FACE COUNTER USING MATLAB

A project report submitted

in partial fulfilment of the requirement for the award of the degree of

Bachelor of Technology (Honors)

in

Electronics and Communication Engineering

By

KARUTURI SURYA SUMANTH (EL112006)

JAYANTI VENKATA SAI KIRAN (EL112051)

PEDDINTI NAGA BABU (EL112338)

Under the supervision of

Amit Prakash

Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

JAMSHEDPUR – 831014

November - 2015

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR

CERTIFICATE

This is to certify that this project report "**Face counter using Matlab**" is a bonafide record of work done by

KARUTURI SURYA SUMANTH (EL112006)

JAYANTI VENKATA SAI KIRAN (EL112051)

PEDDINTI NAGA BABU (EL112338)

Submitted in partial fulfilment for the award of degree of Bachelor of Technology (Honors) in the department of Electronics and Communication Engineering of National Institute of Technology, Jamshedpur during the year 2012-2016.

Prof. Amit Prakash   Dr. S. N. Singh

(Prof & Supervisor)   (Prof & Head)

Department of ECE   Department of ECE

Project viva-voice held on _____

CANDIDATE' DECLARATION

We hereby declare that the work which is being presented and the project entitled "**Face counter using Matlab**" in partial fulfilment of the requirements for the reward of the degree of B**achelor of Technology** in **Electronics and Communication Engineering** and submitted in the department of Electronics and Communication Engineering , NIT Jamshedpur is an authentic record of our own work carried under the esteemed supervision of **Amit Prakash, Professor**, Department of Electronics and Communication Engineering , NIT Jamshedpur. The matter presented in this project has not been submitted by us for the award of any degree or diploma to other universities.

K SURYA SUMANTH   **J V SAI KIRAN**   **P NAGA BABU**

 (EL112006)   **(EL112051)**   **(EL112338)**

This is to certify that above statement made by the candidates is true and correct to the best of our knowledge and belief. The work has been carried out by Karuturi Surya Sumanth, J V Sai Kiran and Peddinti Naga Babu under my guidance.

                                           Prof.Amit Prakash

                                           (Professor and Supervisor)

                                           Department of ECE

Place : NIT Jamshedpur

Date

ACKNOWLEDGEMENT

We would like to express our gratitude and sincere thanks to our honorable , esteemed supervisor **Prof.Amit Prakash**, Department of Electronics and Communication Engineering for his constant motivation and support during the course of our thesis. He is not only a great lecturer with great vision, but also a kind person. We truly appreciate and value his esteemed guidance and encouragement from the beginning to the end of this project.

We would also like to thank our **Prof. Dr. S.N Singh**, Head of Department of Electronics and Communication Engineering for providing facilities and infrastructure in the department.

We also thank all our faculty members for providing a solid background for our studies and research. They all have been great sources of inspiration for all of us.

Our full dedication to the work would not have been possible without the support of our parents and all our friends. Last but not the least , we also would like to express our gratitude to the almighty for he is the supreme source of all knowledge and wisdom in this world.

Regards

KARUTURI SURYA SUMANTH (EL112006)

JAYANTI VENKATA SAI KIRAN (EL112051)

PEDDINTI NAGA BABU (EL112338)

TABLE OF CONTENTS

Abstract

Object detection and tracking are important in many computer vision applications including activity recognition, automotive safety and surveillance. 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.

In this project an object detection system that will detect faces and count the number of persons that appear in a live video that is being captured by the Webcam. This real-time face detection program is developed using Matlab. A GUI interface allows us to start the camera and to count the number of faces in that video and to stop the camera. It can run in Matlab or as a stand-alone application.

During the Execution process

· A Matlab code is prepared and is run in a computer with webcam.

· When we run the code in MATLAB, A GUI interface will be opened.

· Video is recorded using the webcam.

· While recording, program counts the number of faces in front of webcam at

That instant.

· Count will be appeared in the GUI interface.

Following are the practical applications of the face counter .

Ø Used in surveillance.

Ø Used in security systems.

Ø Used in statistical analysis of digital image.

CHAPTER 1: INTRODUCTION

1.1 Introduction to Image Processing

Image Processing is a technique to enhance raw images received from cameras/sensors placed on space probes, aircrafts and satellites or pictures taken in normal day-today life for various applications. An Image is rectangular graphical object. Image processing involves issues related to image representation, compression techniques and various complex operations, which can be carried out on the image data. The operations that come under image processing are image enhancement operations such as sharpening, blurring, brightening, edge enhancement etc. Image processing is any form of signal processing for which the input is an image, such as photographs or frames of video; the output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Image processing usually refers to digital image processing, but optical and analog image processing are also possible.

Many techniques have been developed in Image Processing during the last four to five decades. Most of the methods are developed for enhancing images obtained from unmanned space probes, space crafts and military reconnaissance flights. Image Processing systems are becoming widely popular due to easy availability of powerful personnel computers, large memory devices, graphics softwares and many more.

Image processing involves issues related to image representation, compression techniques and various complex operations, which can be carried out on the image data. The operations that come under image processing are image enhancement operations such as sharpening, blurring, brightening, edge enhancement.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools.
- Analysing and manipulating the image.

- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

1.2 Face Detection and tracking

Object detection and tracking are important in many computer vision applications, including activity recognition, automotive safety and surveillance. Presented here is an object detection 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.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

1.3 Literature Survey:

- The most common technique for counting crowds at protests and rallies is Jacobs's Method, named for its inventor, Herbert Jacobs. Jacobs's method involves dividing the area occupied by a crowd into sections, determining an average number of people in each section, and multiplying by the number of sections occupied. But this technique proved to be very slow and less accurate.
- Even in the ATMs, there is no proper surveillance for checking if any other person is watching your password. The CC camera present at the ATM premises captures and records the video that can be used for investigation but it doesn't provide any live warnings or alerts.
- For Image Processing in a live video, there are many Algorithms available. But the Viola- jones algorithm is the most commonly used algorithm for face detection.
- This algorithm using a cascade of "weak-classifiers", using simple Haar features, can yield impressive results after excessive training.
- The same algorithm can be trained for the detection of other types of objects. (Example: cars, hands).
- This algorithm will compute the result extremely fast .

