A
Project Report
On

"Convict Recognition"

Prepared by:
Nikunj Khandar (16IT041)
Jimesh Langadiya (16IT045)

Under the supervision of
Prof. Kamlesh Makvana

A Report Submitted to
Charotar University of Science and Technology
for Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology
in Information Technology
IT345 Software Group Project-II (5th sem)

Submitted at

DEPARTMENT OF INFORMATION TECHNOLOGY
Chandubhai S. Patel Institute of Technology
At: Changa, Dist: Anand – 388421
October 2018

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

This is to certify that the report entitled "**Convict Recognition"** is a bonafied work carried out by **Nikunj Khandar (16IT041) and Jimesh Langadiya (16IT045)** under the guidance and supervision of **Prof. Kamlesh Makvana** for the subject **Software Group Project-II (IT345)** of **5th** Semester of Bachelor of Technology in **Information Technology** at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate herself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,

Prof. Kamlesh Makvana
Assistant Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Parth Shah
Head & Associate Professor
Department of Information Technology
CSPIT, Changa, Gujarat.

Chandubhai S Patel Institute of Technology

At: Changa, Ta. Petlad, Dist. Anand, PIN: 388 421. Gujarat

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to Prof Kamlesh Makvana for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents & member of Charusat University for their kind co-operation and encouragement which help me in completion of this project.
We would like to express our special gratitude and thanks to industry persons for giving us such attention and time.
My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Nikunj Khandar
(16IT041)

Jimesh Langadiya
(16IT045)

ABSTRACT

There is an abnormal increase in the crime rate and also the number of criminals is increasing, this leads towards a great concern about the security issues. Crime preventions and criminal identification are the primary issues before the police personnel, since property and lives protection are the basic concerns of the police but to combat the crime, the availability of police personnel is limited. With the advent of security technology, cameras especially CCTV have been installed in many public and private areas to provide surveillance activities. The footage of the CCTV can be used to identify suspects on scene. In this paper, an automated facial recognition system for criminal database was proposed using known Haar feature-based cascade classifier. This system will be able to detect face and recognize face automatically in real time. An accurate location of the face is still a challenging task. Viola-Jones framework has been widely used by researchers in order to detect the location of faces and objects in a given image. Face detection classifiers are shared by public communities, such as OpenCV.

CHAPTER- 1: INTRODUCTION

1.1: Description

Face is most commonly used biometric to recognize people. Face recognition has received substantial attention from researchers due to human activities found in various applications of security like airport, criminal detection, face tracking, forensic etc. Compared to other biometric traits like palm print, Iris, finger print etc., face biometrics can be non-intrusive. They can be taken even without user's knowledge and further can be used for security based applications like criminal detection, face tracking, airport security, and forensic surveillance systems. Face recognition involves capturing face image from a video or from a surveillance camera. They are compared with the stored database. Face biometrics involves training known images, classify them with known classes and then they are stored in the database. When a test image is given to the system it is classified and compared with stored database. Face biometrics is a challenging field of research with various limitations imposed for a machine face recognition like variations in head pose, change in illumination, facial expression, aging, occlusion due to accessories etc. Various approaches were suggested by researchers in overcoming the limitations stated.

1.2: Purpose:

The face is crucial for human identity. It is the feature which best distinguishes a person. Face recognition is an interesting and challenging problem and impacts important applications in many areas such as identification for law enforcement, authentication for banking and security system access [8], and personal identification among others. Face recognition is an easy task for humans but it's entirely different task for a computer. A very little is known about human recognition to date on How do we analyze an image and how does the brain encode it and Are inner features (eyes, nose, mouth) or outer features (head shape, hairline) used for a successful face recognition? Neurophysiologist David Hubel and Torsten Wiesel has shown that our brain has specialized nerve cells responding to specific local features of a scene, such as lines, edges, angles or movement. Since we don't see the world as scattered pieces, our visual cortex must somehow combine the different sources of information into useful patterns. Automatic face recognition is all about extracting those meaningful features from an image, putting them into a useful representation and performing some classifications on them. Face recognition based on the geometric features of a face is probably the most instinctive approach for Human identification. The whole process can be divided in three major steps where the first step is to find a good database of faces with multiple images for each individual. The next step is to detect faces in the database images and use them to train the face recognizer and the final step is to test the face recognizer to recognize faces it was trained for. Nowadays, face detection is used in many places especially the websites hosting images like Picassa, Photobucket and Facebook. The automatically tagging feature adds a new dimension to sharing pictures among the people who are in the picture and also gives the idea to other people about who the person

is in the image. In our project, we have studied and implemented a pretty simple but very effective face detection algorithm which takes human skin color into account. Our aim, which we believe we have reached, was to develop a system that can be used by police or investigation department to recognize criminal from their faces. The method of face recognition used is fast, robust, reasonably simple and accurate with a relatively simple and easy to understand algorithms and technique.

