bounding box.
- **facedetector**: Holds the handle of vision.cascadeObjectDetector().
- **h_im**: Variable that holds the handle of the image that is going to be masked.
- **points**: Matrix that contains Eigen corner points along with Harris corner points.
- **posc**: Array that contains column values of detected corner points.
- **posr**: Array that contains row values of detected corner points.
- **R(i,j)**: Array that contains values of all corner points detected.
- **tc**: Variable that hold centroid column coordinate value.
- **tr**: Variable that hold centroid row coordinate value.
- **valrc**: Array that consists of Edge coordinate values and centroid value.
- **videoFrame**: Hold the handle of one frame of video which is extracted using step() function.

Chapter 8

Snapshots

This chapter consists the results drawn after successful execution of our project. Since our project include GUI output as the result, we include snapshots of the GUI's to show the outcome of the project.

8.1 Graphical User Interface Design Environment (GUIDE)

- The Quick start menu of GUIDE to design GUI.

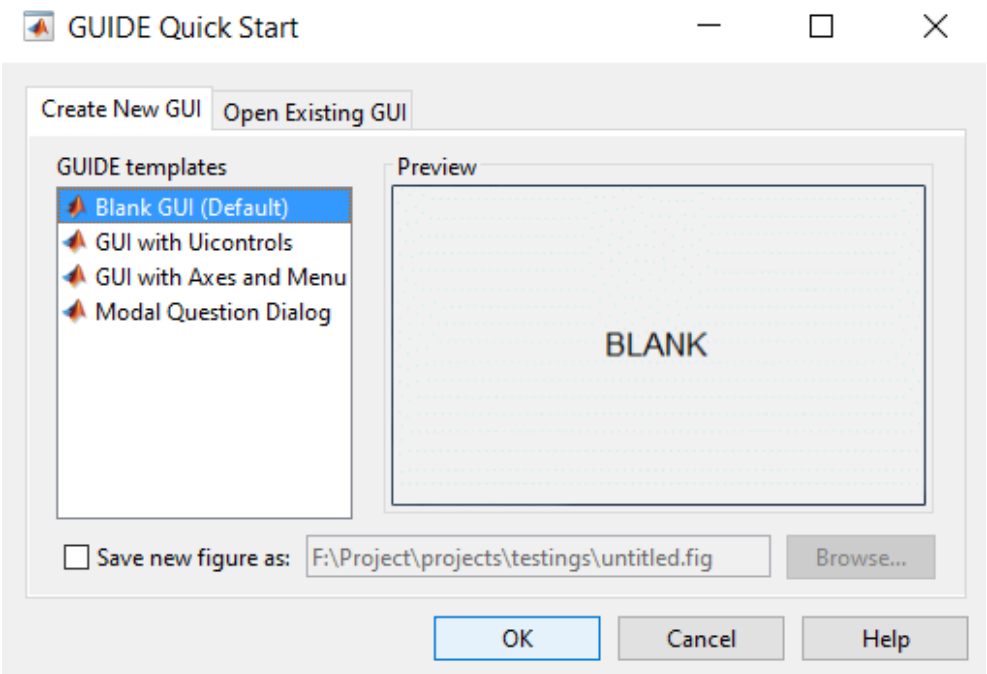

Figure 8.1: GUIDE Quick start

- The workbench of GUIDE where we have workspace and many components that can be added to the GUI such as Push button,Radio button,static text, pop-up menu and so on.

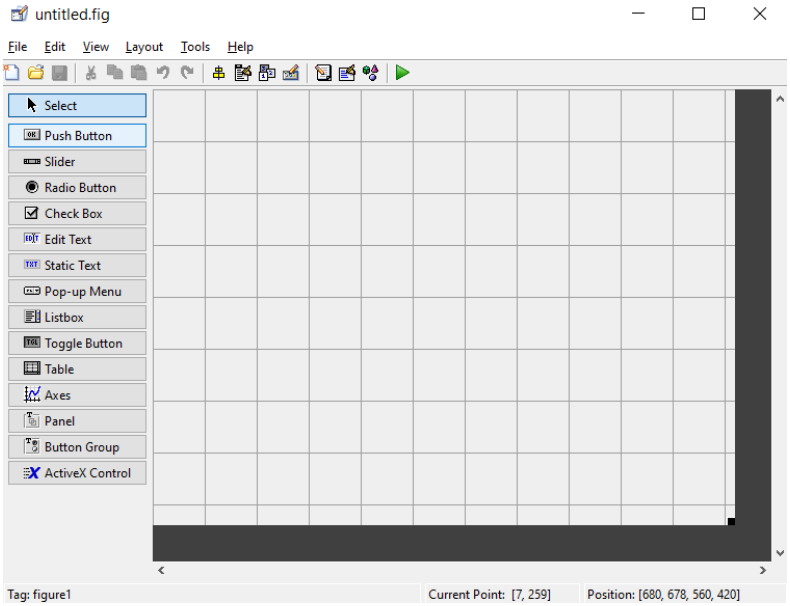

Figure 8.2: GUIDE workbench

- The GUI developed in workbench.

