Smart Surveillance System Using Artificial Intelligence

[1]Prof. Puneshkumar U. Tembhare, Assistant Professor, Priyadarshini College Of Engineering, Nagpur, Maharashtra

[2]Anup K. Katre, Bachelor of Engineering IV yr, Priyadarshini College Of Engineering, Nagpur, Maharashtra

[3]Dhanashri A. Adgurwar, Bachelor of Engineering IV yr, Priyadarshini College Of Engineering, Nagpur, Maharashtra

[4]Sharayu M. Bagmare, Bachelor of Engineering IV yr, Priyadarshini College Of Engineering, Nagpur, Maharashtra

[5]Vaibhav D. Rahangdale, Bachelor of Engineering IV yr, Priyadarshini College Of Engineering, Nagpur, Maharashtra

[6]Sanket S. Harinkhede, Bachelor of Engineering IV yr, Priyadarshini College Of Engineering, Nagpur, Maharashtra

Abstract: Recognition of the face is one of the biometric techniques used to recognize any facial image. Since birth, faces have played a crucial role in any individual's social interaction that gives the individual a distinctive identity. Face detection and recognition is a very popular topic in biometric research as it is used for the safety and security of an individual. Face recognition technology has attracted a large number of researchers due to its wide range of application value as well as market potentials, such as fraud detection and video surveillance. Face recognition plays a very important role in the surveillance system, as there is no need for the participation of an object. We used an image set algorithm with the help of OpenCV and Python programming and having a pre-existing dataset of faces. The module is divided into three sub modules: 1] Detection module 2] Training module 3] recognition module.

Index Terms **- Face detection, Face Recognition, OpenCV, Haar cascade, Adaboost algorithm.**

I. INTRODUCTION

The application of facial recognition was widely viewed as something straight out of science fiction. Yet this revolutionary technique has not only been feasible over the past decade but has also been popular. Face detection and recognition is a technology used to recognize an object from a source of a video or a picture. Woodrow Wilson Bledsoe, Helen Chan Wolf, and Charles Bisson were the founders of facial recognition. In the 1960s Woodrow Wilson Bledsoe presented the idea of facial recognition. Bledsoe developed a system called RAND tablet, which could hand-classify images of people. RAND tablet is a tool that people might use with a stylus that released electromagnetic pulses to enter horizontal and vertical coordinates into a grid. The program may be used to manually monitor the coordinate positions of various facial features like the eyebrows, nose, hairline and mouth. Then, these data collection may be put into a database. Then, when a new image of person was presented to the machine, it was able to retrieve the image from the database that more closely matched the person. Since then the recognition system has been continuously being developed and refined, the device is slowly maturing and eventually being used in everyday human life. Several branches benefit from this development. Law enforcement authorities use facial recognition technologies to protect societies against impostors. Retailers are into robbery and conflict reduction. Airports improve comfort and health for passengers. Mobile phone providers today use facial recognition technologies to provide new levels of biometric authentication for users. In this article, together with OpenCV, we propose a face detection and recognition framework with the aid of python. The framework comprises three modules which are module for identification, module for instruction and module for acknowledgment. Recognition module recognizes the face that gets into the camera's field of vision and saves the face in JPG format as file. The training modules then train the system with the aid of the Haar Cascade Algorithm proposed by Paul Viola and Michael Jones.

The purpose of this article is to provide a simpler routine for human-machine interaction when user authentication is needed through face detection and recognition. A computer can identify and recognize a person's face with the help of a regular web camera; a custom login screen will be created with the ability to filter user access based on facial features of the users. The aims of this study are to include a collection of detection algorithms which can later be bundled in an easily portable framework among the various processor architectures we see in today's machines (computers).Viola-Jones uses the Haar-Like Functionality, Cascade Filtering, and Adaboost principle. This algorithm goes through 3 steps for face sensing:

I.1 Integration of Image

The Haar-like function model is used. Alternatively, the principle of Integral Image computes hair-like characteristics. Integral Image means the Summed Area Table principle (a data structure and an algorithm), which is applied to measure the sum of values in a rectangular subset of the grid effectively and efficiently. [7] The integral picture principle thus measures in constant

time rectangular functions (hair characteristics). The position integral image (x, y) is the sum of the pixels above and to the left of (x, y), inclusive.