1.4 Objective :

With the implementation of this project , we aim at providing more efficient and faster face counting techniques to analyze the number of people in a gathering . Also for improving the security and surveillance system of the ATMs and other online transactions with the help of image processing.

CHAPTER 2: DETAILED DESCRIPTION OF OUR PROJECT

Theory

In this project an object detection system that will detect faces and count the number of persons that appear in a live video that is being captured by the Webcam. This real-time face counter program is developed using MATLAB version R2013a. A graphic user interface (GUI) allows users to perform tasks interactively through controls like Start Camera, Count Face and Stop. You can easily create a GUI and run it in MATLAB or as a stand-alone application.

There are different types of algorithms used in face detection. Here, we have used Viola-Jones algorithm for face detection with MATLAB program.
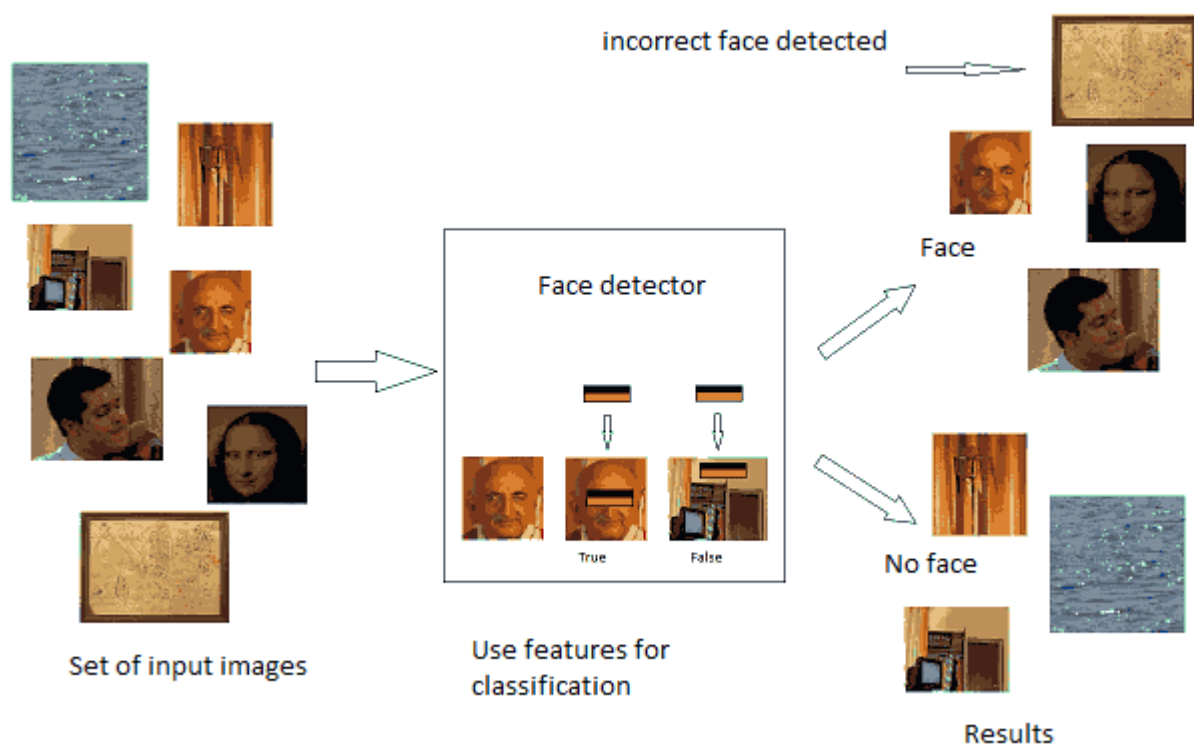
2.1 Viola-Jones algorithm

we decipher the Viola-Jones algorithm, the first ever real-time face detection system. There are three ingredients working in concert to enable a fast and accurate detection: the integral image for feature computation, Adaboost for feature selection and an attentional cascade for efficient computational resource allocation. Here we propose a complete algorithmic description, a learning code and a learned face detector that can be applied to any color image. Since the Viola-Jones algorithm typically gives multiple detections, a post-processing step is also proposed to reduce detection redundancy using a robustness argument.

A face detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. One natural framework for considering this problem is that of binary classification, in which a classifier is constructed to minimize the misclassification risk. Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance. This task requires an accurate numerical description of what sets human faces apart from other objects. It turns out that these characteristics can be extracted with a remarkable committee learning algorithm called Adaboost, which

relies on a committee of weak classifiers to form a strong one through a voting mechanism. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms. An operational algorithm must also work with a reasonable computational budget. Techniques such as integral image and attentional cascade make the Viola-Jones algorithm highly efficient: fed with a real time image sequence generated from a standard webcam, it performs well on a standard PC.

The problem to be solved is detection of faces in an image. A human can do this easily, but a computer needs precise instructions and constraints. To make the task more manageable, Viola–Jones requires full view frontal upright faces. Thus in order to be detected, the entire face must point towards the camera and should not be tilted to either side. While it seems these constraints could diminish the algorithm's utility somewhat, because the detection step is most often followed by a recognition step, in practice these limits on pose are quite acceptable.