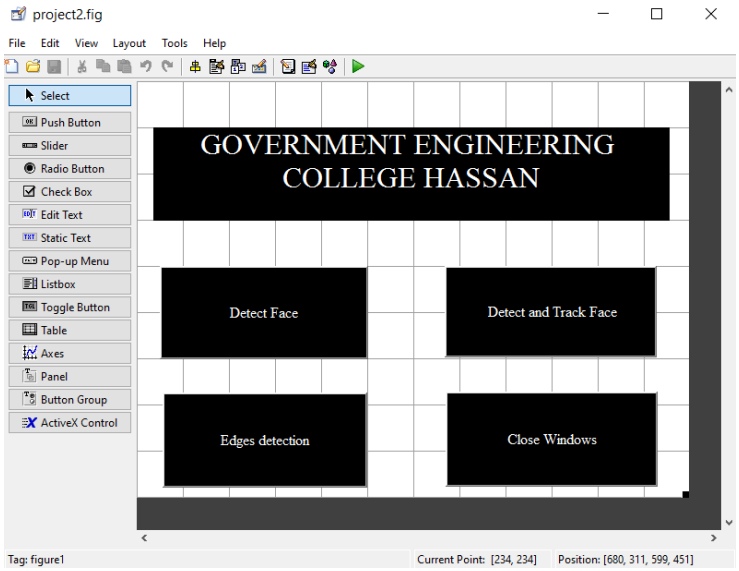

Figure 8.3: GUIDE GUI

- Final GUI after building the it in GUIDE workspace. Each option in the in the GUI is linked to a certain function such as face detection, edge & corner

detection, face tracking. Callbacks are designed in such a way that on click of each options the corresponding GUI pops-up.

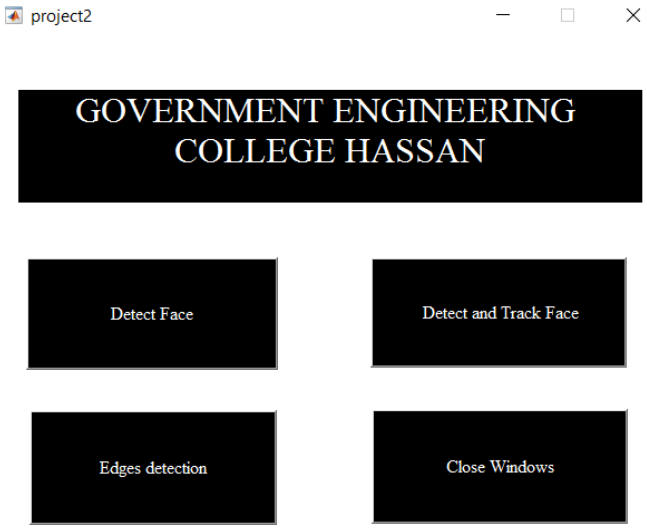

Figure 8.4: GUI of FDT system

8.2 Detection of Face

This section deals with detection of the face in the first frame of the video sequence.