The key advantage of a hair-like feature over most other features is its calculation speed. A hair-like feature of any size can be measured in constant time due to the use of integral images (about 60 microprocessor instructions for a 2-rectangle function). Haar Wavelets or Haar Features used is called as Viola-Jones Algorithm.

I.2 Usage Of Adaboost Algorithm

We are only interested in selected few features from the large number of features computed in part I which would enable us to detect the face with great precision. To this, we use the Adaboost Algorithm to pick the key features and to train classifiers that will use them. The purpose of this algorithm is to construct a strong classifier from a linear weak classification combination. Adaboost is an effective boosting algorithm combining simple statistical learners thus significantly reducing both the training error and the more elusive generalization error. A concept called Adaboost which selects the best features and then trains the classifiers which is used. This algorithm creates a sturdy classifier using a linear combination of weighted simple weak classifiers.

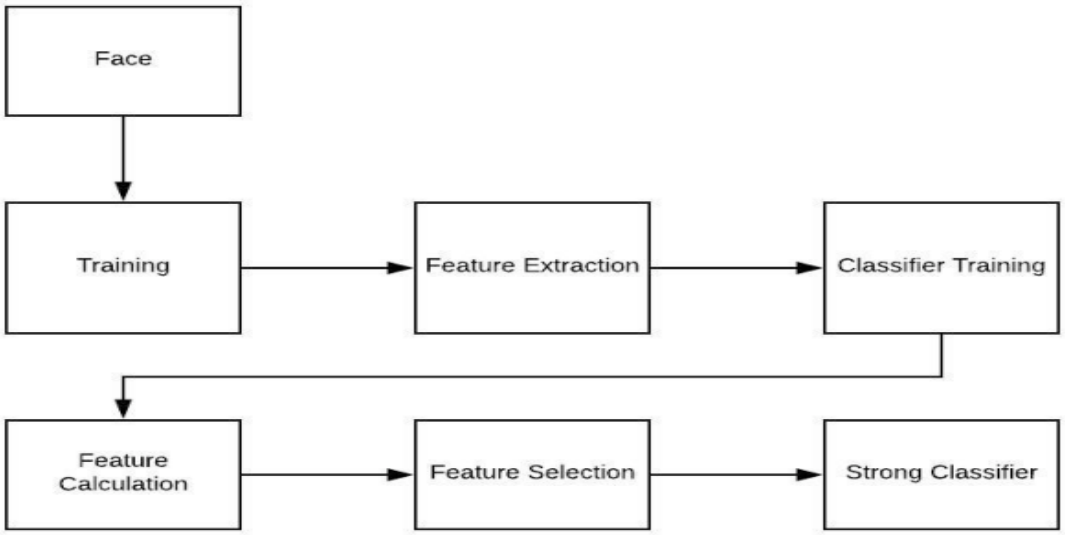


FIGURE 1: Adaboost Training model

Algorithm: - The **Adoboost algorithm.**

1. Input: S = {(x1,y1), ...,(xN,yN)} Number of iterations T.
2. Initialize: d (1) n = 1/N for all n = 1, ..., N
3. Do for t = 1, ..., T,

(a) Train classifier with respect to the weighted sample set {S, d (t)}and obtain hypothesis ht : x 7−→ {−1, +1}, i.e. ht = L(S, d (t)).
(b) Calculate the weighted training error εt of ht : εt = X N n=1 d (t) n I(yn 6= ht(x)n),
(c) Set : αt = 1 2 log 1 − εt εt .
(d) Update the weights: d (t+1) n = d (t) n exp{−αtynht(xn)}/Zt , where Zt is a normalization constant, such that PN n=1 d (t+1) n = 1.

4. Break if εt = 0 or εt ≥ 1 2 and set T=t-1.
5. Output: fT (x) = PT t=1 P αt T r=1 αr ht(x)

I.3 Building Cascade Structure

This cascade is made up of classifiers. This operates in a way that the initial classifiers are simpler and are used to reject the majority of sub-windows, and to obtain low false positives at the end of complex classifiers. [9] Classifiers are trained using the Adaboost Algorithms principle mentioned above. The deeper we go into the cascade the harder the classifier's. The cascade classifier comprises of a number of stages, where each stage is a group of weak learners. These weak learners are simple classifiers which are known as decision stumps. Each stage is trained by using a technique called boosting. Boosting provides the potential to train a highly precise classifier by taking the weighted average of decisions which are made by the weak learners.

