**PROJECT REPORT ON**

**Implementation Of Image Processing On CCTV**



**AMBEDKAR INSTITUTE OF ADVANCED COMMUNICATION TECHNOLOGY & RESEARCH**

**Submitted by Under the Guidance of**

Mayank (13110102814) Mr.Gangadharppa  
AyushSaini (11410102814)  
Vijay RatnaPrabhakar (12410102814)  
Amit (10810102814)

**ACKNOWLEDGEMENT**

I express my sincere gratitude to Mr.Gangadharppa , for his valuable guidance and timely suggestions during the entire duration of my dissertation work, without which this work would not havebeen possible. I would also like to convey my deep regards to all other faculty members, who have bestowed their great effort and guidance at appropriate times withoutwhich it would have been very difficult on my part to finish this work. Finally I would also like to thank my friends for their advice and pointing out my mistakes.

**ABSTRACT**

The current state of a cctv camera is that it is used just as video capturing device. But with the right modifications and technology we can convert the basic idea of a cctv camera.Our project is based on implementation of image processing on cctv cameras. It will help to increase the current capabilities of a cctv camera.

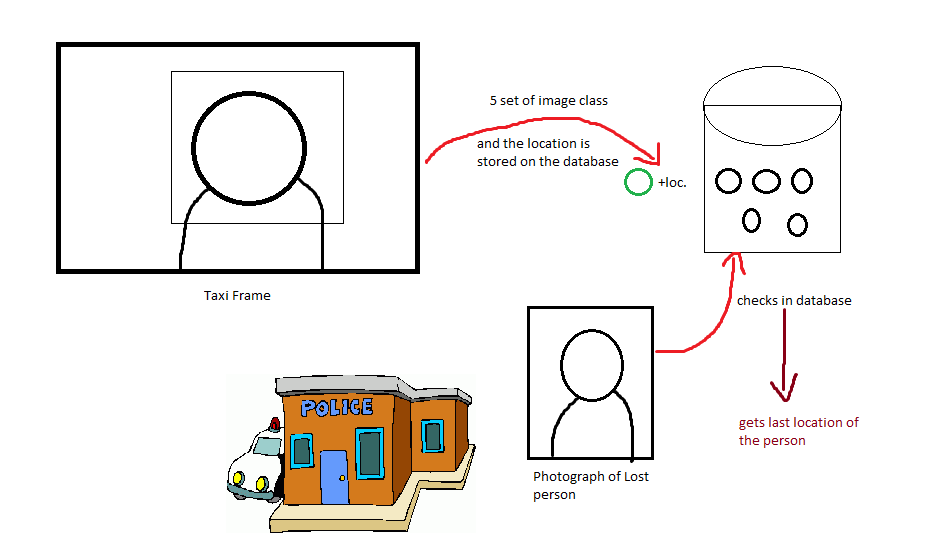
A CCTV camera based on implementation of image processing is a camera that can process, compress and store digtal images, and zoom to a captures image using a digital image-capturing component therein. It can also restore the captured image to either an analog or a digital signal for transmission or relay of captured images immediately or at a later time. Moreover, the camera can also perform motion image analysis and tracking and can also perform optical or digital image zooming on tracked image to monitor an object and ensure a clear image is captured.

The automatic detection and tracking of a photographed object of the present project is accomplished by analiysing the difference between previous and their subsequent images. During the process of photographing, the camera transmits image data to a monitor centre and also compares the image being photographed with the previously stored digital image.

CONTENTS

1. Overview
2. Detection of Face
   1. Viola Jones
   2. Feature detection
   3. RANSAC
   4. Category Detection
   5. SVM
3. Face Detected(Code)
4. Further Code Overview
   1. Designing Face Recogisation System
5. Feature Extraction Function Code
6. Final Storage Code
7. Detection Code for Police.
8. Future Developments

**Overview**

In this project of implementation of Image processing (Computer Vision) on CCTV footage we are mainly focusing on the camera inside the taxies.  
So, there was an initial project various features to make a car or viehcle secure like alarm system, GPS tracking etc. So, what we propose with this project is that how about govt. offers every Taxi this level of security and return put up a CCTV cam inside each taxi. For security and surviellence purposes.  
 So, the CCTV part of the whole project is done in this project.

So, Consider a case that a person in sitting in that smart Taxi. The sytem takes 5 photos and makes a class of photographs.  
Now, this person never reaches home. His Mom takes his photo to the nearest police station to make a FIR.  
Now what the police officials can do is scan that photo and scan the whole database if the photo matches the police official will know all the details including the last location, No. plate of the Taxi, Taxi person name and address etc.

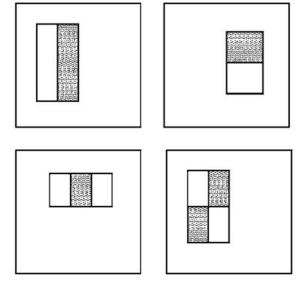
**So, this project is divided into several parts:**-Detection of face from CCTV  
-Storage of the face from CCTV on database  
-Assigning the face the same class.  
- Making the Police end code for the recognisation of the face.  
-Comparing both the faces.  
-Displaying the class of the image.

