Institute of Engineering and Management, Kolkata

Department of Electronics and Communication Engineering

Subject Name: Digital Signal Processing Laboratory

Subject Code: PCCEC-592

Topic: 5th Semester Lab Project

Group Members:

Samaroho Chattopadhyay, Sec- A, Roll No: – 22,
Enrollment No: 12020002003033

Aim:- Creating a Real time Face , Eyes and Upper Body detection system using MATLAB.

Introduction:-

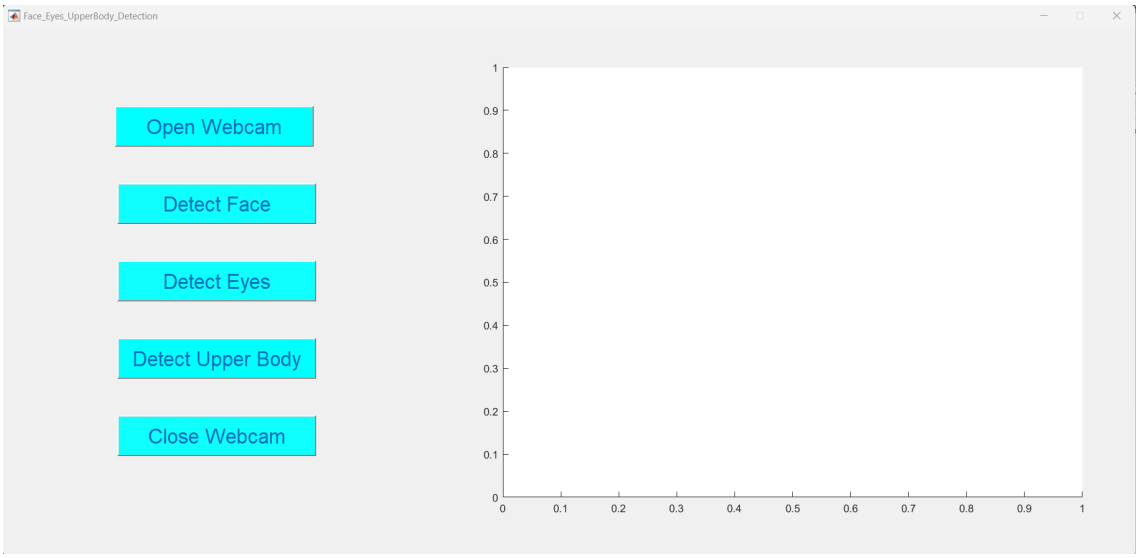
Object detection and tracking are important in many computer vision applications, including activity recognition, automotive safety and surveillance. Presented here is an face detection using MATLAB system that can detect not only a human face but also eyes and upper body . Face detection is an easy and simple task for humans, but not so for computers. It has been regarded as the most complex and challenging problem in the field of computer vision due to large intra-class variations caused by the changes in facial appearance, lighting and expression. Such variations result in the face distribution to be highly nonlinear and complex in any space that is linear to the original image space . Face detection is the process of identifying one or more human faces in images or videos. It plays an important part in many biometric, security and surveillance systems, as well as image and video indexing systems.

Graphical User Interface(GUI):-

A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons, menus and other visual indicators or representations (graphics). GUIs graphically display information and related user controls, unlike text-based interfaces, where data and commands are strictly in text. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or by a finger on a touch screen. The first human/computer text interface worked through keyboard input, with what is called a prompt (or DOS prompt). Commands were typed on a keyboard at the DOS prompt to initiate responses from a computer. The use of these commands and the need for exact spelling created a cumbersome and inefficient interface. Arguably, the introduction and popularization of GUIs is one of the most important factors that made computer and digital technologies more accessible to average, less tech-savvy users. GUIs are, in fact, created to be intuitive enough to be operated even by

relatively unskilled personnel who have no knowledge of any programming language. Rather than being fundamentally machine-centered, they are now the standard in software application programming because their design is always user-centered.

This face detection using MATLAB program can be used to detect a face, eyes and upper body on pressing the corresponding buttons. The program output screen is presented below:-



Perks of using a GUI:-

- Simplicity.
- It is visually appealing and makes anyone get involved in working with the machine.
- Even a guy with no computer knowledge can use the computer and perform basic functions. GUI is responsible for that.

- Searching becomes very easy as GUI provides a visual representation of the files present and provides details about it.
- Each and every response from the computer is visually communicated through GUI.
- A user with no computer knowledge can literally start learning about the machine because of GUI as it provides scope for users to explore and provides discoverability.