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

- Robust – very high detection rate (true-positive rate) & very low false-positive rate always.
- Real time – For practical applications at least 2 frames per second must be processed.
- Face detection only (not recognition) - The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

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

In the Detecting features stage this algorithm has four Stages:

1. Hair Feature Selection
2. Creating an Integral Image
3. Adaboost Training
4. Cascading Classifiers

Matlab code for using the CascadeObjectDetector() function on pictures :

```
function [ ] = Viola_Jones_img( Img )
%Viola_Jones_img( Img )
% Img - input image
% Example how to call function: Viola_Jones_img(imread('name_of_the_picture.jpg'))

faceDetector = vision.CascadeObjectDetector;
bboxes = step(faceDetector, Img);
figure, imshow(Img), title('Detected faces');hold on
```

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

```
for i=1:size(bboxes,1)
   rectangle('Position',bboxes(i,:),'LineWidth',2,'EdgeColor','y');
end
end
```


Detected Face using the
cascadeObjectDetector function in MATLAB.
This function uses the Viola–Jones
algorithm to detect faces
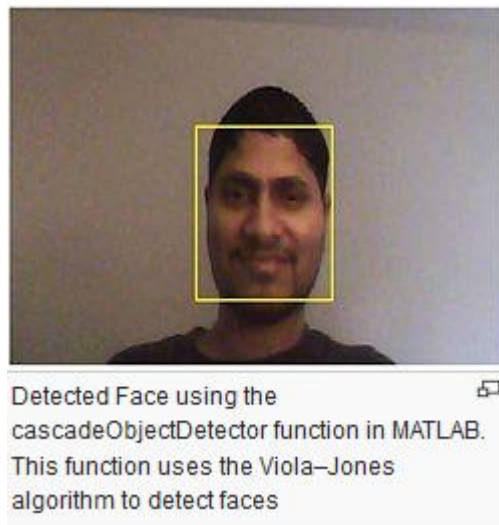
2.2 Advantages of Viola-jones Algorithm:-

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

2.3 Disadvantages of Viola jones Algorithm:-

- Detector is most effective only on frontal images of faces
- It can hardly cope with 45° face rotation both around the vertical and horizontal axis.
- Sensitive to lighting conditions
- We might get multiple detections of the same face, due to overlapping sub-windows.

CHAPTER 3 : INTRODUCTION TO MATLAB

The name MATLAB stands for MATRIX LABORATORY. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.
MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

There are various tools in Matlab that can be utilized for image processing, such as Simulink, GUI etc. Simulink contains various toolboxes and image processing toolbox is one such example. Simulink is used for simulation of various projects. GUI is another important tool in Matlab. It can be designed either by manual programming which is tedious task or by using guide. GUI is explained in next section.

3.1 GUI :

A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components, which enable a user to perform interactive tasks. The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed. GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders—just to name a few. GUIs created using MATLAB tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots. The following figure illustrates a simple GUI that you can easily build

The GUI contains
•An axes component
•A pop-up menu listing three data sets that correspond to MATLAB
Functions: peaks, membrane, and sinc
•Astatic text component to label the pop-up menu
•Three buttons that provide different kinds of plots: surface, mesh, and contour

When you click a push button, the axes component displays the selected data set using the specified type of 3-D plot.

Typically, GUIs wait for an end user to manipulate a control, and then respond to each user action in turn. Each control, and the GUI itself, has one or more call-backs, named for the fact that they "call back" to MATLAB to ask it to do things. A particular user action, such as pressing a screen button, or passing the cursor over a component, triggers the execution of each call back. The GUI then responds to these events. You, as the GUI creator, write call-backs that define what the components do to handle events. This kind of programming is often referred to as event-driven programming. In event-driven programming, call back execution is asynchronous, that is, events external to the software trigger call back execution. In the case of MATLAB GUIs, most events are user interactions with the GUI, but the GUI can respond to other kinds of events as well, for example, the creation of a file or connecting a device to the computer.

You can code call-backs in two distinct ways:

•As MATLAB language functions stored in files
•As strings containing MATLAB expressions or commands (such as 'c = sqrt(a*a + b*b); 'or' print')Using functions stored in code files as call-backs is preferable to using strings, because functions have access to arguments and are more powerful and flexible. You cannot use MATLAB scripts (sequences of statements stored in code files that do not define functions) as call-backs. Although you can provide a call back with certain data and make it do anything you want, you cannot control when call-backs execute. That is, when your GUI is being used, you have no control over the sequence of events that trigger particular call-backs or what other call-backs might still be running at those times. This distinguishes event-driven programming from other types of control flow, for example, processing sequential data files.

A MATLAB GUI is a figure window to which you add user-operated components. You can select, size, and position these components as you like using call-backs you can make the components do what you want when the user clicks or manipulates the components with keystrokes.

You can build MATLAB GUIs in two ways:

•Use GUIDE (GUI Development Environment), an interactive GUI construction kit. This approach starts with a figure that you populate with components from within a graphic layout editor. GUIDE creates an associated code file containing call-backs for the GUI and its components. GUIDE saves both the figure (as a FIG-file) and the code file. Opening either one also opens the other to run the GUI.
•Create code files that generate GUIs as functions or scripts (programmatic GUI construction).

Using this approach, you create a code file that defines all component properties and behaviors. When a user executes the file, it creates a figure, populates it with components, and handles user interactions. Typically, the figure is not saved between sessions because the code in the file creates a new one each time it runs.