**Detection of the face:**In this project we use the Viola Jones algorithm for the face detection

**Viola Jones**

Three main ideas:

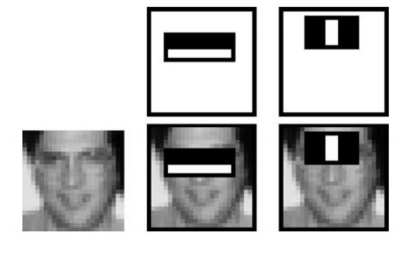
• introduction of a new image representation called the Integral Image  
 • simple and efficient classifier which is built using the AdaBoost learning algorithm  
 • method for combining classifiers in a “cascade” which allows background regions of the image to be quickly discarded Operating on 384x288 pixel images, faces are detected at 15 frames per second on a 700 MHz Intel Pentium III

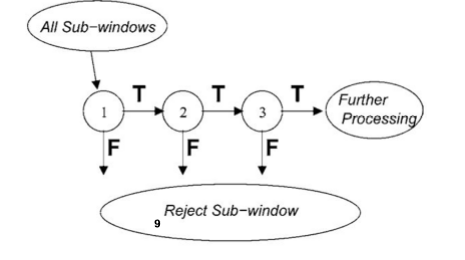
**Features**

The simple features used are reminiscent of Haar basis functions

• two-rectangle feature: difference between the sum of the pixels within two rectangular regions  
 • three-rectangle feature: sum within two outside rectangles subtracted from the sum in a center rectangle   
• four-rectangle feature: difference between diagonal pairs of rectangles.

Given that the base resolution of the detector is 24×24, the exhaustive set of features is quite large, 160,000 • a very efficient way to compute them is needed -> integral image  
 • only the useful ones have to be used -> AdaBoost

** The first two features** selected by AdaBoost for the task of face detection are easily interpreted:   
• first feature: the region of the eyes is often darker than the region of the nose and cheeks  
 • second feature: the eyes are darker than the bridge of the nose. So, the systems is essentially based on correlation!

**Cascaded classifier**  
 Smaller, and therefore more efficient, classifiers can be constructed which reject many of the negative subwindows while detecting almost all positive instances:   
• simpler classifiers are used to reject the majority of subwindows   
• then, more complex classifiers are called upon to achieve low false positive rates.

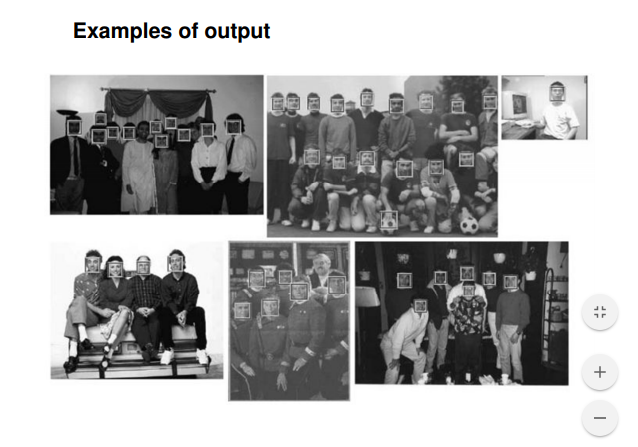
T: true F: false

**The cascaded classifier for face detection**  
 … has 38 layers and 6060 features:  
 • first classifier: two features, rejects about 50% of non-faces (while correctly detecting close to 100% of faces)   
• second classifier: ten features, rejects 80% of non-faces   
• 3rd and 4th: 25 features

**Training**   
• Base resolution: 24x24 pixels   
• several thousands faces and non-faces   
• training time: weeks on a 466 MHz AlphaStation XP900

**Position and scale invariance**

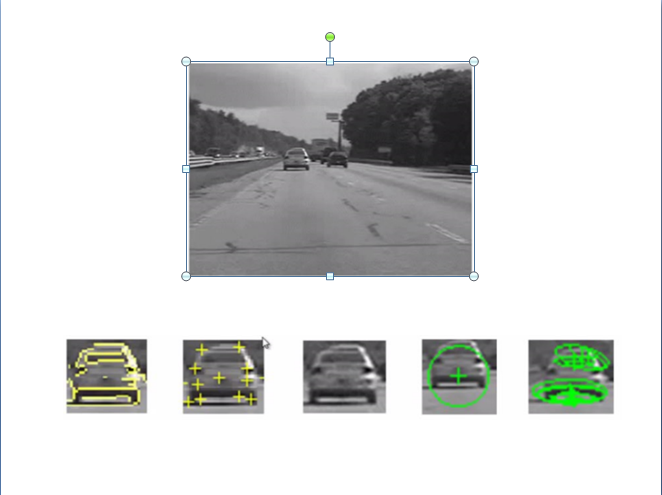
The final detector is scanned across the image at multiple scales and locations:  
 • Scaling is achieved by scaling the detector itself, rather than scaling the image (features can be evaluated at any scale with the same cost)   
• The detector is also scanned across location, by shifting the window some number of pixels

**Preprocessing**   
• Sub-windows were variance normalized during training to minimize the effect of different lighting conditions.   
• The same is done during detection as well. Final postprocessing   
• It is useful to postprocess the detected sub-windows in order to combine overlapping detections into a single detection.