1.3: Scope:

This Project detect face and recognition the Convict Person in public cameras. So that we can trace that person and stop the crime rate of the city and we can also find the person that try to break prison and escape from the prison

1.4: Objective:

PREVENT RETAIL CRIME

Face recognition is currently being used to instantly identify when known shoplifters, organized retail criminals or people with a history of fraud enter retail establishments. Photographs of individuals can be matched against large databases of criminals so that loss prevention and retail security professionals can be instantly notified when a shopper enters a store that prevents a threat. Face recognition systems are already radically reducing retail crime. According to our data, face recognition reduces external shrink by 34% and, more importantly, reduces violent incidents in retail stores by up to 91%.

PROTECT LAW ENFORCEMENT

Mobile face recognition apps, like the one offered by Face First, are already helping police officers by helping them instantly identify individuals in the field from a safe distance. This can help by giving them contextual data that tells them who they are dealing with and whether they need to proceed with caution. As an example, if a police officer pulls over a wanted murderer at a routine traffic stop, the officer would instantly know that the suspect may be armed and dangerous, and could call for reinforcement.

AID FORENSIC INVESTIGATIONS

Facial recognition can aid forensic investigations by automatically recognizing individuals in security footage or other videos. Face recognition software can also be used to identify dead or unconscious individuals at crime scenes.

PROTECT SCHOOLS FROM THREATS

Face recognition surveillance systems can instantly identify when expelled students, dangerous parents, drug dealers or other individuals that pose a threat to school safety enter school grounds. By alerting school security guards in real time, face recognition can reduce the risk of violent acts.

CHAPTER- 2: SYSTEM REQUIEMENTS

2.1: User characteristics:

User must know how to operate basic Python code and able to compile and run that code.

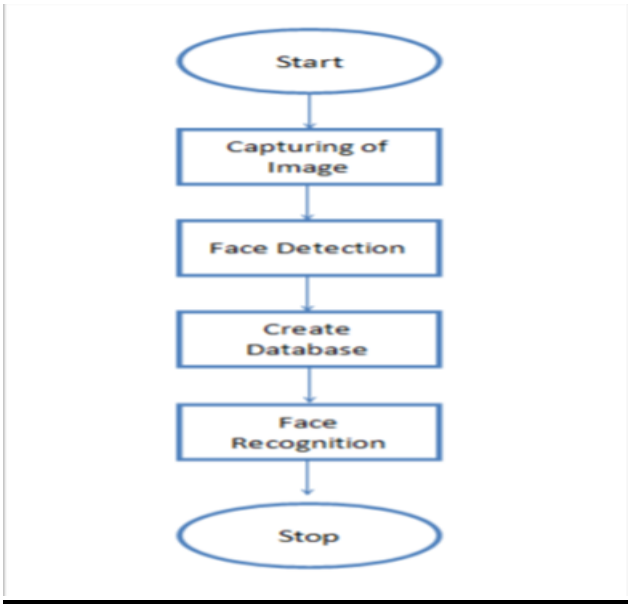
User should have bit of knowledge about database.

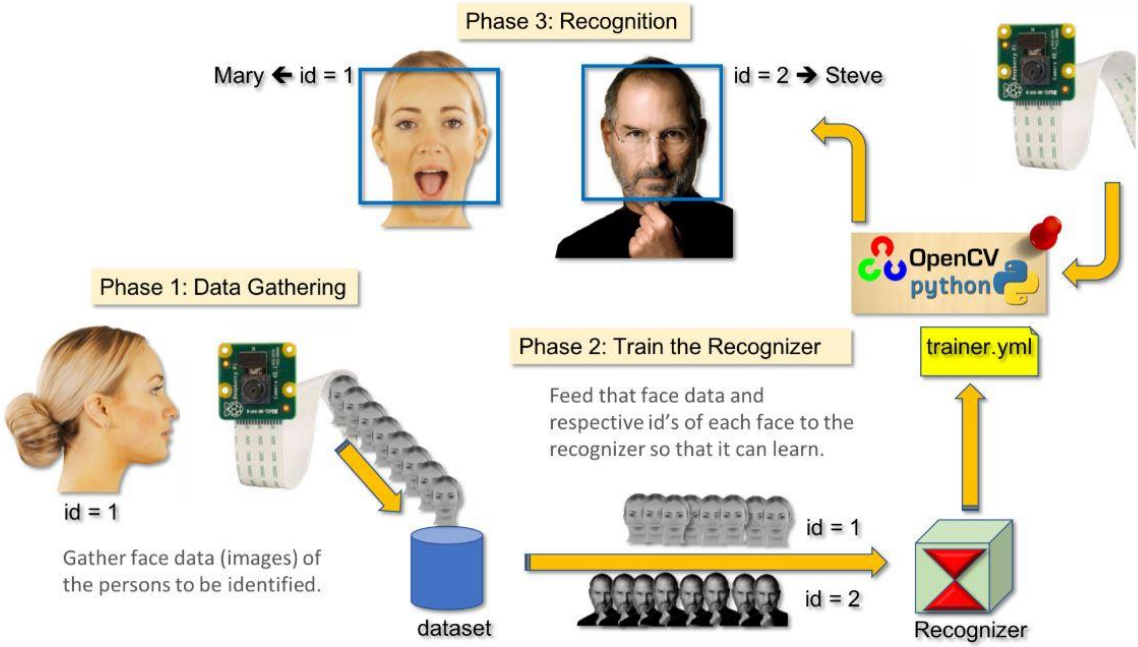
2.2: Tools and languages used (For development):