The code files of the two approaches look different. Programmatic GUI files are

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

generally longer, because they explicitly define every property of the figure and its controls, as well as the call-backs. GUIDE GUIs define most of the properties within the figure itself. They store the definitions in its FIG-file rather than in its code file. The code file contains call-backs and other functions that initialize GUI when it opens.
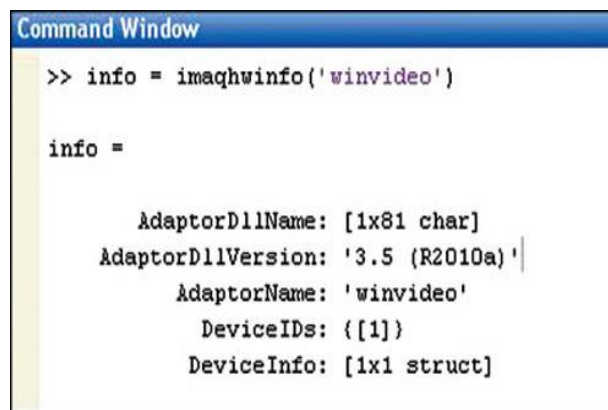
You can create a GUI with GUIDE and then modify it programmatically. However, you cannot create a GUI programmatically and then modify it with GUIDE. The GUI-building technique you choose depends on your experience, your preferences, and the kind of application you need the GUI to operate. This table outlines some possibilities.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.
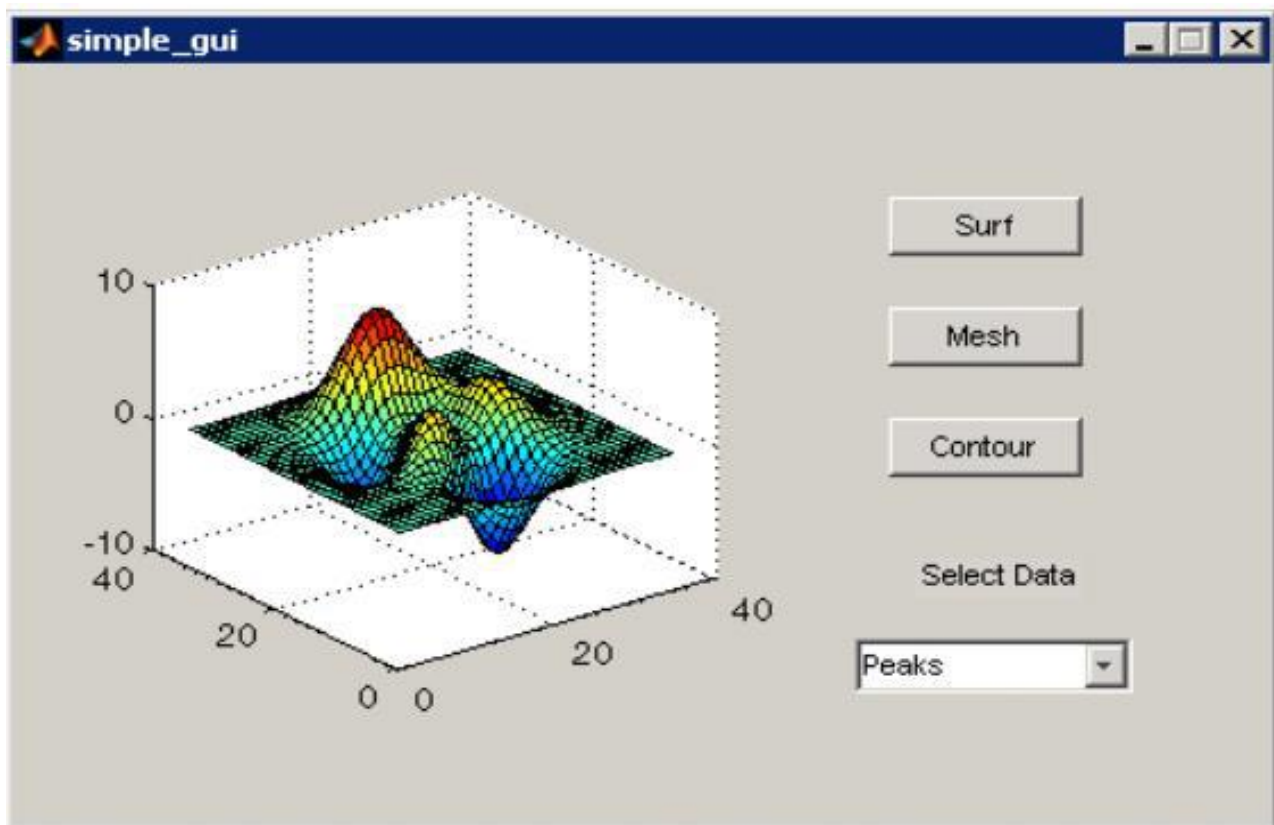
CHAPTER 4: IMPLEMENTATION AND RESULTS

4.1 Implementation of Algorithm :-

The program (testing.m) has many functions. Do not edit the functions as these are linkers and non-executable codes. First, you have to find the format supported by the camera and its device ID using the command given below (also shown in Fig. 2):

Info = imaqhwinfo('winvideo')

After finding the device ID, you can change the device ID number in your source code.
The device ID here is {1}, so we have written '1' in the code, as mentioned below:

Vid = videoinput('winvideo',1,'YUY2_640x840');

We also have other formats in MATLAB. You can check which format your camera supports by using the commands below (also shown in Fig. 3):
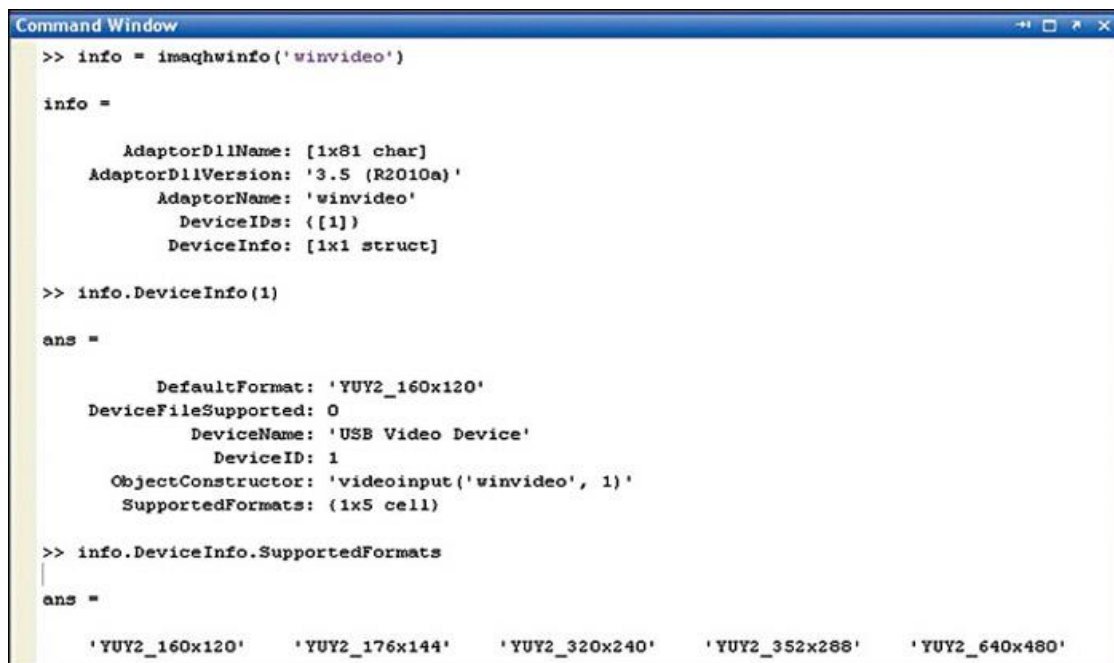
Info.DeviceInfo(1)