**Old School Object Detection**

* Segmentation and Blob analysis
* Template Matchine

**Feature are Critical to Computer Vision**

* Edge
* Corner
* Template
* SURF
* MSER

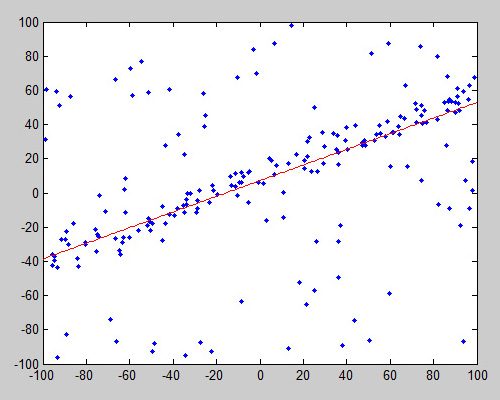
**RANSAC in GeometricTransformEstimator**

Random sample consensus, or RANSAC, is an iterative method for estimating a mathematical model from a data set that contains outliers. The RANSAC algorithm works by identifying the outliers in a data set and estimating the desired model using data that does not contain outliers.

RANSAC is accomplished with the following steps

1. Randomly selecting a subset of the data set
2. Fitting a model to the selected subset
3. Determining the number of outliers
4. Repeating steps 1-3 for a prescribed number of iterations

For example, the equation of a line that best fits a set of points can be estimated using RANSAC.

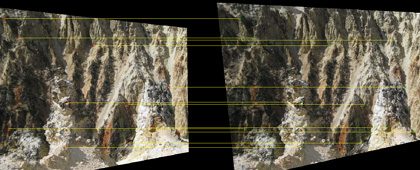


Data points shown in blue, with the line of form y = mx+c estimated using RANSAC indicated in red.

In computer vision, RANSAC is used as a robust approach to [estimate the fundamental matrix](https://in.mathworks.com/help/releases/R2014a/vision/ug/find-fundamental-matrix-describing-epipolar-geometry.html)in stereo vision, for finding the commonality between two sets of points for feature-based [object detection](https://in.mathworks.com/discovery/object-detection.html), and registering sequential video frames for [video stabilization](https://in.mathworks.com/help/vision/examples/video-stabilization-using-point-feature-matching.html).

[[](https://in.mathworks.com/content/mathworks/in/en/discovery/ransac/_jcr_content/mainParsys/image_1.adapt.full.high.jpg/1485858838393.jpg)](https://in.mathworks.com/content/mathworks/in/en/discovery/ransac/_jcr_content/mainParsys/image_1.adapt.full.high.jpg/1485858838393.jpg)

Frames of video stitched together to create a video mosaic. RANSAC is used to estimate the geometric transform between video frames (see [example](https://in.mathworks.com/help/vision/examples/video-mosaicking.html) for details).



Stereo rectification using feature point matching. RANSAC is used to estimate the fundamental matrix

**Category Detection** = Features + Machine learning

**Typical Features:**

* Histogram of oriented Gradients(HOG)
* Haar like features
* Local binary patterns

**Machine Learning**

* Requires input data and known reponses
* Build model to predict responses to a new data

**Typical Machine learning classifiers**

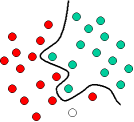
* SVM
* Adaboost
* Cascade of Adaboost
* K-nearest neighbor

**SVM**

upport Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the illustration below. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).



The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below. Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.



The illustration below shows the basic idea behind Support Vector Machines. Here we see the original objects (left side of the schematic) mapped, i.e., rearranged, using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as mapping (transformation). Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all we have to do is to find an optimal line that can separate the GREEN and the RED objects.

**Code for face detection**

faceDetector = vision.CascadeObjectDetector();

%Get the input device using image acquisition toolbox,resolution = 640x480 to improve performance

obj =imaq.VideoDevice('winvideo', 1, 'YUYV\_320x240','ROI', [1 1 320 240]);

set(obj,'ReturnedColorSpace', 'rgb');

figure('menubar','none','tag','webcam');

while (true)

frame=step(obj);

bbox=step(faceDetector,frame);

boxInserter = vision.ShapeInserter('BorderColor','Custom',...

'CustomBorderColor',[255 255 0]);

videoOut = step(boxInserter, frame,bbox);

imshow(videoOut,'border','tight');

f=findobj('tag','webcam');

if (isempty(f));

[hueChannel,~,~] = rgb2hsv(frame);

% Display the Hue Channel data and draw the bounding box around the face.

figure, imshow(hueChannel), title('Hue channel data');

rectangle('Position',bbox,'EdgeColor','r','LineWidth',1)

hold off

noseDetector = vision.CascadeObjectDetector('Nose');

faceImage = imcrop(frame,bbox);

imshow(faceImage)

noseBBox = step(noseDetector,faceImage);

noseBBox(1:1) = noseBBox(1:1) + bbox(1:1);

videoInfo = info(obj);

ROI=get(obj,'ROI');

VideoSize = [ROI(3) ROI(4)];

videoPlayer = vision.VideoPlayer('Position',[300 300 VideoSize+30]);

tracker = vision.HistogramBasedTracker;

initializeObject(tracker, hueChannel, bbox);

while (1)