8.2.1 Video 1

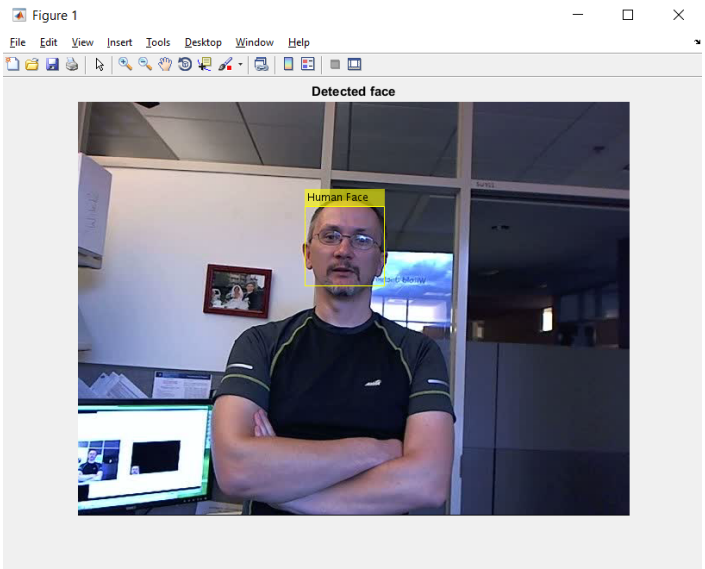

Figure 8.5: Detected Face of video 1

8.2.2 Video 2

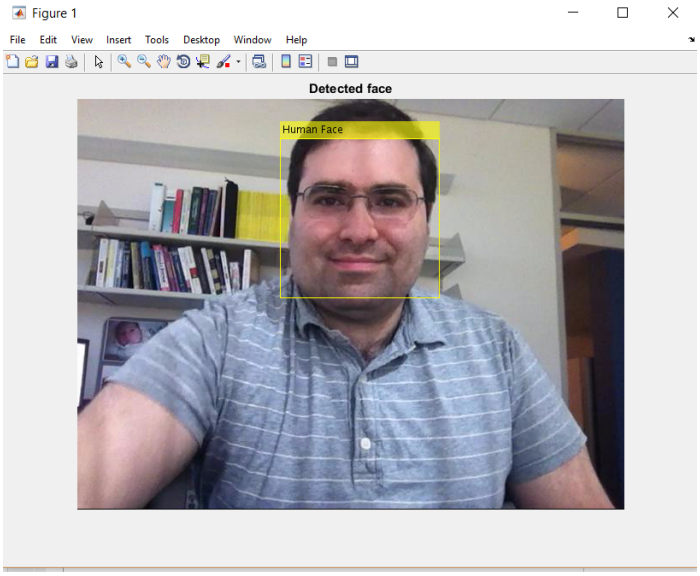

Figure 8.6: Detected Face of video 2

8.3 Edge & Corner detection

8.3.1 Video 1

- The rectangle is drawn around the detected face region to avoid that region from masking.

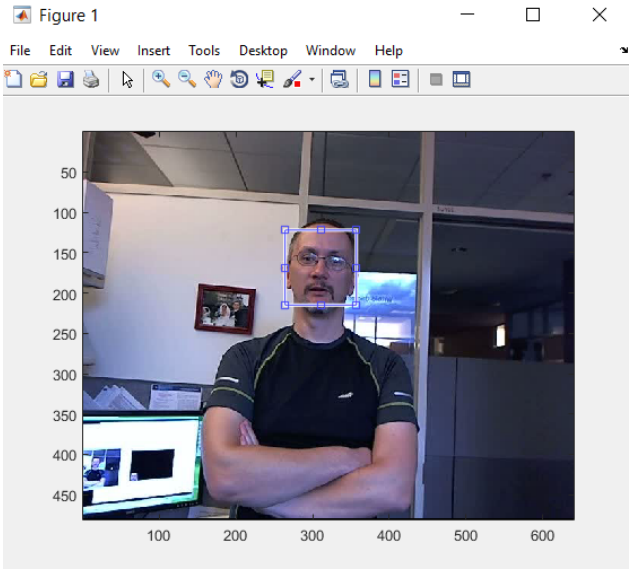

Figure 8.7: Face Region of video 1

- Region outside rectangle is masked and the detected corner points are marked inside the face region.

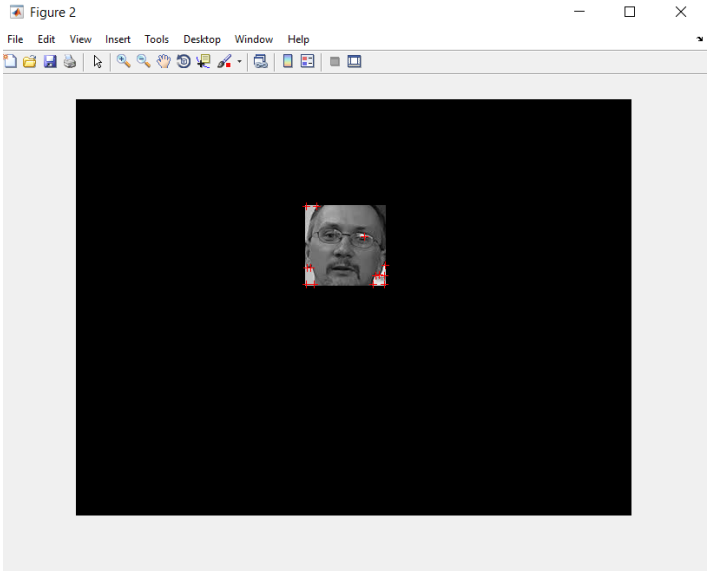

Figure 8.8: Masked Region of video 1

- The detected points alongwith centroid point in the face region of first frame.

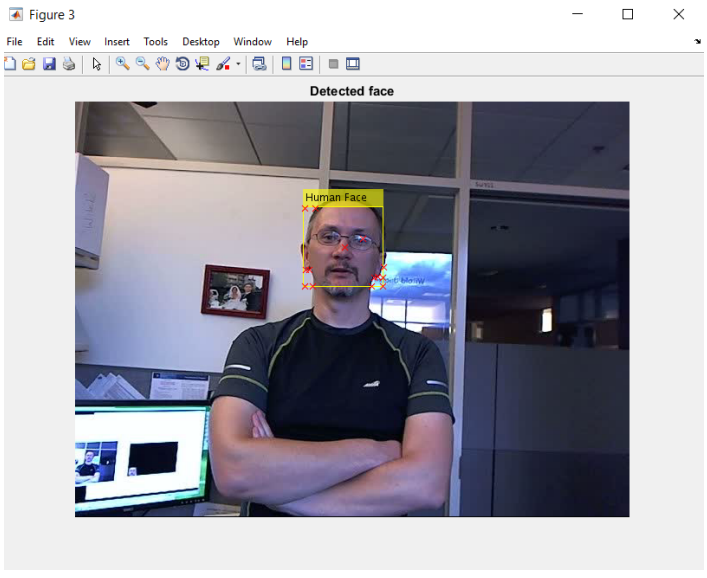

Figure 8.9: Corner and Centroid points of video 1

8.3.2 Video 2

- The rectangle is drawn around the detected face region to avoid that region from masking.