- Programming Language: Python
- IDE: Spyder
- Open CV (Open source computer vision library).
- NumPy (for supporting large multi-dimensional arrays).
- OS library (File handling)
- PIL (Python imaging library).
- SQL Lite (For Database connectivity)
- Camera

CHAPTER-3: SYSTEM DESIGN

3.1: Project Flow:





CHAPTER- 4: IMPLEMENTATION PLANNING

4.1: OpenCV and LBPH

OpenCV (Open Source Computer Vision) is a bunch of programming functions which is used for real-time computer vision, developed by Intel's research center which was supported by Willow Garage and Itseez is maintaining now. OpenCV was developed to bring a common platform for applications of computer vision and also accelerate the use of commercial products in machine perception. OpenCV makes easy for businesses to modify and utilize the code since it is a BSD-licensed.

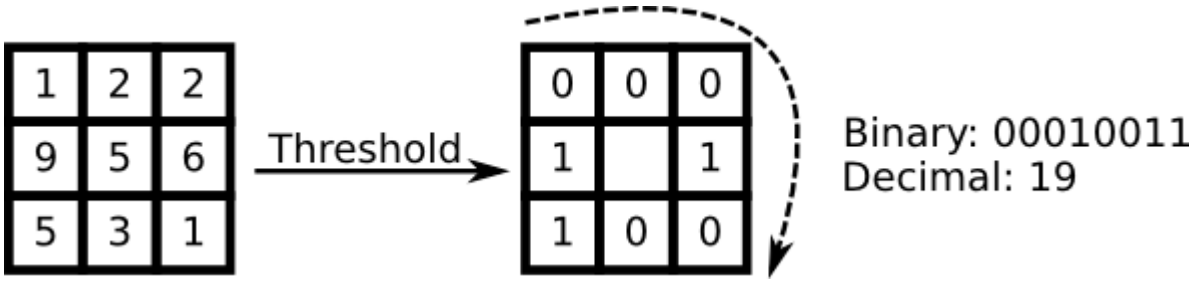
You treat your data as a vector somewhere in a high-dimensional image space. We all know high-dimensionality is bad, so a lower-dimensional subspace is identified, where (probably) useful information is preserved. The Eigen faces approach maximizes the total scatter, which can lead to problems if the variance is generated by an external source, because components with a maximum variance over all classes aren't necessarily useful for classification (see http://www.bytefish.de/wiki/pca_lda_with_gnu_octave). So to preserve some discriminative information we applied a Linear Discriminant Analysis and optimized as described in the Fisherfaces method. The Fisherfaces method worked great... at least for the constrained scenario we've assumed in our model.

Now real life isn't perfect. You simply can't guarantee perfect light settings in your images or 10 different images of a person. So what if there's only one image for each person? Our covariance estimates for the subspace *may* be horribly wrong, so will the recognition. Remember the Eigenfaces method had a 96% recognition rate on the AT&T Facedatabase? How many images do we actually need to get such useful estimates? Here are the Rank-1 recognition rates of the Eigenfaces and Fisherfaces method on the AT&T Facedatabase, which is a fairly easy image database:

So in order to get good recognition rates you'll need at least 8(+-1) images for each person and the Fisherfaces method doesn't really help here. The above experiment is a 10-fold cross validated result carried out with the facerec framework at: https://github.com/bytefish/facerec. This is not a publication, so I won't back these figures with a deep mathematical analysis. Please have a look into [KM01] for a detailed analysis of both methods, when it comes to small training datasets.

So some research concentrated on extracting local features from images. The idea is to not look at the whole image as a high-dimensional vector, but describe only local features of an object. The features you extract this way will have a low-dimensionality implicitly. A fine idea! But you'll soon observe the image representation we are given doesn't only suffer from illumination variations. Think of things like scale, translation or rotation in images - your local description has to be at least a bit robust against those things. Just like **SIFT**, the Local Binary Patterns methodology has its roots in 2D texture analysis. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the

center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. You'll end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels you'll end up with 2^8 possible combinations, called *Local Binary Patterns* or sometimes referred to as *LBP codes*. The first LBP operator described in literature actually used a fixed 3 x 3 neighborhood just like this:



4.2: Algorithm Development

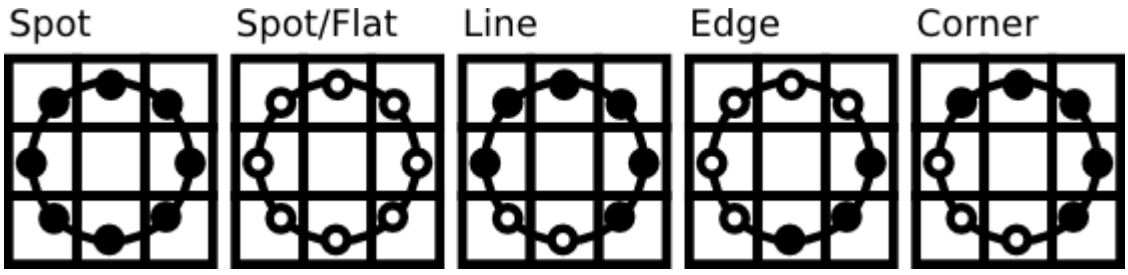
A more formal description of the LBP operator can be given as:

$$\mathrm{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \qquad (1)$$

This description enables you to capture very fine grained details in images. In fact the authors were able to compete with state of the art results for texture classification. Soon after the operator was published it was noted, that a fixed neighborhood fails to encode details differing in scale. So the operator was extended to use a variable neighborhood in [AHP04]. The idea is to align an abritrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:



For a given Point (x_c, y_c) the position of the neighbor $(x_p, y_p), p \in P$ can be calculated by:

$$x_p = x_c + R\cos(\frac{2\pi p}{P})$$

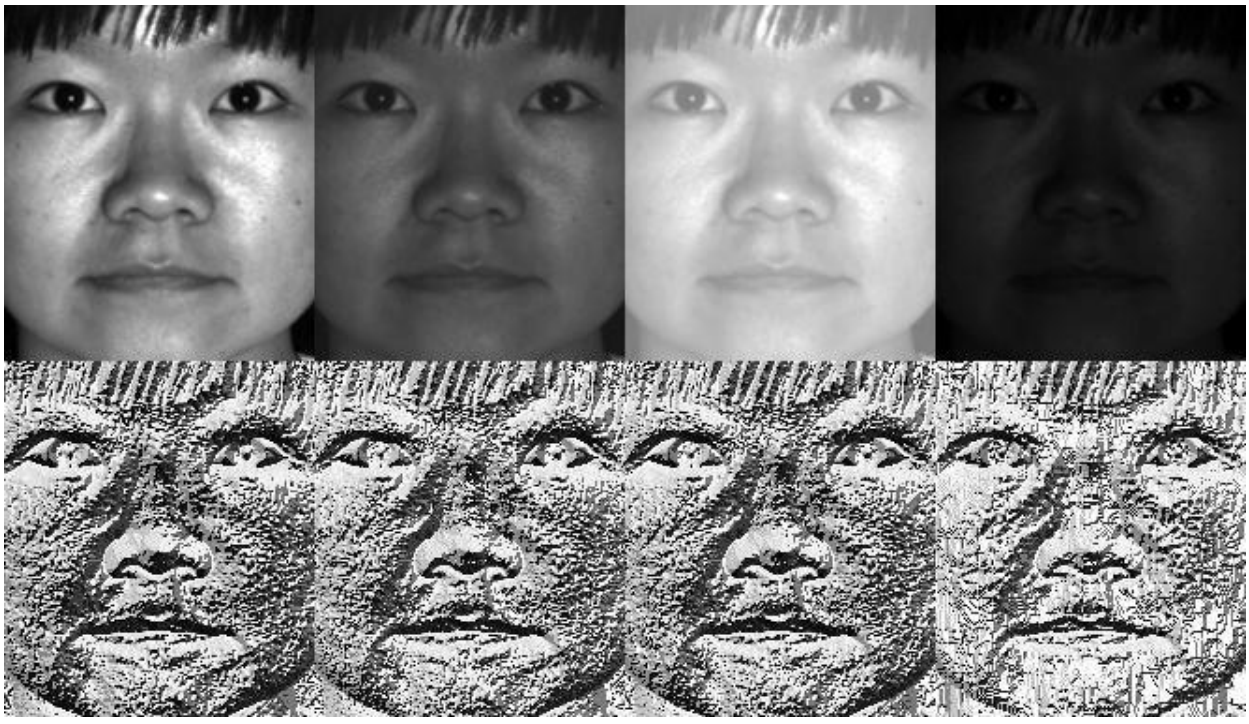
$$y_p = y_c - R\sin(\frac{2\pi p}{P})$$

Where R is the radius of the circle and P is the number of sample points.