% Extract the next video frame

frame = step(obj);

% RGB -> HSV

[hueChannel,~,~] = rgb2hsv(frame);

% Track using the Hue channel data

bbox = step(tracker, hueChannel);

% Insert a bounding box around the object being tracked

videoOut = step(boxInserter, frame, bbox);

%Insert text coordinates

% Display the annotated video frame using the video player object

step(videoPlayer, videoOut);

pause (.2)

end

% Release resources

release(obj);

release(videoPlayer);

close(gcf)

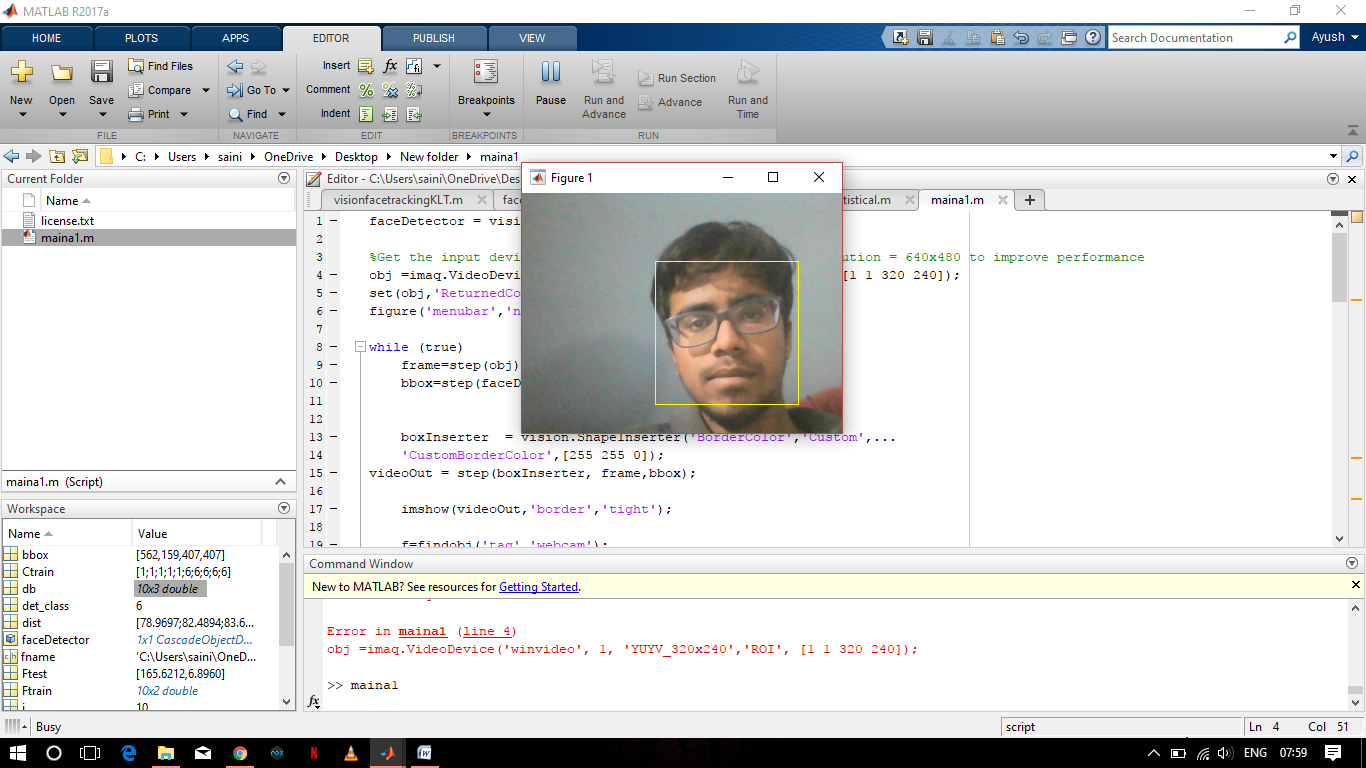
break

end

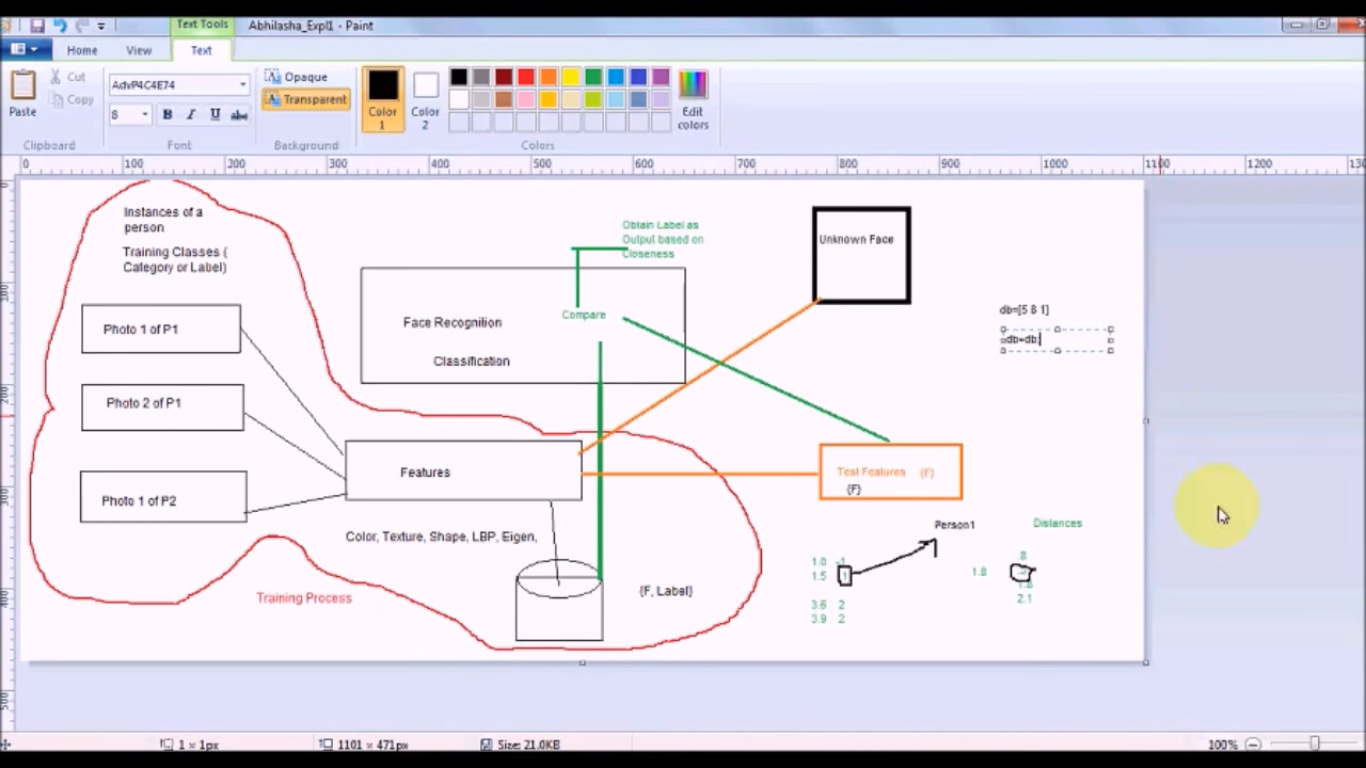
pause(0.05)

end

release(obj)

**Output:**

**Coding Overview:**

****

**Designing face Recogninsation System**

Here In the diagram we have a database of photo graphs of different people.

Different Photos of the same person can be classfied into a class and now all these photos can be called as part of this class.  
To recognise an unknown face if we subtract two photo graphs we might get many diiference even if they are of the same person.  
So, to remove this redundancy we use Features.  
Here in this case we assume that the mean and the standard deviation as the unique charactistic feature of a photograph.

In our case, we have to first extract the photo from the CCTV footage and store it.  
Then extract it features and store it on database.  
Now this feature function will be avilable to both the unknown face and the face extracted from the footage.

Features extracted will be stored on a database.  
Now for the facial recognisation system firstly the feature extraction function works on both the unknown face and the known face.  
Now these extracted features are compared while finding the distances between the features and finding the least amongst them and declaring the class with which it finds the least distance.