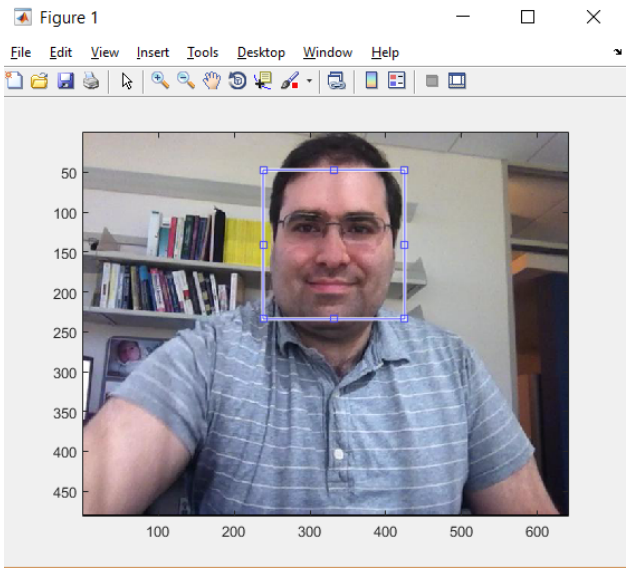

Figure 8.10: Face Region of video 2

- Region outside rectangle is masked and the detected corner points are marked inside the face region.

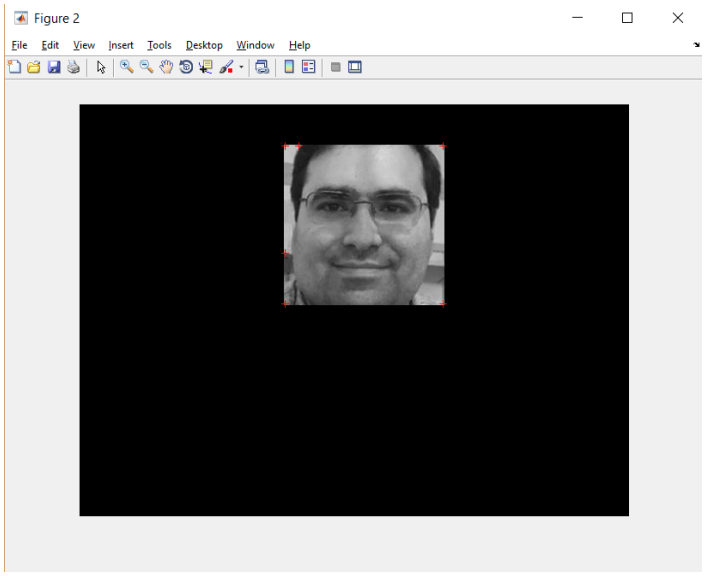

Figure 8.11: Masked Region of video 2

- The detected points alongwith centroid point in the face region of first frame.

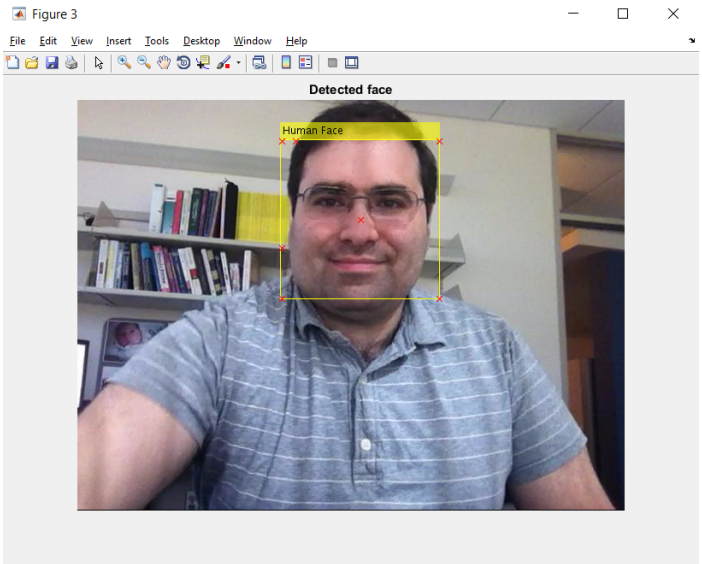

Figure 8.12: Corner and Centroid points of video 2

8.4 Face Tracking

8.4.1 Video 1

- The rectangle is drawn around the detected face region to track in the further frames.