The operator is an extension to the original LBP codes, so it's sometimes called *Extended LBP* (also referred to as *Circular LBP*) . If a points coordinate on the circle doesn't correspond to image coordinates, the point get's interpolated. Computer science has a bunch of clever interpolation schemes, the OpenCV implementation does a bilinear interpolation:

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

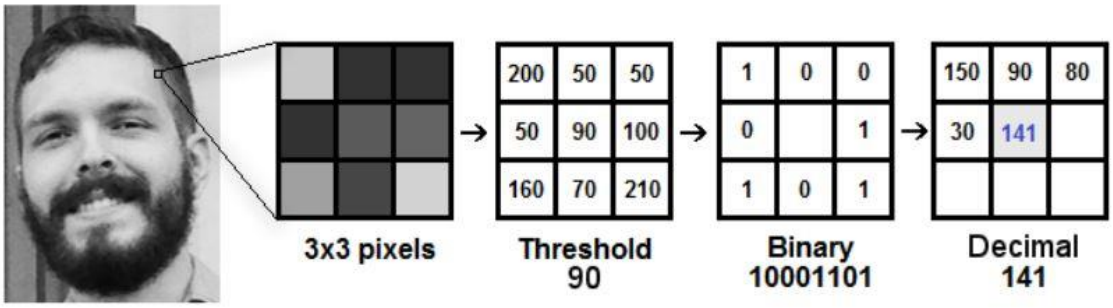
By definition the LBP operator is robust against monotonic gray scale transformations. We can easily verify this by looking at the LBP image of an artificially modified image (so you see what an LBP image looks like!):



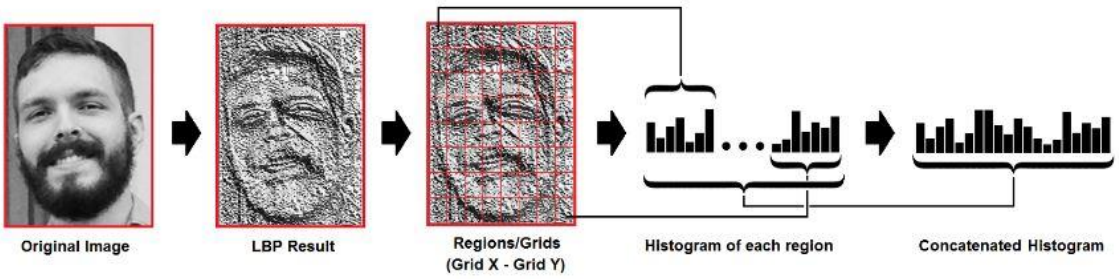
So what's left to do is how to incorporate the spatial information in the face recognition model. The representation proposed by Ahonen et. al [AHP04] is to divide the LBP image into m local regions and extract a histogram from each. The spatially enhanced feature vector is then obtained by concatenating the local histograms (**not merging them**). These histograms are called *Local Binary Patterns Histograms*.

Working of Algorithm

The threshold frequency of the captured images is converted in to binary form and further it is converted into decimal matrix form.



Further using the matrix a histogram is plotted and is matched with the histogram of the existing image in the database and the result matching is given as an output.