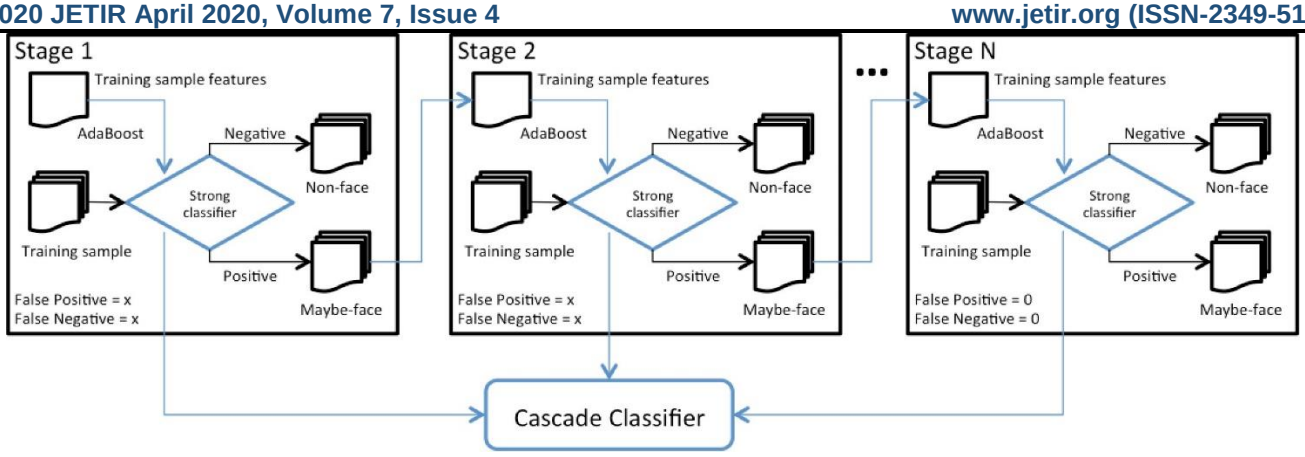


FIGURE 2: Flow chart of Cascade Classifier

Finally, in the recognition module the fundamental components of the face from the new video are extracted. Then those features get compared with the list of elements stored during training and the ones with the best match is found and name of the person recognized is displayed. This monitoring system fulfills the basic needs of face detection and recognition system, also takes the cost into consideration to ensure the pervasive mode as economical as possible. Furthermore, it can also be combined with real-time analysis algorithms.

II. PROBLEM DEFINITION

II.1 Face Detection

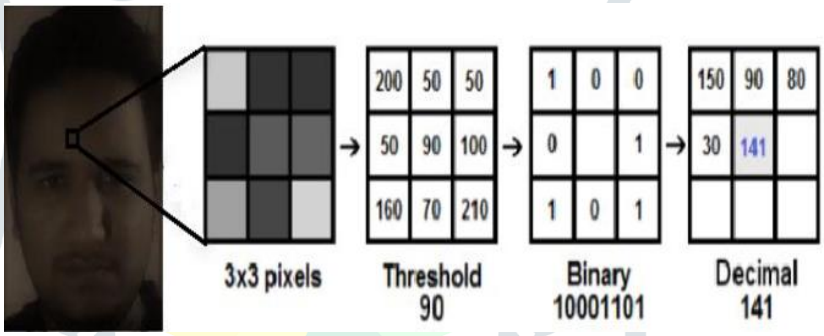


FIGURE 3: Face pixels selection

This device is capable of identifying the faces of the image taken from HD Video to analyze and identify the faces. From Section IV above, face detection determines where a face is located in an image, and is achieved by scanning the different image scales and extracting the exact patterns to identify the face. The Prototype is designed from OpenCV with the Haar-Like feature. Face detection for hair classifiers is used to create a search window that slides through an image and tests whether or not a certain region of an image looks like face. To identify a certain picture as face or non-face, hair-like features and a large set of very weak classifier use one single feature. That function is defined by the prototype and its coordinate relative to the search window that is the origin of the feature's size. The search window scans the first classifier on the cascade quickly if the classifier returns false then the calculation on that window also ends and no detected facial (positive) occurs. In addition, if the classifier returns true, then the window will be moved in the cascade to the next classifier to do the exact same thing. When all classifiers for that window return true, the result will also return true for the detection of certain window face.