GUIDE feature in MATLAB:-

The guide command opens GUIDE, a UI design environment. The GUIDE environment provides a set of tools for creating user interfaces (UIs). These tools simplify the process of laying out and programming UIs.

guide(filename) opens the specified MATLAB figure file for editing in GUIDE. If the figure file is not on the MATLAB path, specify the full path. Only one filename can be opened at a time.

guide(figs) opens each of the Figure objects in figs in a separate copy of the GUIDE design environment. Use this syntax if you want to edit one or more preexisting figures in GUIDE that have been saved to variables.

VIOLA-JONES ALGORITHM:-

There are different types of algorithms used in face detection. Here, we have used Viola-Jones algorithm for face detection using MATLAB program. This algorithm works in following steps:

1. Creates a detector object using Viola-Jones algorithm

2. Takes the image from the video

3. Detects features

4. Annotates the detected features

Viola Jones algorithm is named after two computer vision researchers who proposed the method in 2001, Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

Given an image(this algorithm works on grayscale image), the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

The Viola Jones algorithm has four main steps, which we shall discuss in the sections to follow:

1. Selecting Haar-like features

2. Creating an integral image
3. Running AdaBoost training
4. Creating classifier cascades

What are Haar-Like Features?

Haar-like features are digital image features used in object recognition. All human faces share some universal properties of the human face like the eyes region is darker than its neighbour pixels, and the nose region is brighter than the eye region.

There are 3 types of Haar-like features that Viola and Jones identified in their research:

1. Edge features
2. Line-features
3. Four-sided features

What are Integral Images?

An integral image (also known as a summed-area table) is the name of both a data structure and an algorithm used to obtain this data structure. It is used as a quick and efficient way to calculate the sum of pixel values in an image or rectangular part of an image.

How is AdaBoost used in viola jones algorithm?

Next, we use a Machine Learning algorithm known as AdaBoost. The number of features that are present in the 24×24 detector window is nearly 160,000, but only a few of these features are important to identify a face. So we use the AdaBoost algorithm to identify the best features in the 160,000 features. In the Viola-Jones algorithm, each Haar-like feature represents a weak learner. To decide the type and size of a feature that goes into the final classifier, AdaBoost checks the performance of all classifiers that you supply to it. The classifiers that performed well are given higher importance or weight. The final

result is a strong classifier, also called a boosted classifier, that contains the best performing weak classifiers. So when we're training the AdaBoost to identify important features, we're feeding it information in the form of training data and subsequently training it to learn from the information to predict. So ultimately, the algorithm is setting a minimum threshold to determine whether something can be classified as a useful feature or not.

What are Cascading Classifiers?

We have a 24×24 window which we slide over the input image, and we need to find if any of those regions contain the face. The job of the cascade is to quickly discard non-faces, and avoid wasting precious time and computations. Thus, achieving the speed necessary for real-time face detection. We set up a cascaded system in which we divide the process of identifying a face into multiple stages. In the first stage, we have a classifier which is made up of our best features, in other words, in the first stage, the subregion passes through the best features such as the feature which identifies the nose bridge or the one that identifies the eyes. In the next stages, we have all the remaining features.

Components required:-

MATLB VERSIONS OF:

1. 2012Ra

2. 2018Ra

>Command window

>editor window

>run

>saving the code of loading the data sheet in file

>saving implementation code in another file

Source Code:-

```
function varargout =
Face_Eyes_UpperBody_Detection(varargin)
% FACE_EYES_UPPERBODY_DETECTION MATLAB code for
Face_Eyes_UpperBody_Detection.fig
%      FACE_EYES_UPPERBODY_DETECTION, by itself, creates a
new FACE_EYES_UPPERBODY_DETECTION or raises the existing
%      singleton*.
%
%      H = FACE_EYES_UPPERBODY_DETECTION returns the
handle to a new FACE_EYES_UPPERBODY_DETECTION or the
handle to
%      the existing singleton*.
%
%
FACE_EYES_UPPERBODY_DETECTION('CALLBACK',hObject,eventData
,handles,...) calls the local
%      function named CALLBACK in
FACE_EYES_UPPERBODY_DETECTION.M with the given input
arguments.
%
%
FACE_EYES_UPPERBODY_DETECTION('Property','Value',...)
creates a new FACE_EYES_UPPERBODY_DETECTION or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before
Face_Eyes_UpperBody_Detection_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to
Face_Eyes_UpperBody_Detection_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose
"GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Face_Eyes_UpperBody_Detection

% Last Modified by GUIDE v2.5 27-Sep-2022 18:05:40

% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Face_Eyes_UpperBody_Detection_OpeningFcn, ...
                   'gui_OutputFcn',
@Face_Eyes_UpperBody_Detection_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Face_Eyes_UpperBody_Detection
is made visible.
function Face_Eyes_UpperBody_Detection_OpeningFcn(hObject,
eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to
Face_Eyes_UpperBody_Detection (see VARARGIN)

% Choose default command line output for
Face_Eyes_UpperBody_Detection
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Face_Eyes_UpperBody_Detection wait for user
response (see UIRESUME)
% uiwait(handles.figure1);
```