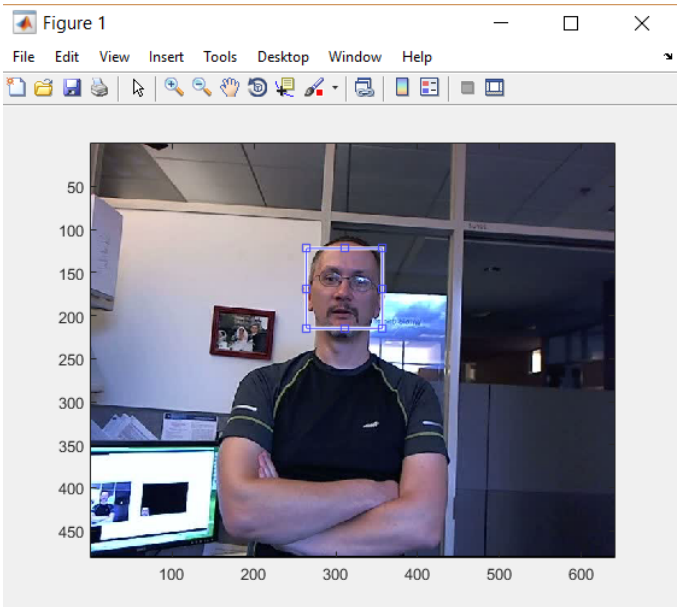

Figure 8.13: Face Region of video 1

- The detected Eigen points fused with Harris corner points and centroid point.

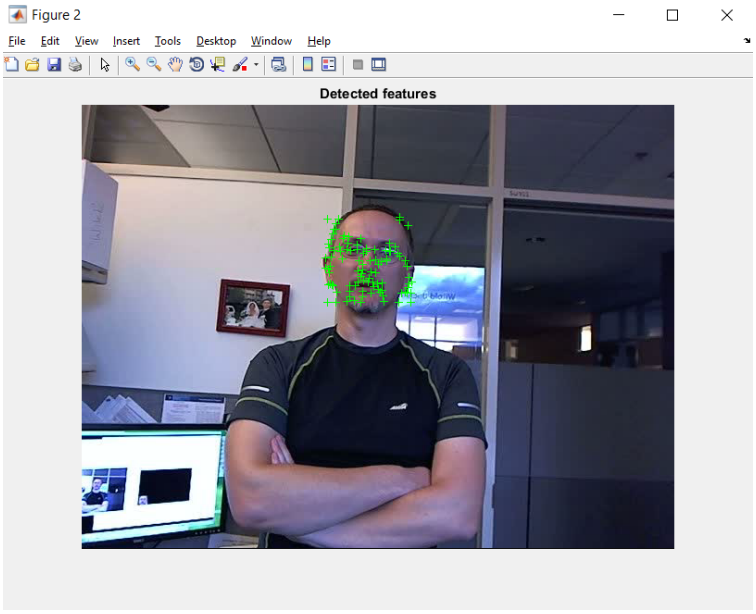

Figure 8.14: Eigen and Corner points of video 1

- Tracking of face wherever it moves in the further frames of the video sequence.