II.2 Gray-Scale Image

We use gray scale image for image related operation because, gray scale image is a one-layer image from 0-255 whereas the colored image has three different layer also color increases the complexity of the model.

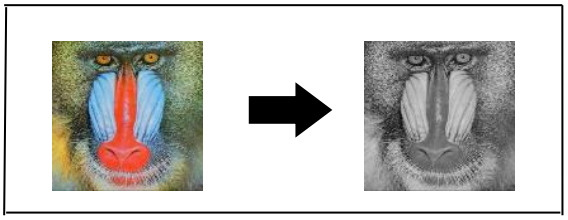


Figure 4**: Grey Scale Image**

II.3 Feature Extraction

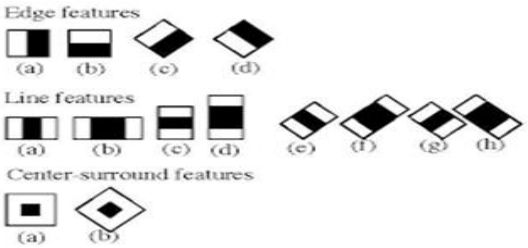


FIGURE 5: Haar Features

After the face detection step, human-face patches are removed from images via feature extraction phase. If we use these patches for face recognition directly, it may have some limitations; first, each patch usually holds over 1000 pixels, which are too enormous to build a robust recognition. Second, face patches may be taken under different illumination, with different face expressions, and with different camera alignment, and may suffer from occlusion and cluttering background. To overcome these drawbacks, we perform feature extractions to do dimensionality reduction, information packing, salience extraction, and noise cleaning. After this step, we transformed face patch into a vector with fixed dimension or a set of fiducial points and their corresponding locations. We can include feature extraction either in face detection or face recognition as per survey form some literatures.

FIGURE 6:Haar Feature Selection

II.4 Model Training

Simply put, model training is the process of taking several face representation belonging to the same person, and creating a classifier that can recognize more instances of face portrayal from the same person. The most favored approach is to use a linear SVM.

We've reportedly already obtained a model's collection of 100 datasets. The training data must include the correct response in this step which is known as an aim or aim attribute. In the training data, the learning algorithm seeks patterns that map the input data attributes to the target (the response you want to predict), and it generates an ML model that extracts those patterns. This means the learned model will be saved later and used for facial recognition.

In this step we will use Local Binary Pattern Histogram (LBPH) to consider the face structure and to evaluate patterns. We'll isolate and generalize the details so it can know what face belongs to somebody.

FIGURE 7: Training Model

The first step in creating the descriptor for the LBP texture is to transform the image to a grayscale. We pick a neighborhood of size r surrounding the center pixel for every pixel in the grayscale image. Of this middle pixel an LBP value is then measured and processed with the same width and height as the reference image in the resulting 2D array. Then pass the features onto our Linear SVM for classification.