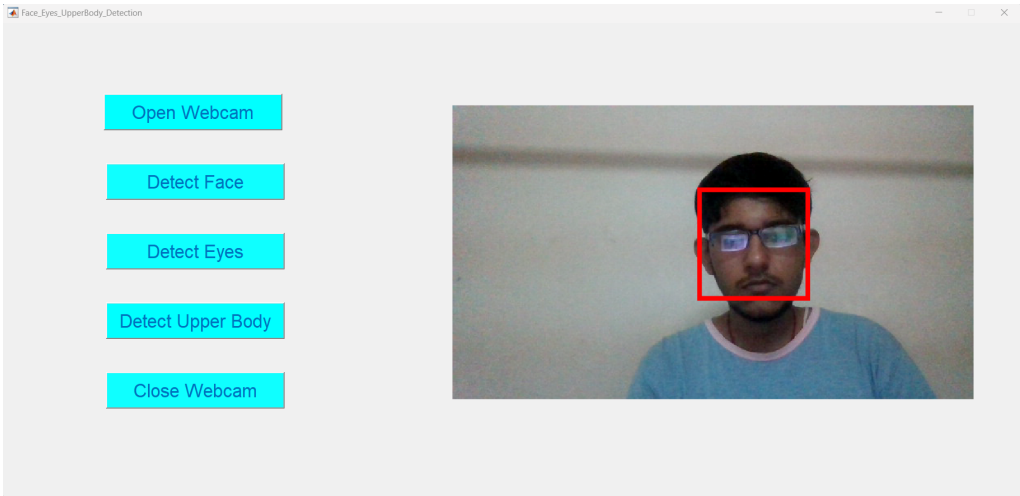
```
% --- Outputs from this function are returned to the
command line.
function varargout =
Face_Eyes_UpperBody_Detection_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Open_Webcam.
function Open_Webcam_Callback(hObject, eventdata, handles)
% hObject    handle to Open_Webcam (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global c;
global tvar;
tvar=true;
c=webcam;
while(true)
    if(tvar==false)
        break;
    else
        e=c.snapshot;
        axes(handles.axes2);
   imshow(e);
   drawnow;
    end
end


% --- Executes on button press in Detect_Face.
function Detect_Face_Callback(hObject, eventdata, handles)
% hObject    handle to Detect_Face (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
FaceDetect=vision.CascadeObjectDetector;
global c;
global tvar;
tvar=true;
c=webcam();
while(true)
    if(tvar==false)
        break;
    else
        e=c.snapshot;
        Box=step(FaceDetect,e);
        imshow(e);
        hold on;
        for i=1:size(Box,1)

rectangle('Position',Box(i,:),'Linewidth',5,'LineStyle','-
','EdgeColor','r');
        end
        hold off;
   drawnow;
    end
end

% --- Executes on button press in Detect_Eyes.
function Detect_Eyes_Callback(hObject, eventdata, handles)
% hObject    handle to Detect_Eyes (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
EyeDetect=vision.CascadeObjectDetector('EyePairBig');
EyeDetect.MinSize=[11 45];
EyeDetect.MergeThreshold=16;
global c;
global tvar;
tvar=true;
c=webcam();
while(true)
    if(tvar==false)
        break;
    else
        e=c.snapshot;
        Box=step(EyeDetect,e);
        imshow(e);
        hold on;
        for i=1:size(Box,1)
```