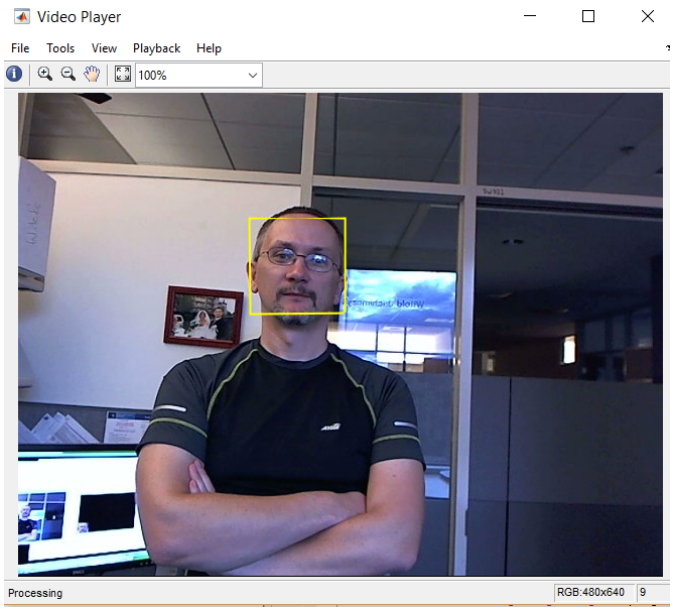

Figure 8.15: Human position 1 of video 1

- Human position 2

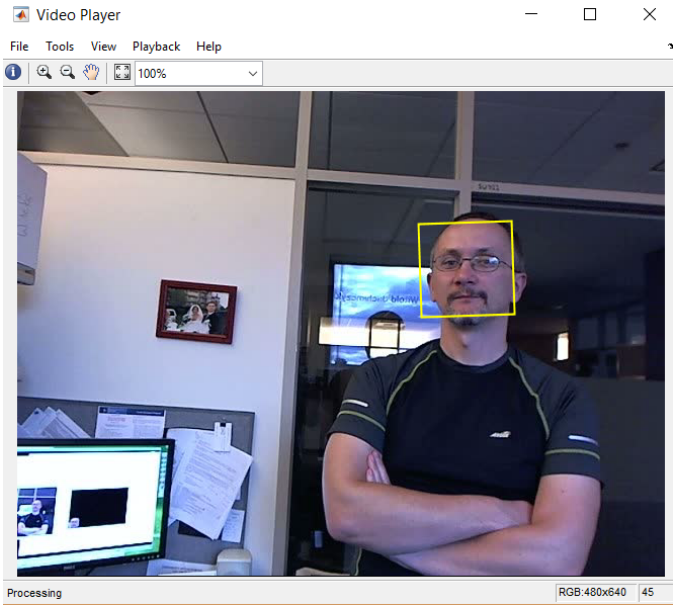

Figure 8.16: Human position 2 of video 1

- Human position 3

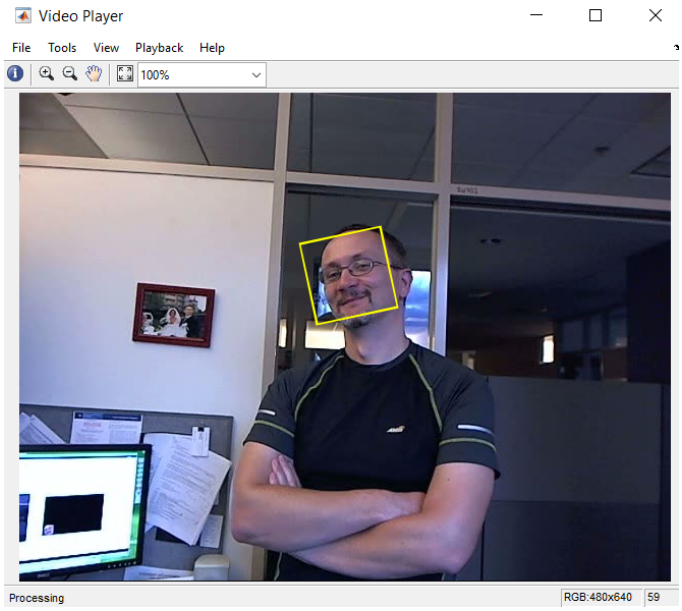

Figure 8.17: Human position 3 of video 1

8.4.2 Video 2

- The rectangle is drawn around the detected face region to track in the further frames.

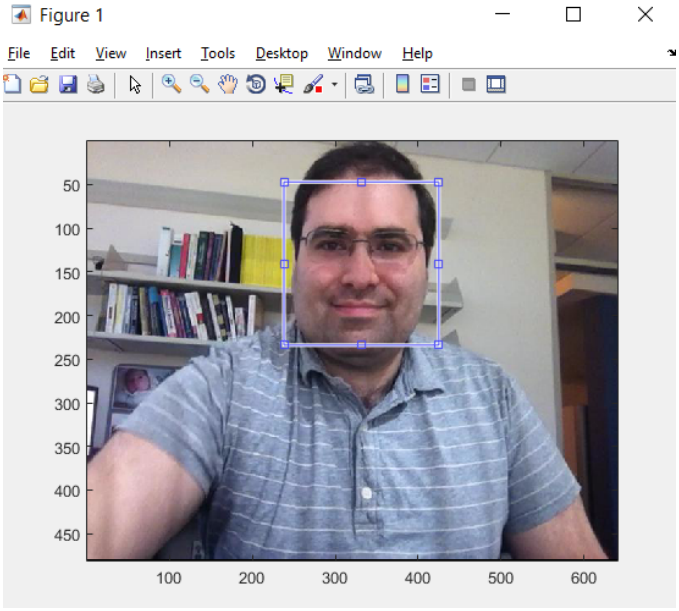

Figure 8.18: Face Region of video 2

- The detected Eigen points fused with Harris corner points and centroid point.

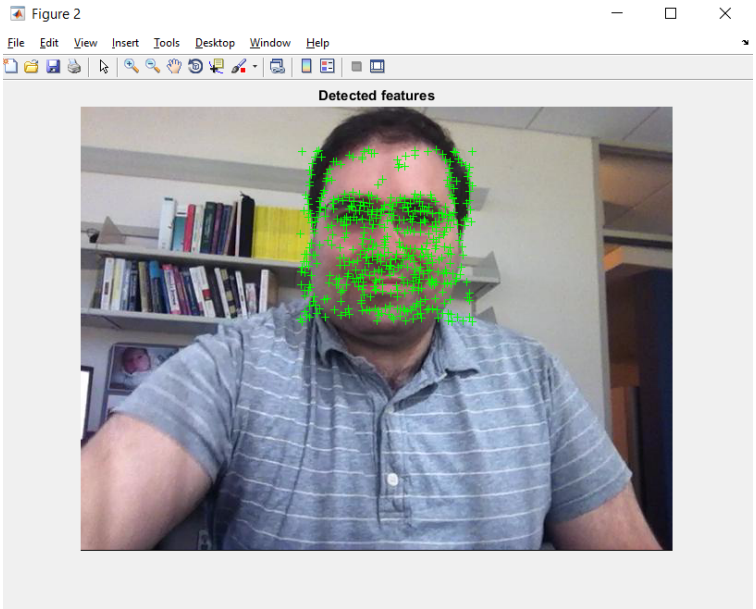

Figure 8.19: Eigen and Corner points of video 2

- Tracking of face wherever it moves in the further frames of the video sequence.

