International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Smart Surveillance System using Adaboost Algorithm

Prof. Punesh Tembhare [1], Anup Katre[2], Dhanashri Adgurwar[3], Sharayu Bagmare[4], Vaibhav Rahangdale[5], Sanket Harinkhede[6]

[1]Professor Punesh Tembhare, Dept. of Computer Technology, Priyadarshini College of Engineering Nagpur, Maharashtra, India
[2]Anup Katre, Dept. of Computer Technology, Priyadarshini College of Engineering, Maharashtra Nagpur, India
[3]Dhanashri Adgurwar, Dept. of Computer Technology, Priyadarshini College of Engineering Nagpur, Maharashtra, India
[4]Sharayu Bagmare, Dept. of Computer Technology, Priyadarshini College of Engineering Nagpur, Maharashtra, India
[5]Vaibhav Rahangdale, Dept. of Computer Technology, Priyadarshini College of Engineering, Maharashtra Nagpur, India
[6]Sanket Harinkhede, Dept. of Computer Technology, Priyadarshini College of Engineering, Maharashtra Nagpur, India

---

Abstract - Recognition of face is one of the biometric techniques implemented for recognition of any facial image. Since birth, faces play a crucial role in any individual's social interaction which gives that individual a distinctive identity. Face detection and recognition is a very popular topic in biometric research as it is used for security and safety of an individual. Face recognition technology has widely attracted the researchers, due to its wide range of application value as well as market potentials, such as fraud detection and video surveillance. Face recognition performs a very crucial role in the surveillance system as there is no need for 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 4] recognition module.

Key Words: **Face detection, Face Recognition, OpenCV.**

1. INTRODUCTION

Face recognition technology was commonly observed as a something straight out of science fiction. But over the past decade, this groundbreaking technology has not just become viable, but also it has become widespread. 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. Face detection and recognition is technology which is used to recognize an individual from a video or photo source. The pioneers of facial recognition were Woodrow Wilson Bledsoe, Helen Chan Wolf and Charls Bisson. In the 1960s the concept of face recognition was introduced by Woodrow Wilson Bledsoe. Bledsoe developed a system called as RAND tablet that could classify photos of faces by hand. RAND tablet is a device that people could use to input horizontal and vertical coordinates on a grid using a stylus that emitted electromagnetic pulses. The system could be used to record manually the coordinate locations of various facial features including the eyes, nose, hairline and mouth. These set of data could then be inserted in a database.Then, when the system was given a new image of an individual, it was able to retrieve the image from the database that most closely resembled to that individual. Ever since then recognition system is being improved and optimized constantly, the technology becomes gradually mature and is more and more widely used in human daily life. There are several industries benefiting from this technology. Law enforcement agencies are using face recognition technology to keep communities safer from impostors. Retailers are preventing crime and violence. Airports are improving traveler's convenience and security. Nowadays mobile phone companies are using face recognition technology to provide consumers with new layers of biometric security. In this paper, we propose a face detection and recognition system with the aid of python along with OpenCV. This system contains three modules which are detection module, training Module and recognition module. The detection module recognizes the face which gets into the field of vision of the camera and saves the face in the form of an image in JPG format. Then the training modules trains the system with the aid of Haar cascade algorithm which was proposed by Paul Viola and Michael Jones.

1.1 Haar Features Selection

First step is to collect the Haar Features. A Haar feature considers adjacent rectangular areas at a specific position in a detection window, which adds the pixel intensities in each area and calculates the difference between these additions.

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

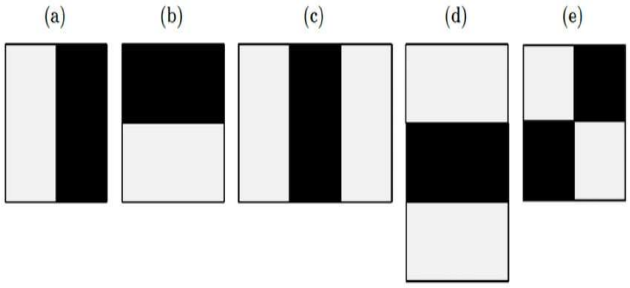

Fig :- Haar features

1.2 Creating Integral Image

To make this procedure simple, Integral Images are used. Most of the features measured are of no significance. The Integral Image is used as a simple and accurate way to measure the number of values (pixel values) in a given image-or a rectangular subset of a grid. It can also, or mostly, be used to determine the average intensity within a given image.