Info.DeviceInfo.SupportedFormats

  In Fig. 3, you can see that format 'YUY2_160x120' is the one supported by thecamera by default. But, there are other formats (resolutions) that your camera

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

can support, as shown in the last line of this screenshot. If you select a different format and device number, you should make changes in the source code accordingly. To detect a face or a particular feature on the faces of people, use the following steps in MATLAB program (testing.m):

1. Define and set-up your cascade object detector using the constructor:

detector = vision.CascadeObjectDetector;

It creates a system object detector that detects objects using Viola-Jones algorithm. Its classification model property controls the type of object to detect. By default, the detector is configured to det ect faces.

2. Call the step method with input image I, cascade object detector, points PTS and any other optional properties.
Below is the syntax for using the step method. Use the step syntax with input image I, selected cascade object detector and other optional properties to perform detection.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

BBOX =step(detector,I)

It returns BBOX, an M-by-4 matrix defining M-bounding boxes, containing detected objects. This method performs multi-scale object detection on input image I. Each row of output matrix BBOX contains a four-element vector (x, y, width and height) that specifies in pixels, the upper-left corner and size of a bounding box. Input image I must be a gray scale or true colour (RGB) image.

3. The third step is:

InsertObjectAnnotation(I,'rectangle',Position,Label)

It inserts rectangles and corresponding labels at the location indicated by the position matrix. The position input must be an M-by-4 matrix, where each row (M) specifies a rectangle as a four-element vector (x, y, width and height). Elements x and y indicate the upper-left corner of the rectangle, and the width and height specify the size.

X = sprintf('%d', no_rows);

This will get the number of rows (which will be equal to number of people)

set(handles.text2,'string',X);

This code will display the value of X in GUI.

4.2 GUIDATA :

guidata(object_handle,data)

data = guidata(object_handle)

guidata(object_handle,data) stores the variable data with the object specified by object_handle. If object_handle is not a figure, then the object's parent figure is used. data can be any MATLAB variable, but is typically a structure, which enables you to add new fields as required.

guidata can manage only one variable at any time. Subsequent calls to guidata(object_handle,data) overwrite the previously stored data.

To change the data managed by guidata:

1. Get a copy of the data with the command data = guidata(object_handle).
2. Make the desired changes to data.
3. Save the changed version of data with the command guidata(object_handle,data).

guidata provides application developers with a convenient interface to a figure's application data:

4.3 Using guidata in GUIDE :-

Notice that you use the input argument hObject in place of gcbo to refer to the object whose callback is executing.

If you use GUIDE, you do not need to call guihandles to create a structure. You can add your own data to the handles structure. For example, this code adds the field, numberOfErrors, to the structure:

Suppose you needed to access the numberOfErrors field in a push button callback. Your callback code now looks something like this:

```
% --- Executes on button press in pushbutton1.

function my_GUIDE_GUI_pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% ...

% No need to call guidata to obtain a structure;
% it is provided by GUIDE via the handles argument
handles.numberOfErrors = handles.numberOfErrors + 1;
% save the changes to the structure
guidata(hObject,handles)
```

Like these for each button in the GUI the Call back functions get called and get executed and update the hObject.

Hence by using the GUIDE , the algorithm works and the count will be updated real-time.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