4.3: Snapshots of project

Fig1.1: Face Detection

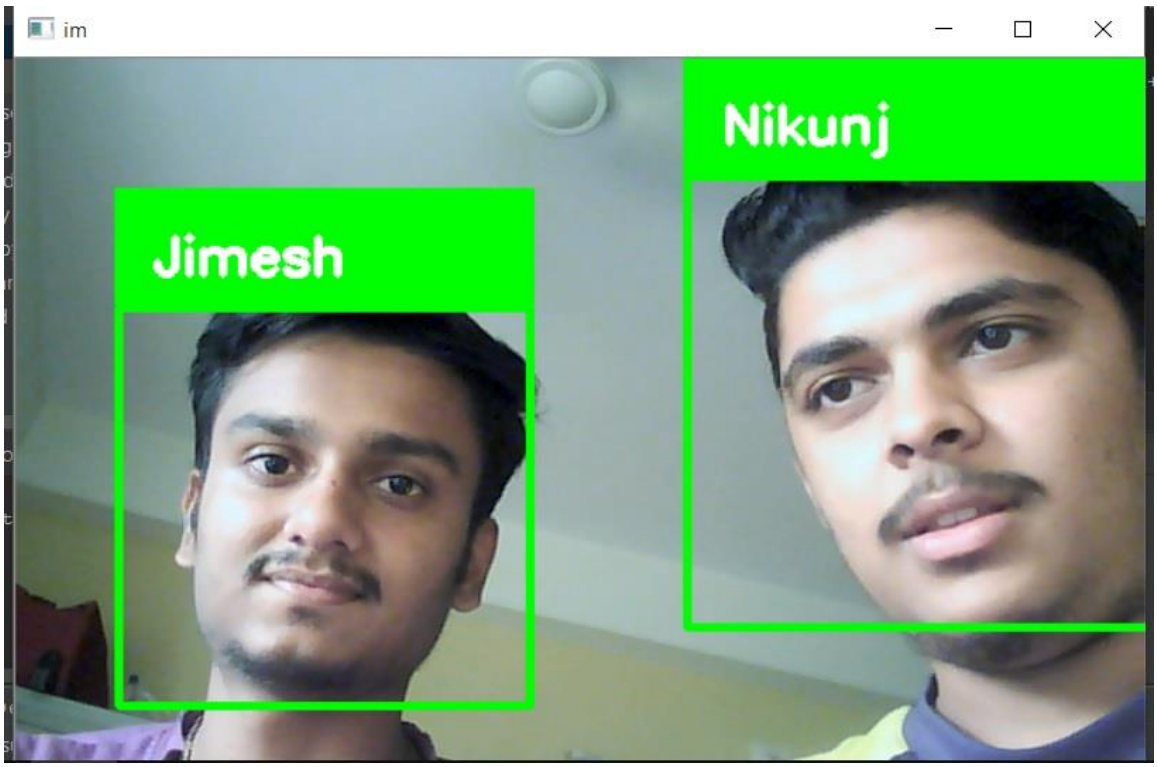


Fig1.2: Datasets (Images of the person a captured frame by frame and stored into a particular location)

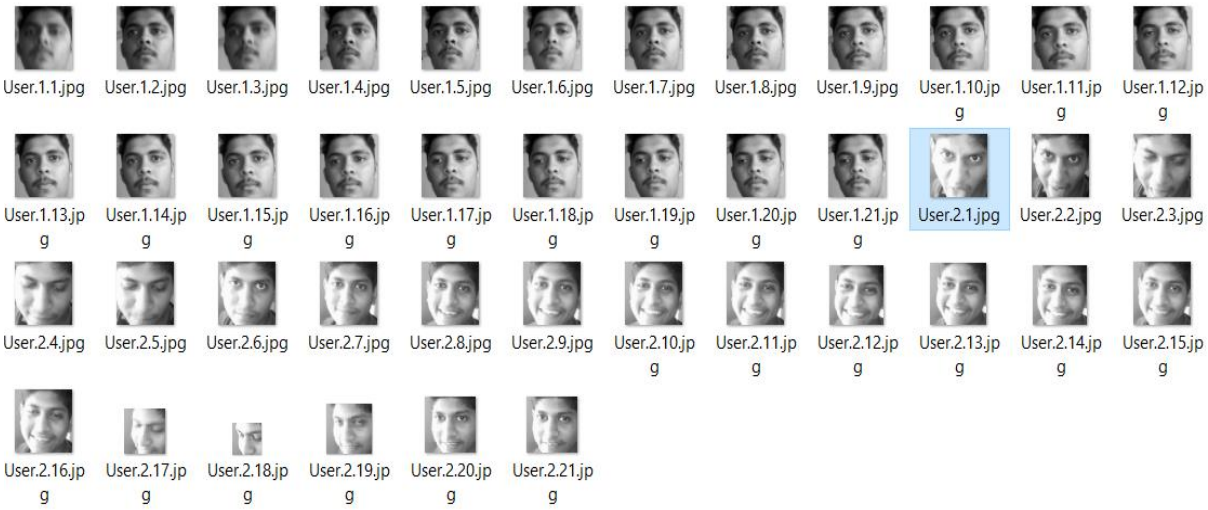


Fig 1.3: Training the faces as per LBPH (Local Binary Patterns Histogram) and generating a .yml file

```
%YAML:1.0
---
radius: 1
neighbors: 8
grid_x: 8
grid_y: 8
histograms:
   - !!opencv-matrix
      rows: 1
      cols: 16384
      dt: f
      data: [ 5.20833349e-03, 5.20833349e-03, 0., 1.73611112e-03,
          3.47222225e-03, 0., 0., 6.94444450e-03, 0., 0., 0., 0., 0., 0.,
          0., 0., 1.73611112e-03, 0., 0., 0., 0., 0., 0., 1.73611112e-03,
          0., 0., 0., 0., 1.73611112e-03, 0., 0., 1.04166670e-02, 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.73611112e-03,
          1.73611112e-03, 3.47222225e-03, 0., 0., 0., 0., 0., 0.,
          1.73611112e-03, 0., 0., 0., 0., 0., 0., 0., 0., 1.73611112e-03,
          0., 0., 0., 0., 0., 0., 1.73611112e-03, 0., 0., 0., 0.,
          3.47222225e-03, 0., 0., 1.73611112e-03, 0., 0., 0., 0., 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.73611112e-03, 0., 0.,
          0., 0., 0., 0., 0., 0., 0., 0., 0., 3.47222225e-03, 0., 0.,
          3.47222225e-03, 0., 3.47222225e-03, 0., 1.04166670e-02, 0., 0.,
          0., 2.60416660e-02, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
```