**Feature extraction code**

function [F]=FeatureStatistical(im)

im=double(im);

m=mean(mean(im));

s=std(std(im));

F=[m,s];

**Final Code for storage**

clear all;

faceDetector = vision.CascadeObjectDetector();

%Get the input device using image acquisition toolbox,resolution = 640x480 to improve performance

obj =imaq.VideoDevice('winvideo', 1, 'YUY2\_320x240','ROI', [1 1 320 240]);

set(obj,'ReturnedColorSpace', 'rgb');

figure('menubar','none','tag','webcam');

i=0;

k=1;

try

load db;

c=size(db,1);

c=c+1;

catch

disp("not yet");

end

while true

frame=step(obj);

bbox=step(faceDetector,frame);

i=i+1;

disp(i);

if i == 10 || i == 20 || i == 30 || i == 40 || i == 50

out = imcrop(frame,bbox);

imshow(out);

out1 = rgb2gray(out);

filename=sprintf('new%d.jpg',k)

imwrite(out1, filename);

im=imread(filename)

F=FeatureStatistical(im);

k=k+1;

try

load db;

F=[F c];

db=[db;F];

save db.mat db

catch

c=1;

db=[F c];

save db.mat db

end

end

boxInserter = vision.ShapeInserter('BorderColor','Custom',...

'CustomBorderColor',[255 255 0]);

videoOut = step(boxInserter, frame,bbox);

imshow(videoOut,'border','tight');

f=findobj('tag','webcam');

if (isempty(f));

[hueChannel,~,~] = rgb2hsv(frame);

% Display the Hue Channel data and draw the bounding box around the face.

figure, imshow(hueChannel), title('Hue channel data');

rectangle('Position',bbox,'EdgeColor','r','LineWidth',1)

hold off

noseDetector = vision.CascadeObjectDetector('Nose');

faceImage = imcrop(frame,bbox);

imshow(faceImage)

noseBBox = step(noseDetector,faceImage);

noseBBox(1:1) = noseBBox(1:1) + bbox(1:1);

videoInfo = info(obj);

ROI=get(obj,'ROI');