4.4 Testing and output :

To test this program, follow the steps given below:
1. Check the device ID and write the device ID number in the source code.
2. Run the program. A GUI will appear, as shown in Figure.
3. Click on Start Camera button to initialize camera settings.
4. Next, click on count face button and the camera will count the faces. If the number of faces increase , the count will automatically be displayed on the GUI.
5. To stop, click Stop button.
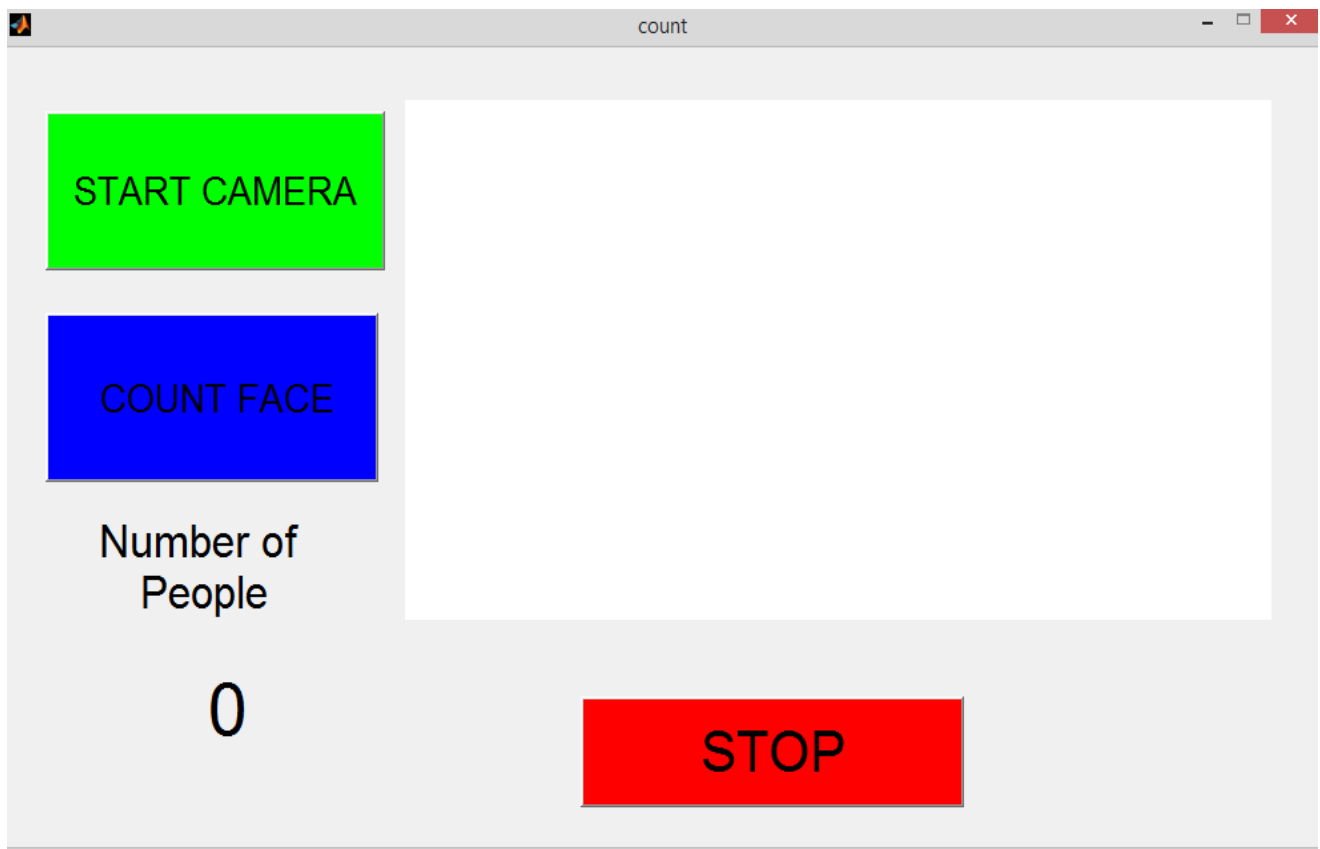
4.5 RESULTS :