Fig 1.4: Database with large amount of data of persons in SQL Lite Studio

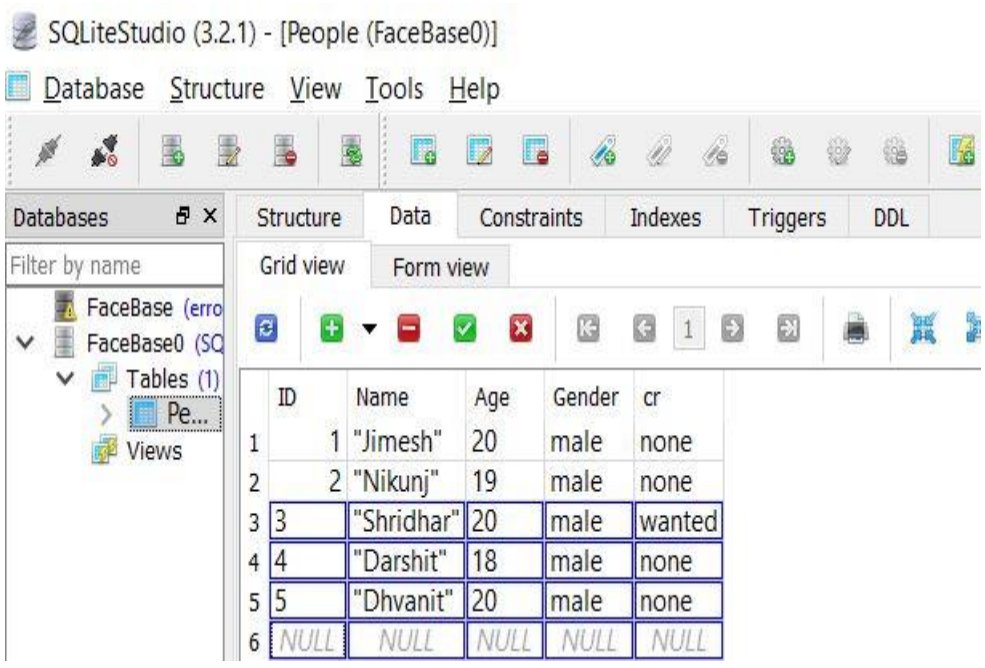


Fig 1.5: Harrcascade for Frontalface detection

```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
            -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
```

Fig1.6 : Criminal Face recognition in local/public cameras

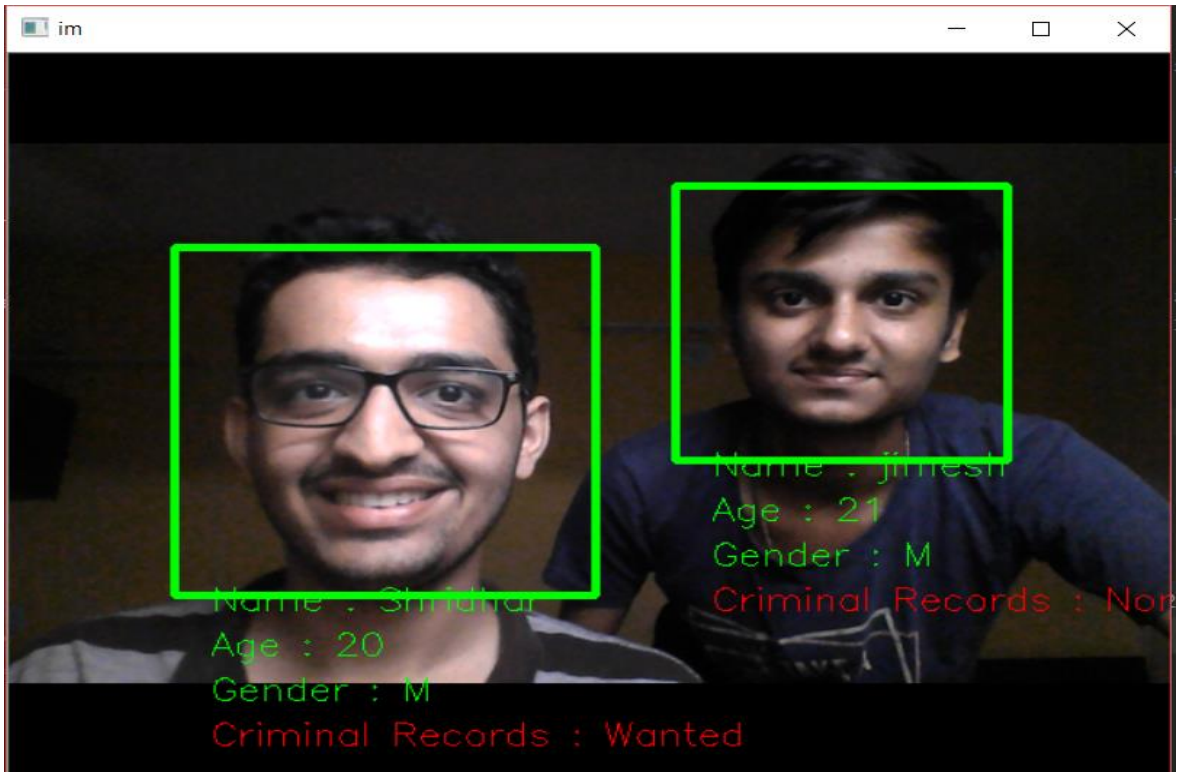


Fig 1.7 Actual implementation in the public cameras

Description : The below image shows real time example of a convict recognition through public cameras. All the information of the convict database is embedded in the public surveillance system and a watchlist is observed through the cameras. At a point when any convict gets detected in any camera his/her location and timing can be known and he can be again taken behind the bars.