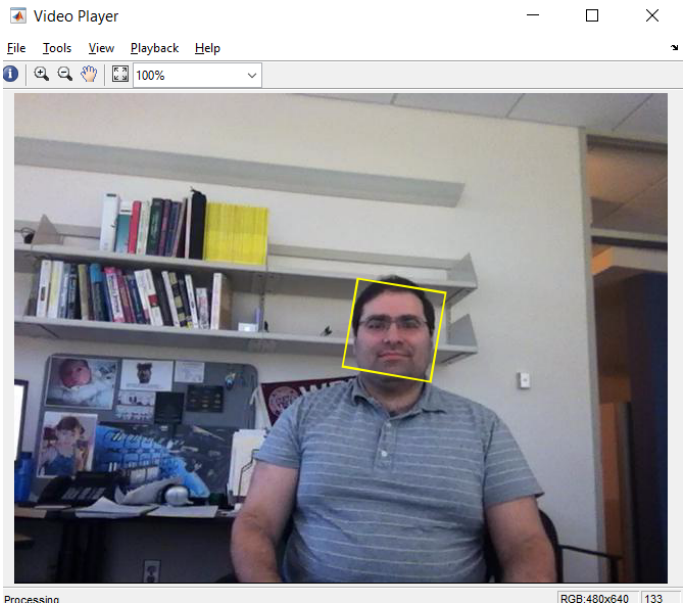

Figure 8.20: Human position 1 of video 2

- Human position 2

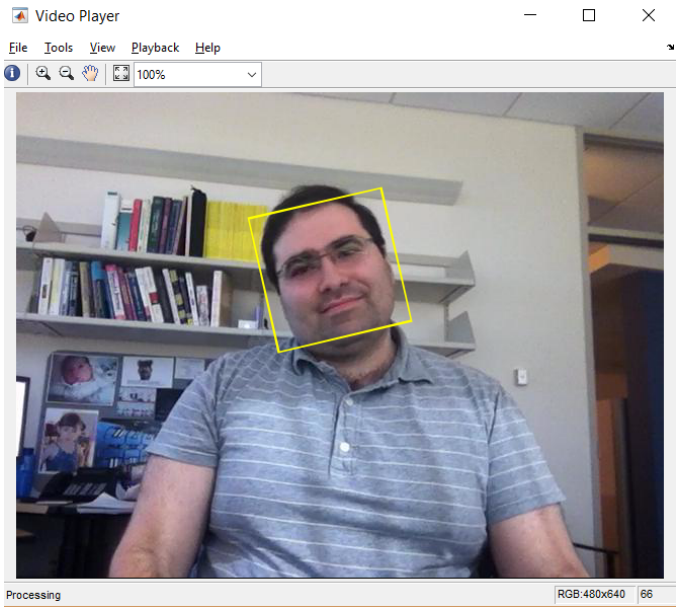

Figure 8.21: Human position 2 of video 2

- Human position 3

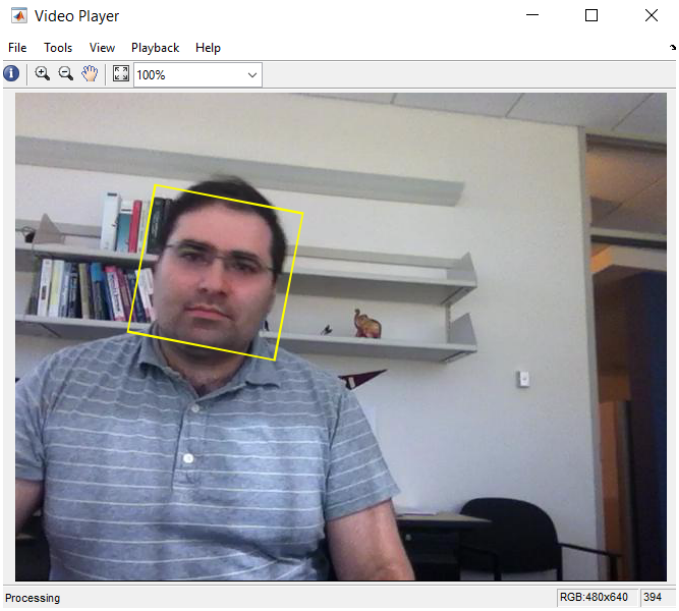

Figure 8.22: Human position 3 of video 2

Chapter 9

Conclusion & Future Works

9.1 Conclusion

The main theoretical aim was to develop a new algorithm or fused algorithm to overcome few shortcomings of the Video Processing, certain challenges are inevitable and cannot solve all the challenges at a time. Even a small improvement in this area is very phenomenal in the field of video processing because of which there is lot of research going on in this field rigorously. In the project we have developed a fused Face detection and tracking system which works based on the point tracking as that of KLT algorithm. From the test reports we could clearly observe that fused FDT algorithm tracks face in few more frames than KLT algorithm alone would have achieved and also because we use centroid as one of the point while tracking, the chances of variation in bounding box size and shape is very negligible compared to KLT algorithm alone.