When one chooses to use the Integral Image, it's generally a good idea to make sure first of all that the image is in grayscale. We need to measure the sum of the pixel values included in several rectangular regions of the image to remove these haar features. We need to measure these areas at various scales (that is, for different rectangle sizes) to make it invariant in dimension.

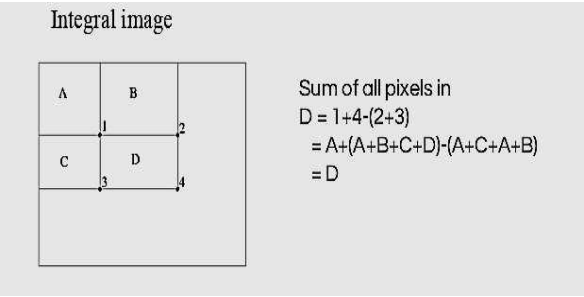

Fig 2:- Integral Image

1.3 Adaboost Training Face Detection

Adaboost is an effective boosting algorithm combining simple statistical learners thus significantly reducing both the training error and the more elusive generalization error. For that very reason Adaboost is used. It selects the few necessary features that when combined / amalgamated provide a classifier that is effective for classifying the facial / required object in an image. The fact that Adaboost is adaptive in nature is what makes it relevant in various scenarios.

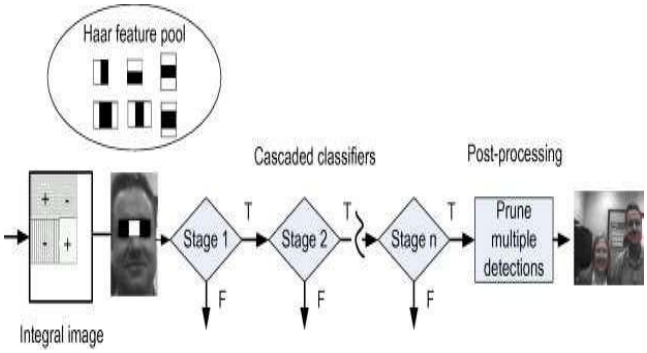

Fig 3:- Adaboost Algorithm

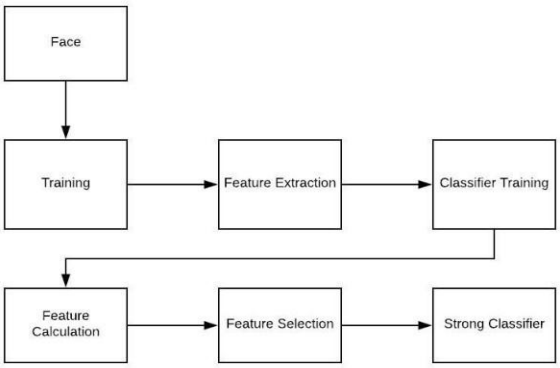

Fig 4:- Several Phases of cascade Classifier.

1.4 Cascading Classifier

The cascade classifier consists of several phases, where each phase is a group of weak learners. Such slow learners are basic classificatory known as stumps for decision. Each stage is trained using a Boosting technique. Boosting offers the opportunity to train a highly accurate classifier by taking the weighted average of decisions that the poor learners produce.

Finally, the fundamental facial components from the new video are extracted in the recognition module. Then these features will be compared with the list of things stored during training and those with the best match will be identified and the known person's name will be displayed. This control system fulfills the basic needs of the face detection and recognition system, and also takes the expense into account to ensure the most economical universal mode possible. This can also be paired with real-time computational algorithms, in turn.