4,4:Psuedo Code:

1-Face_datasets.py

Step--1#Create Cascade with the help of Haarcascade for fontal face

```
vid_cam = cv2.VideoCapture ( 0 )

face_detector = cv2.CascadeClassifier ( 'haarcascade_frontalface_default.xml' )
def insertOrUpdate(Id, Name):
      cursor = conn.execute ( cmd )
```

Step--2#Implementaion for video camera read and take images

```
while (True):

   _, image_frame = vid_cam.read ()

   gray = cv2.cvtColor ( image_frame, cv2.COLOR_BGR2GRAY )

   faces = face_detector.detectMultiScale ( gray, 1.3, 5 )

   for (x, y, w, h) in faces:
      # Crop the image frame into rectangle
      cv2.rectangle ( image_frame, (x, y), (x + w, y + h), (255, 0, 0), 2 )

      # Increment sample face image
      count += 1
```

Step--3 # Save the captured image into the datasets folder

```
      cv2.imwrite ( "dataset/User." + str ( Id ) + '.' + str ( count ) + ".jpg", gray[y:y + h, x:x +
w] )

      cv2.imshow ( 'frame', image_frame )
```

 # Stop video

```
vid_cam.release ()
```

2-Training.py

Step--1#Algorithm used is LBPH so Learning face

```
recognizer = cv2.face.LBPHFaceRecognizer_create()

# Using prebuilt frontal face training model, for face detection
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

Step--2# Create method to get the images and label data
def getImagesAndLabels(path):

    # Get all file path
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]

      for (x,y,w,h) in faces:

          # Add the image to face samples
          faceSamples.append(img_numpy[y:y+h,x:x+w])
          print(id)

          ids.append(id)
          cv2.imshow("training",img_numpy)
          cv2.waitKey(10)

Step--3# Save the model into trainer.yml
assure_path_exists('trainer/')
recognizer.save('trainer/trainer.yml')
cv2.destroyAllWindows()
```

3-Face_Recognition.py

Step—1#Parth for trainer.yml

```
assure_path_exists("trainer/")

recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
```

Step—2#Open camera for detection face

```
while True:
    # Create rectangle around the face
    cv2.rectangle(im, (x - 20, y - 20), (x + w + 20, y + h + 10), (0, 255, 0), 4)

    # Recognize the face belongs to which ID
     Id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
      cv2.putText(im, "Name : {0}".format(str(profile[1])), (x, y + h + 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                (0, 255, 0));
          cv2.putText(im, "Age : {0}".format(str(profile[2])), (x, y + h + 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                (0, 255, 0));
          cv2.putText(im, "Gender : {0}".format(str(profile[3])), (x, y + h + 90),
cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                (0, 255, 0));
          cv2.putText(im, 'Criminal Records : ' + str(profile[4]), (x, y + h + 120),
cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                (0, 0, 255));
```

Step—3#Exit Camera

```
cam.release()
cv2.destroyAllWindows()
```

CHAPTER-5: LIMITATIONS AND FUTURE ENHANCEMENT

5.1: Limitations

- Accuracy is less
- Implementation on public cameras.
- Less Camera quality

5.2: Future Enhancements

- Work on Local-Color-Vector Binary Pattern (LCVBP).
- Develop Attractive front end
- Increase Accuracy
- To reduce the false-positives drastically

CHAPTER 6 : CONCLUSION

6.1: Conclusion

In this project, we are able to detect and recognize faces of the criminals in an image and in a video stream obtained from a camera in real time. We have used Haar feature-based cascade classifiers in OpenCV approach for face detection. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Also, we have used Local Binary Patterns Histograms(LBPH) for face recognition. Several advantages of this algorithm are: Efficient feature selection, Scale and location invariant detector, instead of scaling the image itself, we scale the features Such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, sign boards, number plates etc). LBPH recognizer can recognize faces in different lighting conditions with high accuracy. Also, LBPH can recognize efficiently even if single training image is used for each person. Our application has some disadvantages like: 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.

6.2: References

- **https://opencv.org/links.html**
- **https://pythonprogramming.net/loading-images-python-opencv-tutorial/**
- **https://ijarcce.com/upload/2018/march-18/IJARCCE%2046.pdf**
- https://link.springer.com/chapter/10.1007/978-3-540-24670-1_36
- http://docs.opencv.org/3.0-beta/modules/face/doc/facerec/facerec_api.html