VideoSize = [ROI(3) ROI(4)];

videoPlayer = vision.VideoPlayer('Position',[300 300 VideoSize+30]);

tracker = vision.HistogramBasedTracker;

initializeObject(tracker, hueChannel, bbox);

while (1)

% Extract the next video frame

frame = step(obj);

% RGB -> HSV

[hueChannel,~,~] = rgb2hsv(frame);

% Track using the Hue channel data

bbox = step(tracker, hueChannel);

% Insert a bounding box around the object being tracked

videoOut = step(boxInserter, frame, bbox);

%Insert text coordinates

% Display the annotated video frame using the video player object

step(videoPlayer, videoOut);

pause (.2)

end

% Release resources

release(obj);

release(videoPlayer);

close(gcf)

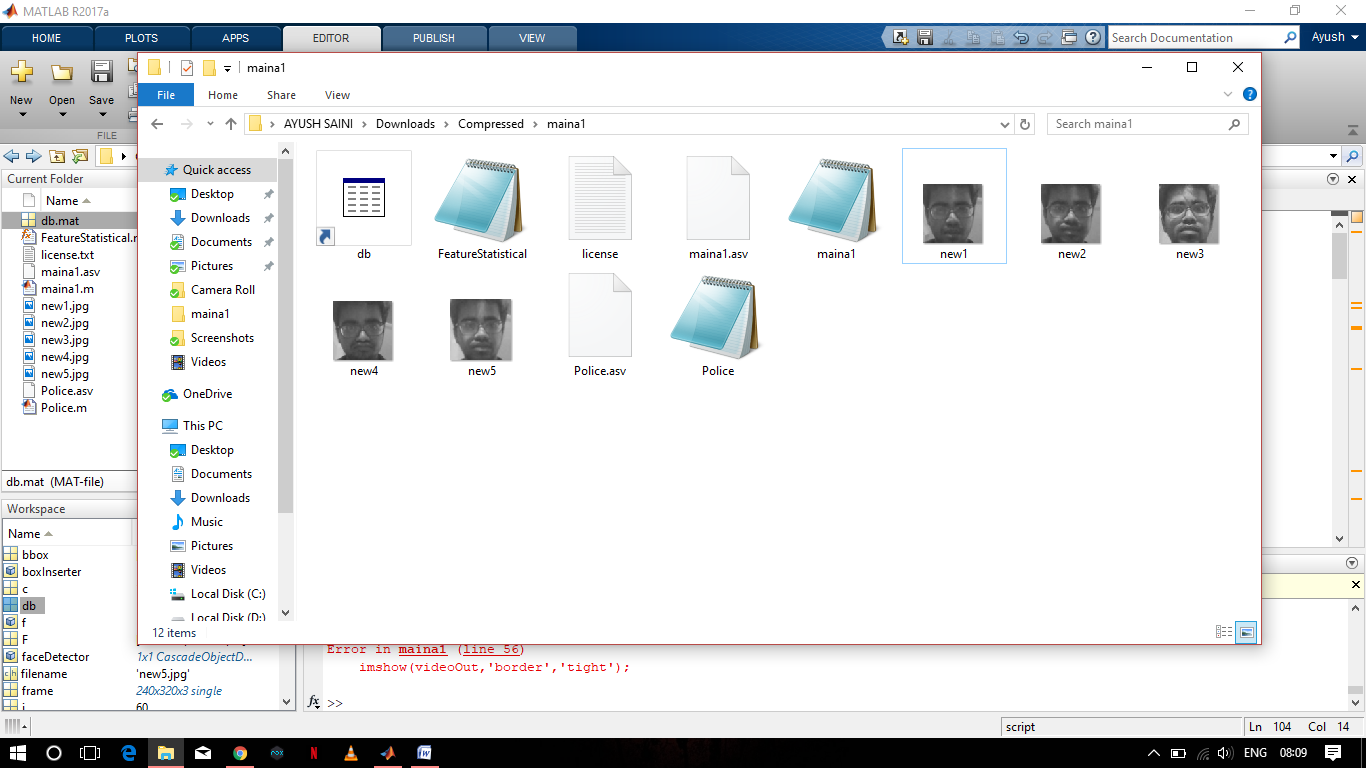
break

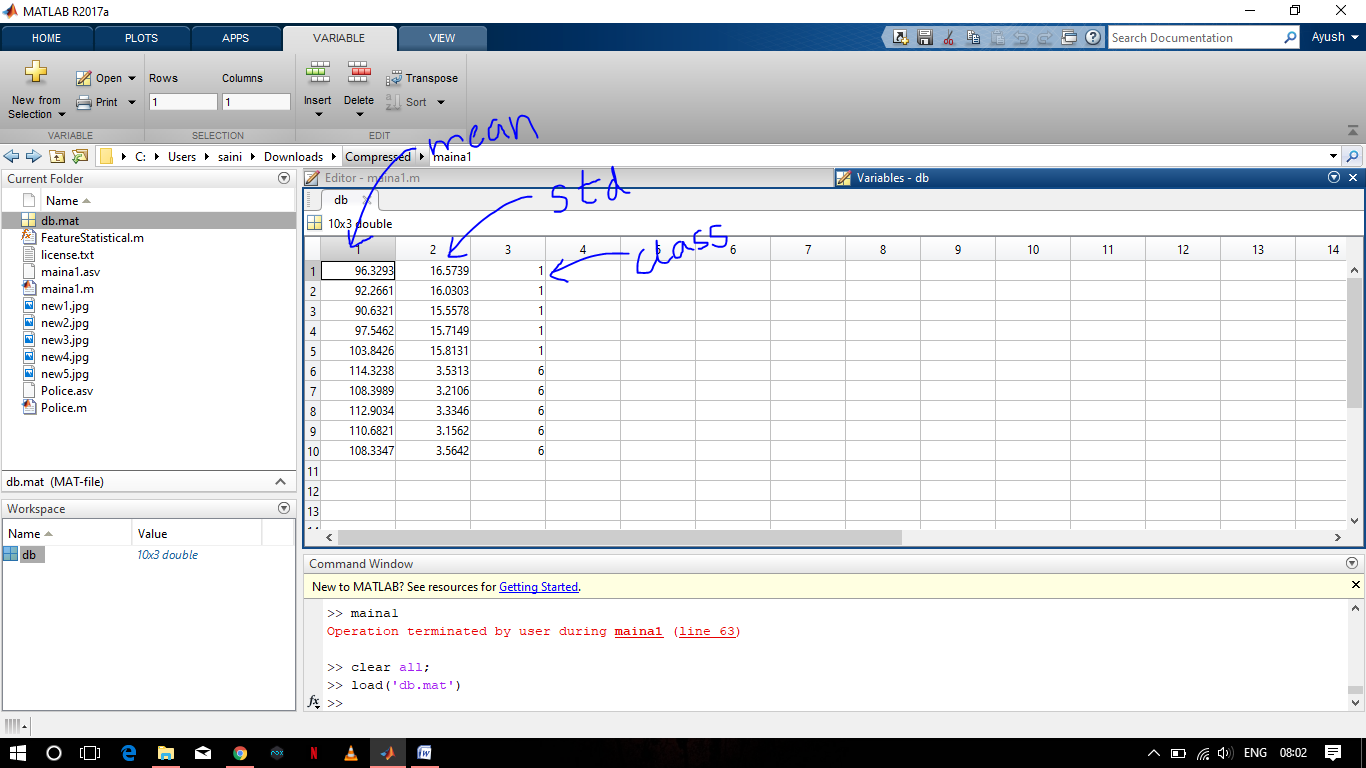
end

pause(0.05)

end

release(obj)

****

****

**Police Detection Code**

clc;

close all;

faceDetector = vision.CascadeObjectDetector();

[fname path]=uigetfile('.jpg','Enter Lost person photo');

fname=strcat(path,fname);

im=imread(fname);

bbox=step(faceDetector,im);

im1 = imcrop(im,bbox);

im2 = rgb2gray(im1);

imshow(im2);

title('Input Image');

Ftest=FeatureStatistical(im2);

load db.mat

Ftrain=db(:,1:2);

Ctrain=db(:,3);

for(i=1:size(Ftrain,1))

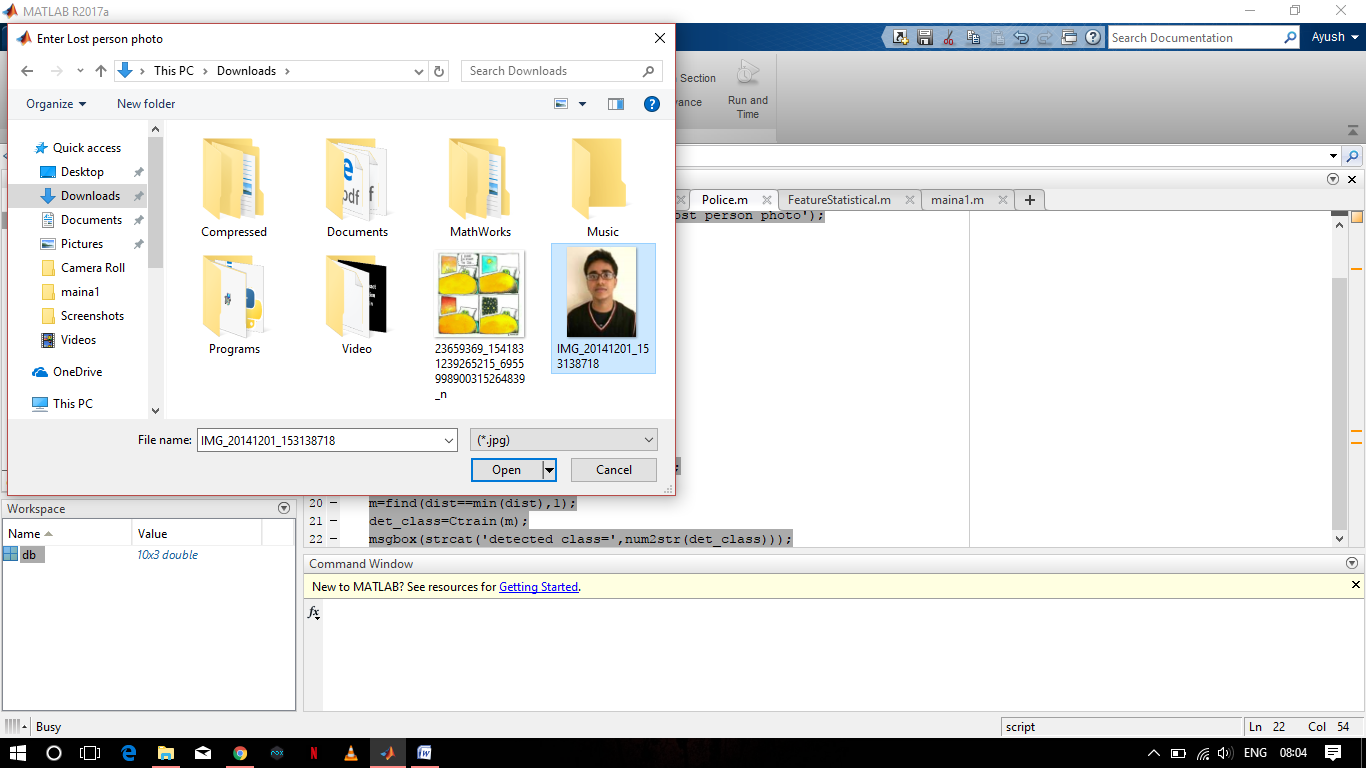
dist(i,:)=sum( abs(Ftrain(i,:)-Ftest));