The output of GUI given below clearly indicates the implemented results of the algorithm designed.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.
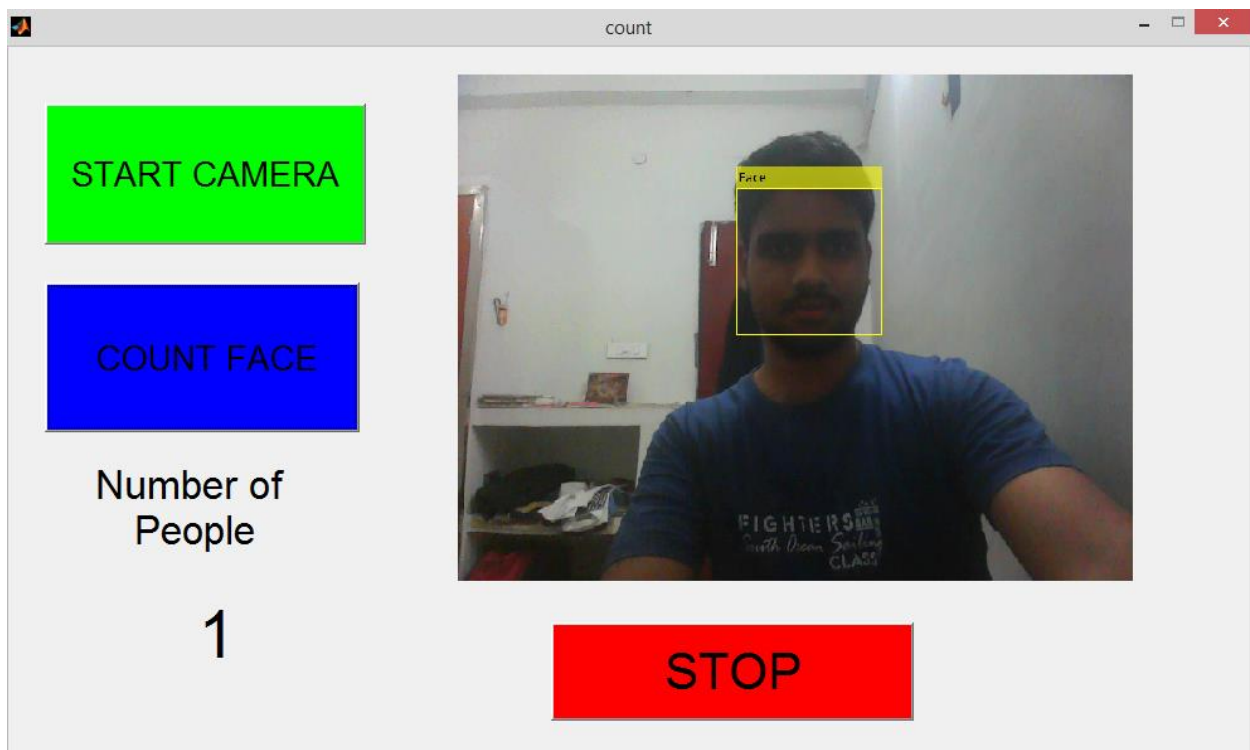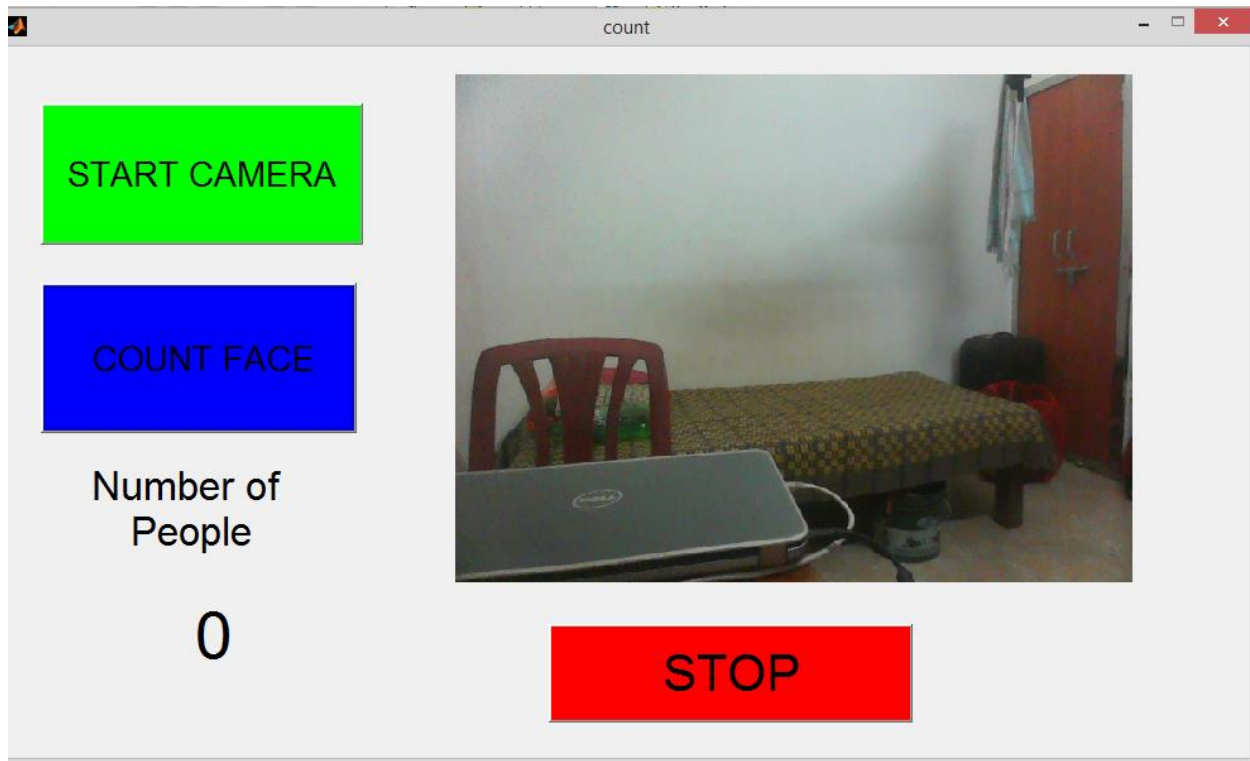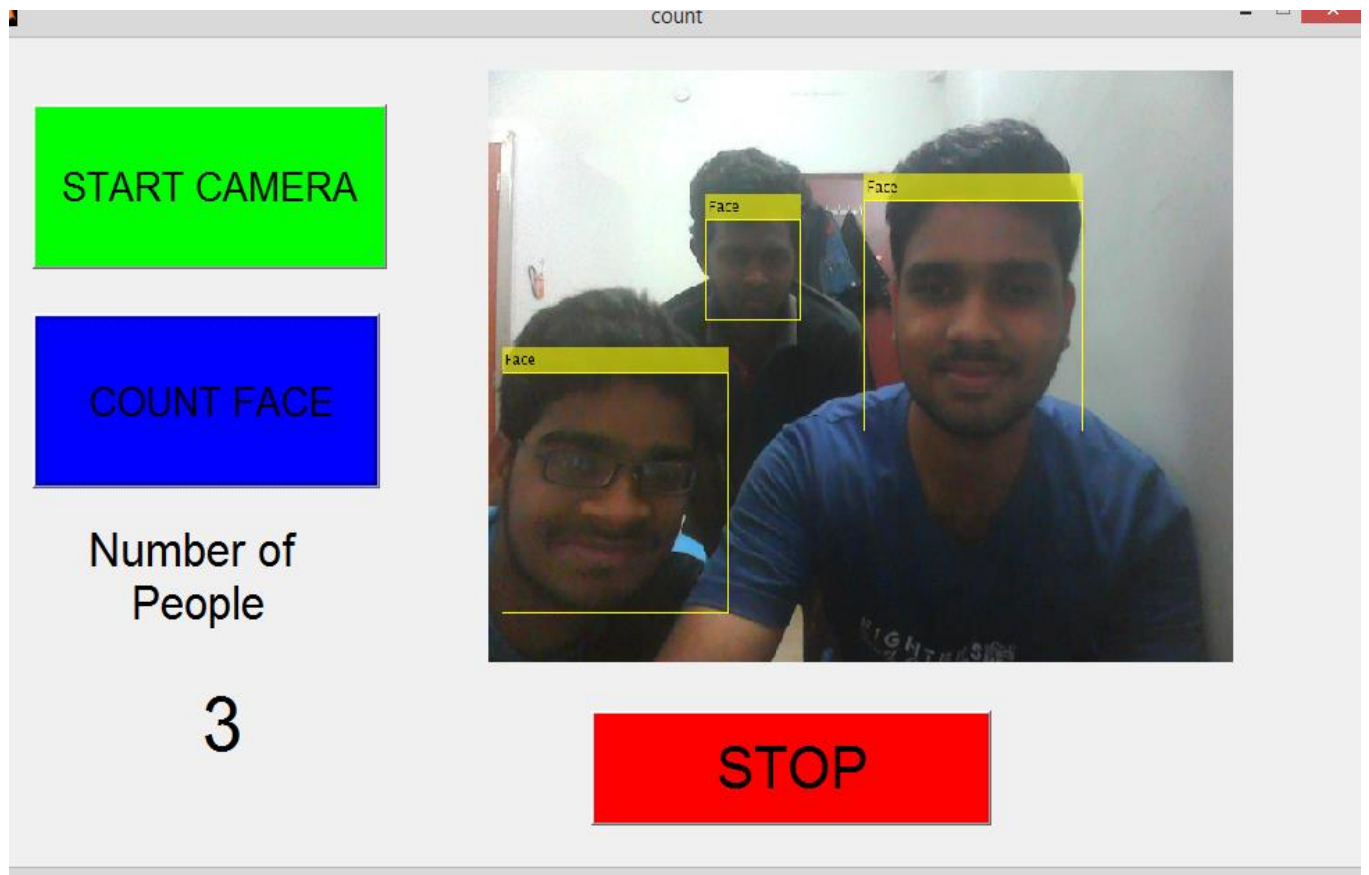
CHAPTER 5: APPLICATIONS AND FUTURE SCOPE