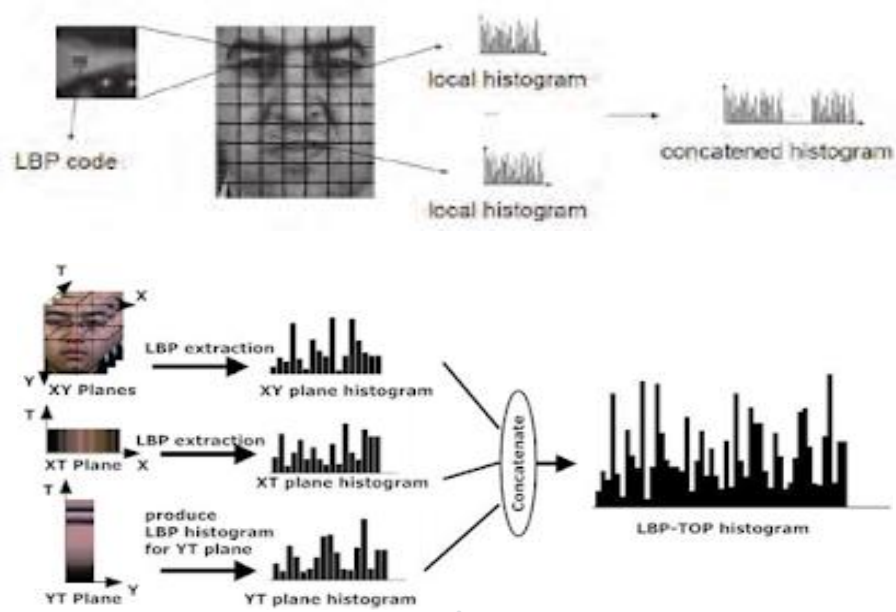


FIGURE 8: Training Dataset

II.5 Face Recognition

After formulating each face patch in the above steps, the last step is to recognize the identities of these faces. For face recognition, a face database is required to build. [4]For each individual, some images are taken and their features are removed and stored in the database. Whenever an input image came, we perform face detection and face extraction and after this we compare features of each face with stored database.

We differentiate two applications of face recognition: identification and verification. Face identification means given a face image, we want the system to tell who he / she is and identify the individual; while in face verification, given a face image and on the basis of an assumption from the identification, we want the system to tell true or false about the guess.

III. DISCRIPTOIN OF TOOLS

In this section, the tools and approach to implement and calculate face detection and tracking using OpenCV are detailed.

III.1 OPENCV

Open Source Computer Vision Library is commonly referred to as OpenCV. This is a library of methods of programming and focuses mainly on Intel's real-time machine vision. The library is a cross-platform library targeted primarily at image processing in real time. This C interface makes OpenCV portable for certain different platforms, such as digital signal processors. In order to increase adoption by a larger consumer, wrappers for languages like C #, Python, Ruby or Java have been created. Since version 2.0, however, OpenCV integrates both its conventional C interface and a modern C++ interface. This new interface aims to remove multiple lines of code which are critical for code-up vision functionality and to avoid common programming errors (memory leaks) that may occur when using OpenCV in C. The C++ interface is now used to build most of the inventions and algorithms in OpenCV. [9]It is much more difficult to deliver C++ code wrappers in other langues rather than C code; as there are usually a few latest OpenCV 2.0 features missing in other language wrappers.

III.2 NumPy

NumPy is a Python programming language library that provides support for large, multi-dimensional arrays and matrices, along with a wide collection of high-level mathematical functions to work on those arrays. NumPy's predecessor, Numeric, was originally created with contributions from several other developers, by Jim Hugunin. In 2005, Travis Oliphant developed NumPy with substantial changes, integrating features of the competing Numarray into Numeric. NumPy is open-source software and has many contributors. NumPy is a package that describes an object with multi-dimensional array and associates fast math functions that work on it. This also offers simple linear algebra routines and an fft and sophisticated random number generation. NumPy replaces Numeric and Numarray, respectively.

IV. PROPOSED SOLUTION

When image quality is taken into consideration, there is a plethora of aspects that influence the system's correctness. It is significantly important to apply distinct image pre-processing techniques to standardize the images that you supply to a face recognition system. [2] Most face recognition algorithms are very responsive to lighting conditions, so that if it was trained to detect an individual when they are in a dark room, it probably won't recognize them in a bright room, etc. This problem is mentioned to as "lumination dependent", and there are also many other issues, such as the face should be in a very steady position within the images (such as the pixel coordinates of eyes have to be same), compatible size, rotation angle, hair and makeup, emotion (smiling, angry, sad, etc.), position of lights (to the left or right, above, etc.). This is why it is foremost to use a good image pre-processing filters before applying face recognition. You should also do things like removing the pixels around the face

which are not in used, such as with an elliptical mask that shows the inner face region, not the image background and hairs, since they modify more than the face does. For clarity, the face recognition system presented in this paper is [1] Eigen faces with the aid of grayscale images. The paper assert how easily is to convert color images to grayscale, and then to apply Histogram Equalization as a very easy method of automatically standardizing the brightness and contrast of the facial images. For better output, you could utilize color face recognition, or employ more processing stages such as edge enhancement, contour detection, motion detection, etc. Also, this code rescales images to a standard size, but this may change the aspect ratio of the face. OpenCV utilize a type of face detector called a Haar Cascade classifier. An image, which can come from a real time video or from file, the face detector examines each image position and classifies it as "Face" or "Not Face." Classification assumes a stiff scale for the face, say 50x50 pixels. Since faces in an image might be larger or smaller than this, the classifier runs over the image multiple times, to search for faces across a range of particular scales.