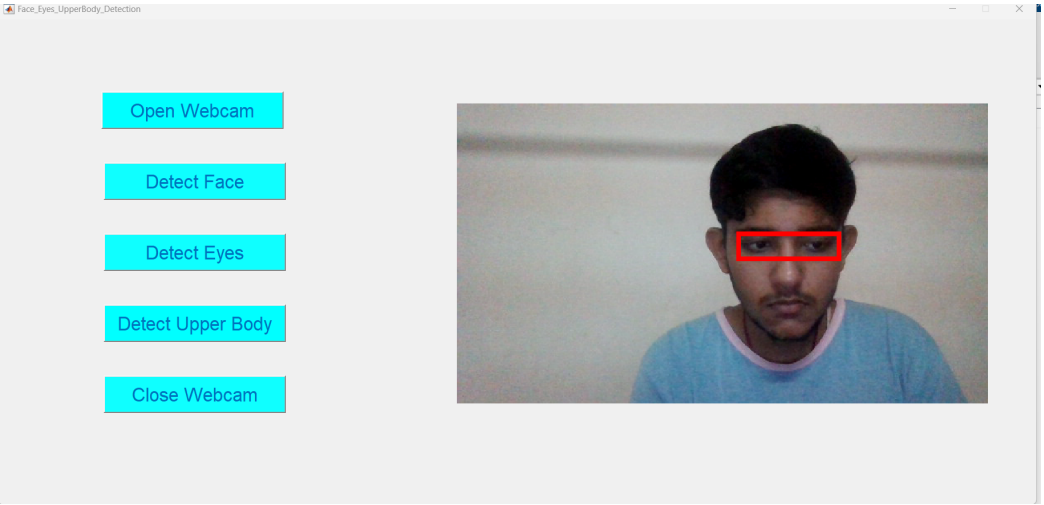
```
rectangle('Position',Box(i,:),'Linewidth',5,'LineStyle','-
','EdgeColor','r');
        end
        hold off;
   drawnow;
    end
end

% --- Executes on button press in Detect_Upper_Body.
function Detect_Upper_Body_Callback(hObject, eventdata,
handles)
% hObject    handle to Detect_Upper_Body (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
UBDetect=vision.CascadeObjectDetector('UpperBody');
UBDetect.MinSize=[60 60];
UBDetect.MergeThreshold=10;
global c;
global tvar;
tvar=true;
c=webcam();
while(true)
    if(tvar==false)
        break;
    else
        e=c.snapshot;
        Box=step(UBDetect,e);
        imshow(e);
        hold on;
        for i=1:size(Box,1)

rectangle('Position',Box(i,:),'Linewidth',5,'LineStyle','-
','EdgeColor','r');
        end
        hold off;
   drawnow;
    end
end

% --- Executes on button press in Close_Webcam.
function Close_Webcam_Callback(hObject, eventdata,
handles)
% hObject    handle to Close_Webcam (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global tvr;
tvr=false;
clear global c;
axes(handles.axes2);
imshow('C:\Users\HP\Downloads\ThankYou.jpg');
```

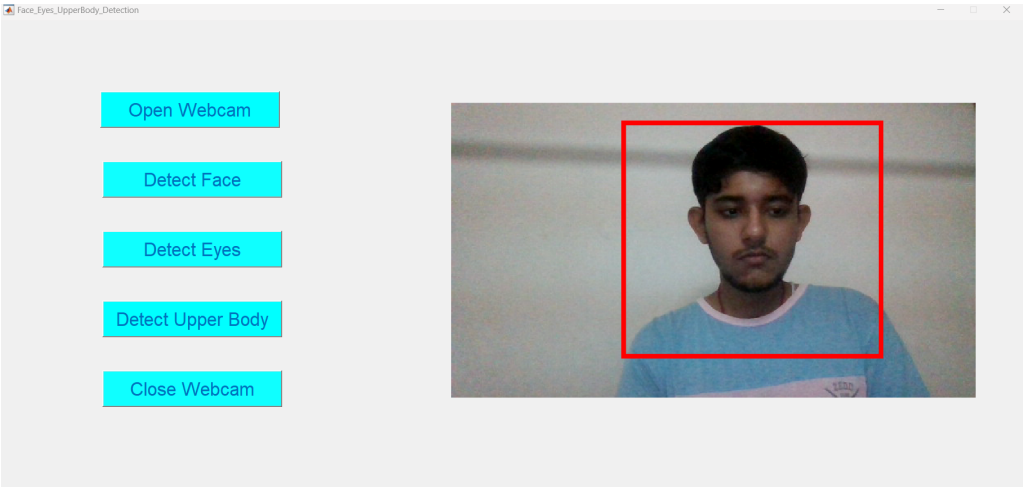
Face Detection:-



Eye Detection:-



Upper Body Detection:-



Video Demonstration:-

https://drive.google.com/file/d/18LC6Zr6MnUmwN7g4oyIuMHJl2HDz0yLm/view?usp=sharing

Explanation of the code:-

I have used the GUIDE feature in MATLAB to design the Graphical User Interface for implementation of different functionalities of face ,eyes and upper body detection.