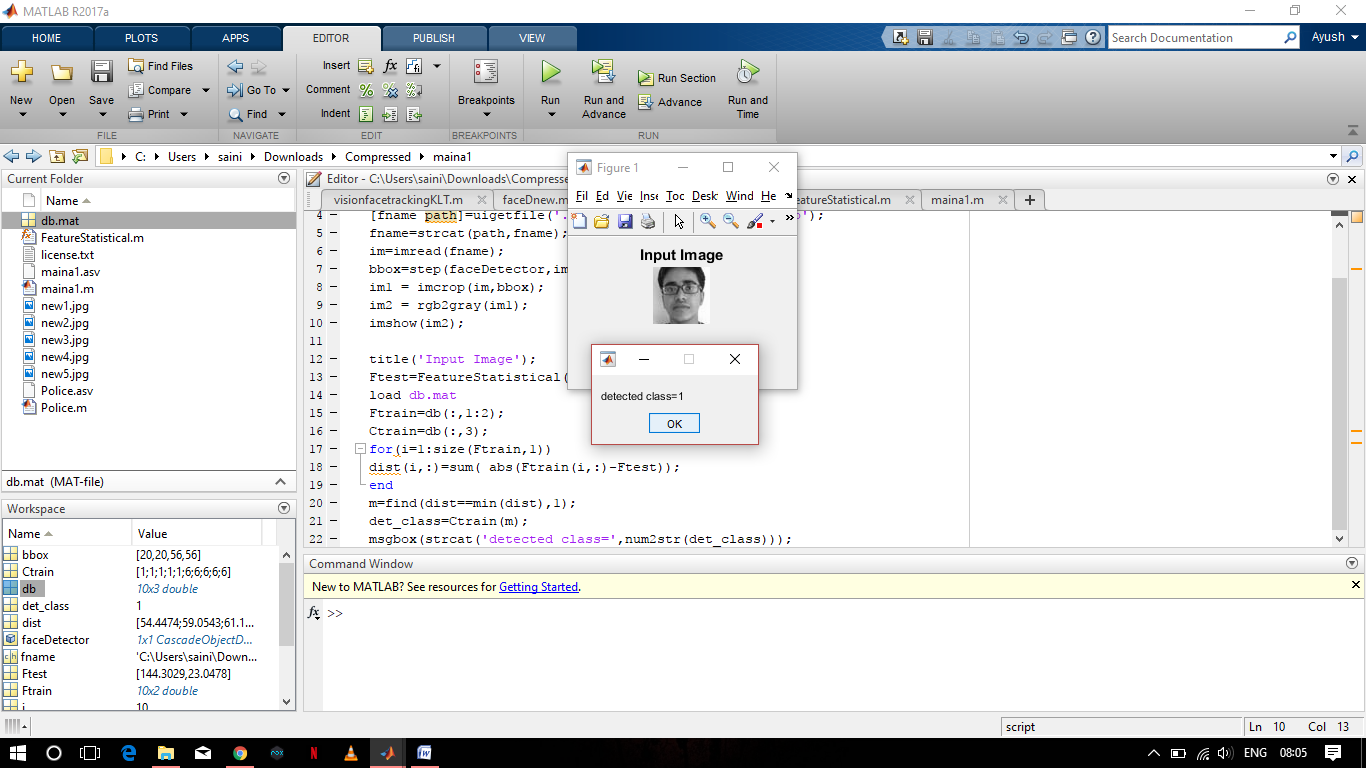
end

m=find(dist==min(dist),1);

det\_class=Ctrain(m);

msgbox(strcat('detected class=',num2str(det\_class)));

****

****

**FUTURE SCOPE**

Now, We have sucessfully recognised the face at the Police Station, we can now implement it with the GPS device and the rest of the system. Police Officials then will be able to get the last location of the person, Name address of the Taxi driver, etc.  
We also have to work on the security of the database system so not to allow anyone else the police offiicials to access the database