V. IMPLEMENTATION

We have used haar-cascade predefined classifiers for the face detection. There are different types of classifiers available

- haarcascade_frontalface_alt.xml classifier
- haarcascade_frontalface_alt.xml classifier
- haarcascade_fullbody.xml classifier
- haarcascade_lefteye_2splits.xml classifier
- haarcascade_righteye_2splits.xml classifier
- haarcascade_smile.xml classifier

V.1 Conversion Of Color Image Into Gray-Scale Image

```
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

V.2 Face Cascade Formation

```
face_cascade=.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
recognizer= cv2.face.LBPHFaceRecognizer_create() #EigenFaceRecognizer_create()
recognizer.read("trainner.yml")
```

V.3 Data Training

```
for root, dirs, files in os.walk(image_dir):
  for file in files:
    if file.endswith("png") or file.endswith("jpg"):
path = os.path.join(root, file)

label=os.path.basename(root).replace(" ", "_").lower()
        if not label in label_ids:
        label_ids[label] = current_id
        current_id += 1
        id_ = label_ids[label]
        pil_image = Image.open(path).convert("L")  #grayscale
        size = (500,500)
        final_image = pil_image.resize(size,Image.ANTIALIAS)
        image_array = np.array(final_image, "uint8")

faces = face_cascade.detectMultiScale(image_array,scaleFactor=1.5,minNeighbors=5)

 for(x,y,w,h) in faces:
         roi = image_array[y:y+h, x:x+w]
                x_train.append(roi)
                y_labels.append(id_)
```

V.4 Image Recognition

```
while(True):
    #Capture frame-by-frames
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray , scaleFactor = 3.4 , minNeighbors = 4)
```

```
for (x, y, w, h) in faces:
        print(x, y, w, h)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y + h, x:x + w]
        eyes=eye_cascade.detectMultiScale(roi_gray)
        for(ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
        id_, conf = recognizer.predict(roi_gray)
        if conf >=45:
            print(conf)
            font = cv2.FONT_HERSHEY_SIMPLEX   name = labels[id_]
            color = (255, 255, 255)
            stroke =2
            cv2.putText(frame , name, (x,y), font, 1, color, stroke, cv2.LINE_AA)