In the GUI I have created five buttons-open webcam ,Detect face ,Detect eyes ,Detect upper body and close webcam. I have also added an axis to display the implementation for the users. After creating the GUI it is stored as a .fig file. I have added live logic to the buttons and axis by incorporating my code of face detection in their respective UI callback methods .Whenever a button is pressed the UI callback method for that button is executed and the programming logic under that button is activated implementing the functionalities.

The main idea of the programming logic is to use the webcam integrated in the laptop or the webcam in the PC to capture live. A control variable is tvar is declared and initialized with Boolean value true and until the variable is true the functionalities are accessible but

once the stop webcam button is pressed the tvar variable is declared as false so the functionalities are no longer accessible.

The logic of the functionalities include running an infinite loop until the control variable tvar is true.
The cascade object detector uses the Viola-Jones algorithm to detect people's faces, noses, eyes, mouth, or upper body. You can also use the Image Labeler to train a custom classifier to use with this System object.

I have created the vision.CascadeObjectDetector object and set its properties according to the feature I am detecting which differs from feature to feature. I have called the object with arguments, as if it were a function later in the code. Inside the infinite while loop I have used a snapshot function to constantly capture photo from the live feed and process each image individually. Each image is converted to greyscale image and then the image is passed as an argument along with the cascade object detector object to a step function which helps to run an algorithm of the object passed to it as an argument. The step function returns the four corner coordinates of the feature in the image .A rectangle function is used to draw a box around that feature to detect that feature in the image and the properties of the rectangle like line width ,line colour along with box variable that consists of the coordinates of the feature in the image are passed onto the rectangle function to display the box around that feature in the live feed to be displayed by the user.

Finally, a hold off function is used to project all the processed images constantly being produced by the statements executing in the infinite while loop onto the same window to maintain a continuity and display as if the features are being detected in the live feed itself. Also, a drawnow function is used to produce an animation effect such that all the images produced onto a common window are seamlessly displayed displaying in the form an animation in continuity and creating that real life feature detection possible.

Applications:-

Facial motion capture:-

Several commercial companies are developing products that have been used, but are rather expensive .It is expected that this will become a major input device for computer games once the software is available in an affordable format, but the hardware and software do not yet exist, despite the research for the last 15 years producing results that are almost usable.

Facial recognition:-

Face detection is used in biometrics, often as a part of (or together with) a facial recognition system. It is also used in video surveillance, human computer interface and image database management.

Photography:-

Some recent digital cameras use face detection for autofocus. Face detection is also useful for selecting regions of interest in photo slideshows that use a pan-and-scale Ken Burns Effect.

Modern appliances also use smile detection to take a photograph at an appropriate time.

Marketing:-

Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face. Once the information is collected, a series of advertisements can be played that is specific toward the detected race/gender/age.

An example of such a system is OptimEyes and is integrated into the Amscreen digital signage system.

Emotional Inference:-

Face detection can be used as part of a software implementation of emotional inference. Emotional inference can be used to help people with autism under stand the feelings of people around them.

Lip Reading:-

Face detection is essential for the process of language inference from visual cues. Automated lip reading has applications to help computers determine who is speaking which is needed when security is important.

Attention Tracking in Retail:-

Instead of asking a pre-selected focus group to wear glasses for a market research study on shelf attention, potentially influencing their behavior, 3D eye tracking allows for accurately tracking passersby's gaze accurately and non-intrusively. By monitoring the customer's attention, retailers can then analyze product performance. Attention tracking helps companies with valuable analytics.

Eye tracking for Automotive:-

Eye tracking for the automotive industry will make roads safer. The limited tracking box and the need for a multisensor set up to track the gaze in 3D limit the applicability of current eye tracking technologies in cars. Gaze recognition will be solving two main issues concerning human-car interaction.

Pedestrian Detection:-

The pedestrian detection task which aims to predict bounding-boxes of all the pedestrian instances in an image which is of paramount importance is done by upper body detection.

CONCLUSION:-

Face recognition is still a challenging problem in the field of computer vision. It has received a great deal of attention over the past years because of its several applications in various domains. Although there is strong research effort in this area, face recognition systems are far from ideal to perform adequately in all situations form real world. Paper presented a brief survey of issues methods and applications in area of face recognition. There is much work to be done in order to realise methods that reflect how humans recognise faces and optimally make use of the temporal evolution of the appearance of the face for recognition.