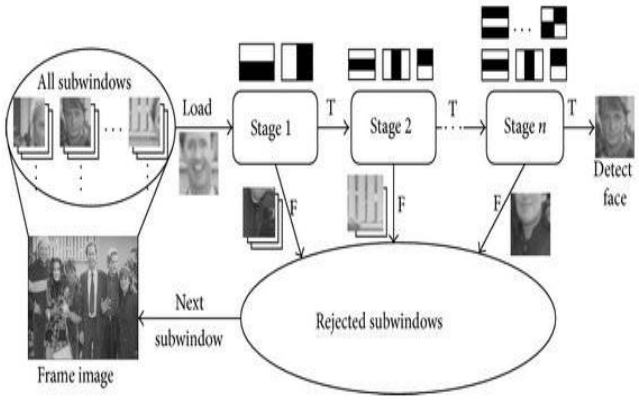

Fig 5:- Stagers of Cascade Classifiers

Any negative outcome at a certain stage leads to the sub-window being rejected as possibly having a face. At low computational cost, the initial classifier removes most negative examples, and the following classifiers remove additional negative examples, which require more computational effort.

This cascade is made up of classifiers. It functions in a way that the initial classifiers are simpler and are used to reject the majority of sub-windows, and to achieve low false positives at the end of complex classifiers. [9] Classifiers are trained using the aforementioned Adaboost Algorithms theory. The deeper we go into the cascade the harder it's for the classifier. The cascade classifier consists of several phases, where each phase is a group of weak learners.

2. PROBLEM DEFINATION

2.1 Face Detection

Face detection is an AI-based computer technology which can recognize and locate human faces in digital images and videos. Applications for face recognition use algorithms that decide whether images are positive (i.e. images with a face) or negative (i.e. images without a face). The algorithms have to be trained on large datasets containing hundreds of thousands of face images and non-face images to be able to do this accurately.

This device is capable of identifying the faces of the image taken from HD Video to analyze and identify the faces. 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.

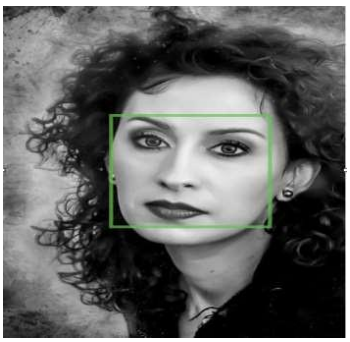

Fig 6:- Face Dection

2.2 Gray Scale Image

For image-related activity we use gray-scale image since the gray-scale image is a one-layer image from 0-255, while the colored image has three different layers, color often increases the model's complexity.

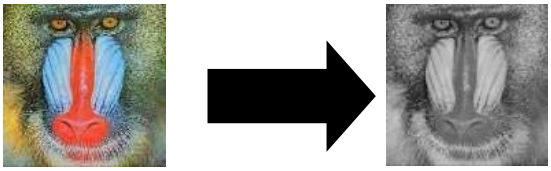

Fig 7:- Gray Scale Image Conversion

2.3 Feature Extraction

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.

Fig 8:- Haar features Extraction

2.4 Model Training

Simply put, model training is the process of taking several face representations that belong to the same person, and constructing a classifier that can identify further face depiction instances from the same person. A linear SVM is the most preferred solution.

2.5 Face Recognition

The last step after each face patch is formulated in the above steps is to identify the identities of those faces. A facet database is needed to construct for face recognition. Some photos are taken for each person, and their features are deleted and stored in the database. Whenever an input image arrives, we perform face detection and face extraction and compare the features of each face with the stored database after this. We distinguish two facial recognition applications: identification, and verification. Face identification means that the face image is given, we want the system to say who it is and identify the person while in face verification, given a face image and on the basis of assumption from the identification, we want the system to tell true or false about the guess.

3. DISCRIPTION OF TOOLS

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

3.1 OPENCV

Open Source Computer Vision Library is widely known as OpenCV. It is a library of programming methods and mainly focused at the real time computer vision which is developed by Intel. The library is cross platform and it mainly aims at real-time image processing. Originally the library is written in C and this C interface makes OpenCV portable to some distinct platforms such as digital signal processors. Wrappers for languages such as C#, Python, Ruby and Java (using JavaCV) have been developed to uplift adoption by a wider user. However, since version 2.0, OpenCV includes both its traditional C interface and a new advance C++ interface. This new interface seeks to eliminate the number of lines of code essential to code up vision functionality as well as eliminate common programming errors such as memory leaks (through automatic data allocation and deallocation) that can occur when using OpenCV in C. The majority of developments and algorithms in OpenCV are now developed in the C++ interface. [9]It is much tougher to provide wrappers in other languages to C++ code as opposed to C code; since the other language wrappers are generally lacking some latest OpenCV 2.0 features.