The theoretical assumption which has been made in the beginning of the project is being practically proved by the results that have been tabulated in the testing chapter of this report.

9.2 Future Enhancements

There is proverb saying that "Imagination is more important than knowledge. Knowledge is limited but imagination encircles the world", it is true in the field of Digital Image Processing; the more we imagine the more we explore and the more we explore the more problems we encounter. Video and video types are changing day by day, video processing is becoming complex time to time. An algorithm because of its robustness which was being used few years ago, is failing to overcome the challenges that are being faced nowadays. This is one major reason for continuous researches that is being carried out in this area.

Even after so many researches being done in this area, till now no algorithm is made which gives 100% results when the algorithm is subjected to different videos having different constraints of its own. By stating all these we want to mention that even after being little more accurate in tracking than that of KLT algorithm, the fused FDT system developed will also abide with many constraints and restrictions, few of them can be tackled to give even better accuracy.
Few future enhancements that can be done above the current state of project are:

- Modify Viola-Jones algorithm to remove the constraint of face being present in first frame itself.
- Faces can be detected in further frames using a loop.
- Generating more points using mid-point theorem from edge and corner points.
- Eliminating the use of Eigen features for tracking, using point tracker only for the corner points and other generated points.
- Reducing the execution time by simplifying the code statements.

References

[1] Ranganatha S, Dr.Y P Gowramma,"Face Recognition Techniques: A Survey", International Journal for Research in Applied Science and Engineering Technology(IJRASET), Vol. 3, Issue IV, pp. 630-635, April 2015.

[2] R. Jagathishwaran, K.S. Ravichandran and Premaladha Jayaraman, "A Survey on Face Detection and Tracking", World Applied Sciences Journal, ISSN 1818-4952, IDOSI Publications, pp. 140-145, 2014.

[3] WU Lushen, WU Peimin, MENG Fanwen, "A Fast Face Detection for Video Sequences", Second International Conference on Intelligent Human-Machine Systems and Cybernetics, Nanchang, China, 2010.

[4] M. Kim, S. Kumar, V. Pavlovic and H. Rowley. "Face Tracking and Recognition with Visual Constraints in Real-World Videos". IEEE Conf. Computer Vision and Pattern Recognition. 2008.

[5] Han-Pang Huang and Chun-Ting Lin, "Multi-CAMSHIFT for Multi-View Faces Tracking and Recognition",Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics, Kunming, China, December 17-20, 2006.

[6] Paul Viola and Michael Jones,"Fast multi-view face detection", Mitsubishi Electric Research Laboratories,TR2003-96, August 2003.

[7] Dorin Comaniciu and Peter Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 5, May 2002.

[8] Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", in CVPR Proc. IEEE Comput. Soc. Conf., pp. 511-518, 2001.

[9] H. Rowley, S. Baluja, and T. Kanade,"Neural network-based face detection". In IEEE Patt.Anal. Mach. Intel., Vol. 20, pp. 22-38, 1998.

[10] G.R. Bradski "Real Time Face and Object Tracking as a Component of a Perceptual User Interface", Proceedings of the 4th IEEE Workshop on Applications of Computer Vision, 1998.

[11] Cheng, Yizong, "Mean Shift, Mode Seeking, and Clustering". IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE), doi:10.1109/34.400568, Vol. 17, pp. 790-799, August 1995.

[12] Jianbo Shi and Carlo Tomasi, "Good Features to Track", IEEE Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.

[13] Carlo Tomasi and Takeo Kanade, "Detection and Tracking of Point Features", Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[14] Bruce D. Lucas and Takeo Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", International Joint Conference on Artificial Intelligence, pp. 674-679, 1981.

[15] http://in.mathworks.com/products/image/index.html

[16] http://in.mathworks.com/help/images/

[17] http://www.tutorialspoint.com/dip/ - for matlab tutorials.

[18] http://in.mathworks.com/academia/students.html?s_tid=acmain_sp_gw_bod

[19] http://in.mathworks.com/help/matlab/creating_guis/about-the-simple-programmatic-gui-example.html

[20] Rafel C Gonzalez and Richard E Woods, "Digital Image Processing", 3rd Edition, Pearson Education, 2003.

[21] Milan Sonka, Vaclav Hlavac and Roger Boyle, " Image Processing, Analysis and Machine Vision", 2nd Edition, Thomoson Learning, 2001.