image item = "myimage.png"
cv2.imwrite(img item,roi gray)
```

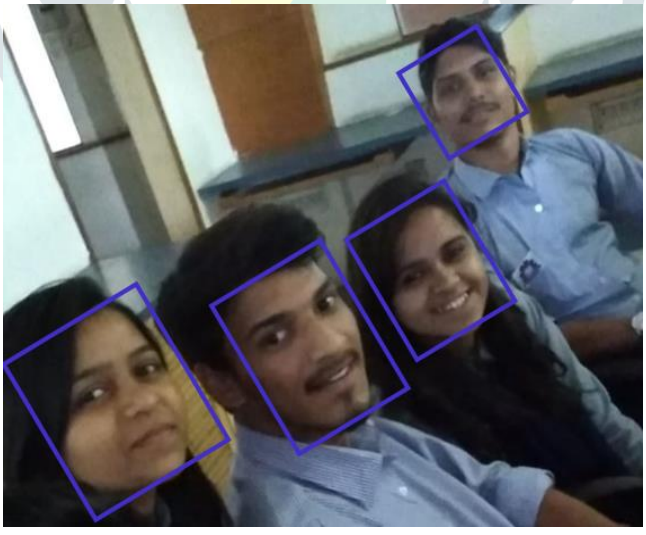
VI. RESULT

Whenever we send the suspected face to the device, it upgrades its data model with the provided face and tries to identify other faces matching the provided face so that it can automatically improve its models leading to correct facial recognition and automated facial identification takes place in live streaming videos or uploaded photographs.

Images as Data set	Accurately Detected Images	Inaccurately Detected Images	Accuracy (Detection Rate) %
Anup Katre (Dataset1)	46	09	91.69 %
Anup Katre (Dataset2)	55	01	97 %

TABLE 1: Result dataset accuracy

The table above shows the accuracy of the proposed model, and also shows the amount of correct and incorrect face-detected images given as the sample data collection.As, if we have all data models as correctly observed images, we can see it gives the highest precision.



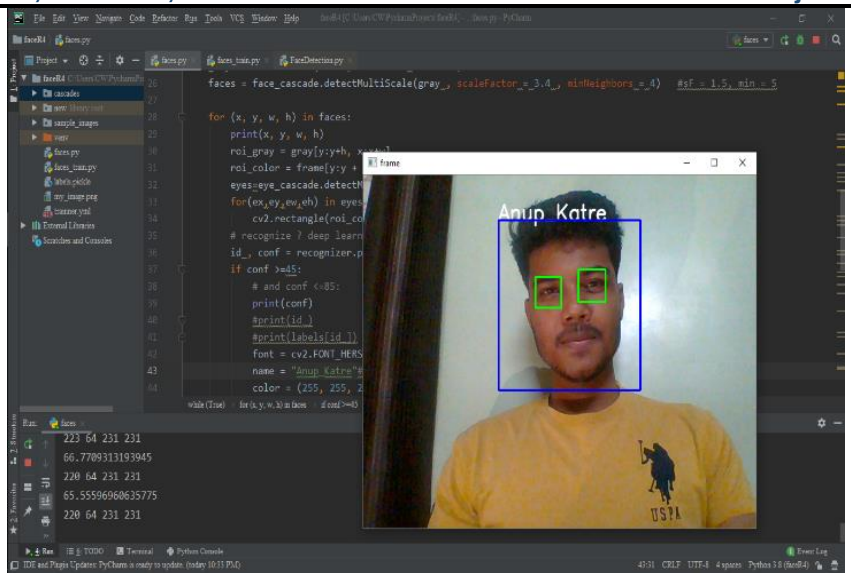


FIGURE 9: Image showing Detection and Recognition of face

VII. CONCLUSION

Face recognition continues to be a difficult topic in computer vision. Owing to its many applications in different domains, it has received a lot of attention over the last few years. Although there is considerable research effort in this field, facial recognition systems are far from ideal for proper performance in all real-world situations. Paper offered a brief survey of face-recognition problem methods and implementations. In order to realize methods that represent how people recognize faces and make effective use of the temporal evolution of the face's appearance for identification, there is much work to be done.

VIII. REFERENCES

[1] M.A.Turk and A.P. Pentaland, “Face Recognition Using Eigenfaces”, IEEE conf. on Computer Vision and Pattern Recognition, pp. 586-591, 1991.
[2] Ahonen, Timo, et al. "Recognition of blurred faces using local phase quantization." Pattern Recognition, 2008. ICPR 2008.19th International Conference on. IEEE 2008.
[3] CH. Y. Lu, CH.SH. Zhang & F. Wen (1999), “Regional Feature based Fast Human Face Detection”, J. Tsinghua Univ. (Sci. and Tech.), China, Vol. 39, Pp. 101–105. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[4] H. J. Jiang (2007), “Research on Household AntiTheft System based on Face Recognition Technology”, Nanjing University of Aeronautics and AstronauticsChina.
[5] https://www.youtube.com/watch?list=PLQVvvaa0QuDfKTOs3Keq_kaG2P55YRn5v&v=OGxgnH8y2NM
[6] https://www.youtube.com/watch?list=PLQVvvaa0Q uDfKTOs3Keq_kaG2P55YRn5v&v=OGxgnH8y2NM
[7] https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face__detection.html
[8] https://pythonprogramming.net/haar-cascade-objectdetection-python-opencv-tutorial/
[9] Open Source Computer Vision Library Reference Manual-Intel [Media]
[10] https://docs.opencv.org/3.4/db/d28/tutorial_cascadeclassifier.html
[11] https://www.pyimagesearch.com/2016/06/20/detecting-cats-in-images-with-opencv/
[12] https://github.com/krishnaik06/Computer-Vision-Tutorial