3.2 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

4. PRPOSED SOLUTON

While evaluating the picture quality, there are a multitude of factors that affect the correctness of the scheme. Applying distinct image pre-processing techniques to standardize the images you send to a facial recognition system is of considerable importance. [2] Most face recognition algorithms are very sensitive to lighting conditions, so if a person has been trained to identify them in a dark room, they probably won't recognize them in a bright room. This problem is referred to as "lumination based," and there are several other issues as well, such as the face should be in a very stable place inside the images (such as the pixel coordinates of the eyes must be the same), consistent height, angle of rotation, hair and make- up, emotion (smiling, angry, sad, etc.), light location (left or right, above, etc.). For this reason, You can also do things like remove the pixels around the face that aren't in use, such as in an elliptical mask displaying the region of the inner face, not the background image and hairs, because they change more than the face does. The face recognition device described in this paper is [1] Eigene faces with the aid of grayscale images, for clarification. The paper asserts how easy it is to convert color images to grayscale, and then apply Histogram Equalization as a very simple method to standardize the brightness and contrast of facial images automatically. You may use color face recognition for better performance, or employ more processing stages such as edge enhancement, contour detection, motion detection, etc. This code often rescues images to a regular scale, but that will adjust the facial aspect ratio. OpenCV uses a face- detector system called a Haar Cascade classifier. The face detector analyses of image location and classifies it as "Face" or "Not Face," an image that may come from a real-time video or picture. Classification assumes a rigid face scale, say 50x50 pixels. As faces in an image can be larger or smaller than this, the classifier runs several times over the image to look for faces across a frame.

5. IMPLIMENTATION

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

- *haarcascade_frontalface_alt.xml (classifier)*
- *haarcascade_fullbody.xml (classifier)*
- *haarcascade_lefteye_2splits.xml (classifier)*
- *haarcascade_righteye_2splits.xml (classifier)*

5.1 Face Detection

Face detection using Haar Cascades is an approach based on machine learning, where a cascade function is trained with a collection of input data. OpenCV already includes several pre-trained facial, hair, smile, etc. classifiers.

```python
import cv2
import os
def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

vid_cam = cv2.VideoCapture(0)

face_detector = cv2.CascadeClassifier("F:\\github"
    "\\Face-Recognition-master\\haarcascade_frontalface_default.xml")

face_id = 5 ##change every time and run again for different user
count = 0

assure_path_exists("F:\\github\\Face-Recognition-master\\dataSet\\")

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:
        cv2.rectangle(image_frame, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1
        cv2.imwrite("F:\\github\\Face-Recognition-master\\dataSet\\User."
        + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
        cv2.imshow('frame', image_frame)

    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    elif count>100:
        break
vid_cam.release()
cv2.destroyAllWindows()
```

Face Detection and Dataset Formation Code in Python

5.2 Dataset Model Training

Usually, this process is referred to as facial recognition enrolment. We name it "enrollment" as we "enroll" and "register" the user in our dataset and application as an example person In order to create a set of training data from the available image collection, you will need to determine the set of requirements that are required to qualify for an image, and then you will need to manually pick images that meet these criteria.

```python
import cv2, os
import numpy as np
from PIL import Image
import os

def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("F:\\github\\Face-Recognition-master"
                                 "\\haarcascade_frontalface_default.xml")

def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L')
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return faceSamples,ids

faces,ids = getImagesAndLabels("F:\\github\\Face-Recognition-master"
                               "\\dataSet\\")
recognizer.train(faces, np.array(ids))
assure_path_exists('F:\\github\\Face-Recognition-master\\trainer\\')
recognizer.save("F:\\github\\Face-Recognition-master"
                "\\trainer\\trainer.yml")
```