5.1 Applications:

The Face counter is a very useful technique in number of practical applications. In a crowd number calculations, earlier mathematical techniques were used. But with the help of face counter, we can easily estimate the crowd number with a better accuracy and precision. This technique is very fast and produces live numbers with help of the video captured. In previously used techniques, the crowd count was calculated by multiplying the average crowd per unit area and the area occupied by the crowd. This would be useful in calculating crowd in ticketless occasions and in the public gatherings.

In the ATMs nowadays, the surveillance systems used are the general cc cameras that capture the video and no further purpose. With the help of face counter, we can know the number people inside the ATM premises and further issue a warning if the count exceeds the limit.

Also during the password entry time in the secure transactions, with help of this technique we can warn the user by issuing an alert, if anyone is peeking from behind. The same can be utilized in ATM pin entry time also.

Coming to small scale applications, we can verify the attendance of the students in the class with this technique. Only requirement is an integrated camera connected to a display device.

5.2 Conclusion

"Face counter using image processing" technique that we propose overcomes all the limitations of the earlier (in use) techniques used for crowd analysis and security systems. Earlier, the number of people in the crowd was calculated using traditional mathematical techniques, that were more time consuming and less accurate. With the help of this face counter technique, we can overcome this problem. Also the application of this project, includes the security and surveillance during the online transactions and in the ATMs, proving a better security solutions. The major advantage of this project is that it is much faster technique in crowd calculation and more efficient in security systems. The accuracy in this system depends upon the quality of the camera used. Better cameras use will provide more accurate results.

5.3 Future work

By implementing this project using better cameras, we can provide better security and also accurate crowd analysis. We can also limit the number of people entering the ATM without any manual work. On introducing this technique in ATMs and online transactions during the password entry time, the transactions would become more secure and safer than before.

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

CHAPTER 6: REFERENCES

6.1: References:

- Digital image processing by Rafael C. Gonzalez and Richard E. Woods.

- www.mathworks.in   Web.

- http://www.electronicsforyou.com Web.

- K.Sung and T.Poggio. "Example-based learning for view-based face detection". In IEEE Patt.Anal.Mach.Intell., volume 20, pages 39-51, 1998.

- C. Papageorgiou, M. Oren and T. Poggio, "A general framework for object detection.", In International Conference on Computer Vision, 1998

- P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 1:511, 2001.

- Implementing Viola–Jones

- Video lecture on Viola–Jones algorithm

- H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In International Conference on Computer Vision, 2000.

- Rapid object detection using a boosted cascade of simple features

- Kwok-wai wong, kin-man lam, wan-chi siu (2001). "an efficient algorithm for human face detection and facial feature extraction under different conditions".

- Tesco face detection sparks needless surveillance panic, Facebook fails with teens, doubts over Google+ | Technology | theguardian.com

APPENDIX

Matlab Program

```matlab
function varargout = count(varargin)
% COUNT MATLAB code for count.fig
%      COUNT, by itself, creates a new COUNT or raises the existing
%      singleton*.
%
%      H = COUNT returns the handle to a new COUNT or the handle to
%      the existing singleton*.
%
%      COUNT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in COUNT.M with the given input arguments.
%
%      COUNT('Property','Value',...) creates a new COUNT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before count_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to count_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 count

% Last Modified by GUIDE v2.5 31-Aug-2013 16:48:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @count_OpeningFcn, ...
                   'gui_OutputFcn',  @count_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

```matlab
            '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 count is made visible.
function count_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 count (see VARARGIN)

% Choose default command line output for count
handles.output = hObject;
axes(handles.axes1);
imshow('blank.jpg');
axis off;
set(handles.text2,'string','0');

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
```

```matlab
function varargout = count_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 start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global vid     % making the variable global
vid = videoinput('winvideo' , 1, 'YUY2_640X480');
% Create a video input with YUY2 format and 640X480 resolution


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

global vid

% Set the parameters for video
triggerconfig( vid ,'manual');           % the trigger occurs only after the trigger
function
set(vid, 'FramesPerTrigger',1);          % one frame acquired per trigger
set(vid, 'TriggerRepeat',Inf);
% Keep executing the trigger every time the trigger condition is met until the stop
function is called
set(vid,'ReturnedColorSpace','rgb');     % to get the rgb colour
image
```

NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR.

```matlab
vid.Timeout = 10;
start(vid);

while (1)
facedetector = vision.CascadeObjectDetector;              % Create a
cascade detector object
trigger(vid);                                %trigger to get the frame from the
video
image = getdata(vid);                        %store that frame in 'image'
bbox = step(facedetector, image);
% position of face in 'bbox' (x, y, width and height)
insert_object = insertObjectAnnotation(image,'rectangle',bbox,'Face');
 % Draw the bounding box around the detected face.
imshow(insert_object);
axis off;                                % invisible the axis from GUI
no_rows = size(bbox,1);
 % get the number of rows (which will be equal to number of people)
X = sprintf('%d', no_rows);
set(handles.text2,'string',X);
 %display the value of X in GUI
end




% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global vid
stop(vid),clear vid %stop the running video
```