Dataset Training Code in Python

5.3 Final Face Recognition

```
import cv2
import numpy as np
import os

def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

recognizer = cv2.face.LBPHFaceRecognizer_create()

assure_path_exists("F:\\github\\Face-Recognition-master\\trainer\\")
recognizer.read("F:\\github\\Face-Recognition-master"
                "\\trainer\\trainer.yml")

cascadePath = "F:\\github\\Face-Recognition-master" \
              "\\haarcascade_frontalface_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX

cam = cv2.VideoCapture(0)

while True:
    ret, im =cam.read()
    grayim = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(grayim, 1.2,5)
    for(x,y,w,h) in faces:
        cv2.rectangle(im, (x-20,y-20), (x+w+20,y+h+20), (0,255,0), 4)
        Id, confidence = recognizer.predict(grayim[y:y+h,x:x+w])
        if(Id == 1):
            Id = "User1{0:.2f}%".format(round(100 - confidence, 2))
        elif(Id == 2):
            Id = "User2{0:.2f}%".format(round(100 - confidence, 2))
        elif (Id == 3):
            Id = "User3{0:.2f}%".format(round(100 - confidence, 2))
        elif (Id == 4):
            Id = "User4{0:.2f}%".format(round(100 - confidence, 2))
        cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)
        cv2.putText(im, str(Id), (x,y-40), font, 1, (255,255,255), 3)
    cv2.imshow('im',im)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

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

Face Recognition Using Python

6. RESULT

Whenever we provide the suspected face to the module, it update its data model with the given face and try to find another faces resembles to the given face so that it can automatically enhance its models which leads to accurate recognition of the face and then in live streamed videos or uploaded images the automatic detection of the faces takes place and it recognizes the face according to the trained model and hence we get the output.

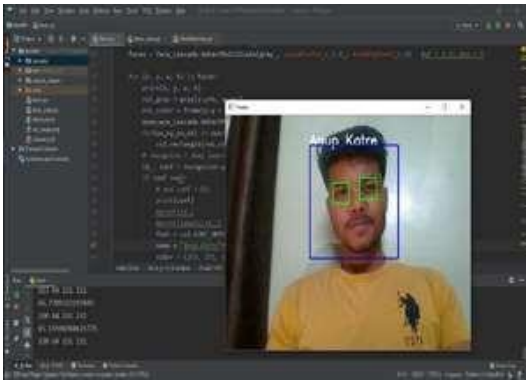

Fig :- Obtained Result

Table -1: DATASET TABLE

Sample Output Table			
Images as Data set	Accuracy Rating	Accuracy Rating after modification	Result in %
Anup kater (Data set 1)	09	52	52%
Dhanashri Adgurwar (Data set 2)	11	62	62%
Sharayu Bagmare (Data set 3)	17	63	63%
Sanket Harinkhede (Data set 4)	8	47	47%

7. 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.

REFERENCES

[1] M.A.Turk and A.P. Pentaland, "Face Recognition Using Eigen faces", 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.),
[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=PLQVvvaa0Q uDfKTOs3Keq_ 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://www.pyimagesearch.com/2016/06/20/ detecting-cats-in-images- with-opencv/
[11] https://github.com/krishnaik06/Computer-Vision-Tutorial
[12] Ahonen, Timo, et al. "Recognition of blurred faces using local phase quantization." Pattern Recognition, 2008. ICPR 2008.19th International Conference on. IEEE 2008.
[13] 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.
[14] H. J. Jiang (2007), "Research on Household AntiTheft System based on Face Recognition Technology", Nanjing University of Aeronautics and AstronauticsChina.
[15] https://www.youtube.com/watch?list=PLQVvvaa 0QuDfKTOs3Keq_kaG2P55YRn5v&v=OGxgnH8y2NM
[16] https://www.youtube.com/watch?list=PLQVvvaa 0Q uDfKTOs3Keq_kaG2P55YRn5v&v=OGxgnH8y2NM
[17] https://docs.opencv.org/3.4/d7/d8b/tutorial_py _face_ detection.html
[18] https://pythonprogramming.net/haar-cascade-objectdetection-python-opencv-tutorial/
[19] Open Source Computer Vision Library Reference Manual-Intel [Media]
[20] https://docs.opencv.org/3.4/db/d28/tutorial_cascadeclassifier.html
[21] https://www.pyimagesearch.com/2016/06/20/detecting-cats-in-images-with-opencv/
[22] https://github.com/krishnaik06/Computer-Vision-Tutorial