DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SOUTHEAST UNIVERSITY

CSE4000: Research Methodology

Real-time Face Recognition System

A dissertation submitted to the southeast university in partial fulfillment of the requirement

For the degree B. Sc. in Computer Science & Engineering.

Author:

Name: Arif Hossain

ID: 2015000000028

Name: Umme Kulsum Sumana

ID: 2015100000024

Name: MD Riajul Islam Riaj

ID: 2015100000066

Supervisor:

Roksana Akter Jolly

Assistant Professor

Department of CSE

Southeast University

Letter of Transmittal

September 15, 2019

The Chairman,

Department of Computer Science & Engineering,

Southeast University,

Banani, Dhaka-1213

Through: Supervisor, Roksana Akter Jolly

Subject: Submission of Research Paper

Dear Sir,

With due respect, we have researched on “Real-time Face Recognition System” under the course, Research Methodology. We want to develop a face recognition-based security system.

So, we try our level best to complete our project. We have given our best effort to complete this research. We are requesting for your kind approval of this report. Hope you will appreciate our hard work and excuse the minor errors.

Thank you.

Sincerely yours,

Arif Hossain | 2015000000028

Umme Kulsum Sumana | 2015100000024

MD Riajul Islam Riaj | 2015100000066

Supervised by

Roksana Akter Jolly
Assistant Professor,
Department of CSE

Certificate

This is to certify that the research paper titled “Real-Time Face Recognition System” is the bona- fide record of research work done by Arif Hossain, Umme Kulsum Sumana, MD Riajul Islam Riaj for the partial fulfillment of the requirements for B.Sc. computer Science & Engineering (CSE) from Southeast University.

This paper was carried out under my supervision and is record of the bona-fide work carried out successfully.

Authors:

Approved by the Supervisor

Arif Hossain | 2015000000028

Roksana Akter Jolly
Assistant Professor, Department of CSE
Southeast University

Umme Kulsum Sumana | 2015100000024

MD Riajul Islam Riaj | 2015100000066

Acknowledgement

Thanks to almighty Allah for letting us to complete this research work successfully in time. We are highly indebted to our research supervisor, Roksana Akter Jolly, Assistant Professor Department of Computer Science Engineering, Southeast University for his guidance, supervision and sympathy during the entire research work.

We would like to express our gratitude all the teachers of the department for their help that has allowed us to reach this stage of life.

We would also like to offer our thanks to our friends for their help, inspiration and constructive criticism that have contributed to the successful completion of the work.

Finally, our sincerest gratitude is to our parents for their constant support and encouragement to complete this work.

Abstract

This research project describes a model for detecting and recognizing of human face real-time using Open CV platform. This happens in two stages, face detection and face recognition. To detect a face, we used a pre trained Haar cascade model which is based on Viola and Jones Rapid Object Detection algorithm. Using a Haar Cascade Classifier that is trained by both positive and negative image samples to detect faces. To recognize we used LBPH Face Recognizer which is reference class of OpenCV. OpenCv has the ability to train a recognizer.

The rate of accuracy we found is about 70 - 80 %.

Table of Content

Chapter 1 Introduction

1.1 What is Face Detection?

Computer technology that is capable of identifying the presence of human faces in digital images. Face detection means that there is a human face in still image or a video. It can also be used in auto-focus cameras and to count how many people entered a given area. There are numerous applications of face detection such as facial motion capture, facial recognition, photography, lip reading, emotional inference etc.

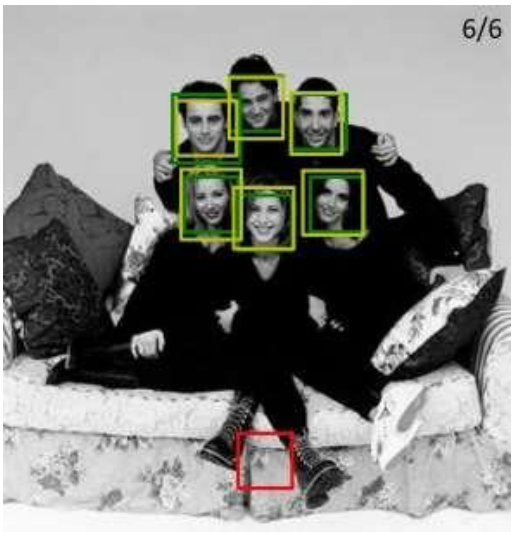


Fig 1: Face Detection

1.2 What is Face Recognition?

One of the most significant face detection apps is facial recognition. Face recognition defines a biometric technology that extends far beyond acknowledging the presence of a human face. In fact, it is trying to determine whose face it is. The method goes with a computer application capturing a digital image of the face of an individual (sometimes drawn from a video frame) and comparing it with images in a recorded record database. While facial identification is not 100% precise, it can determine very correctly when there is a powerful likelihood that the image of a person matches somebody in the database. There are many face recognition apps. For unlocking phones and particular apps, face recognition is already being used. For biometric surveillance, facial recognition is also used. Banks, retail shops, stadiums, airports, and other equipment use facial recognition for crime reduction and violence prevention.

Fig 2: Face Recognition

1.3 Face Detection Phase

The face detection contains three important phases, where first a face detection system is designed by giving some input as some faces and non faces. In this research project we detect the face region using Viola Jones Approach. A classifier or something that identifies face we train something using positive and negative faces, faces and non – faces and when the training is done with the data that we have got we would be able to detect any face in any incoming image. Training means to teach a computer what is a face and not a face. When the computer is trained it will extract certain features and certain details for the features. When all the feature and threshold are extracted during the training phase and stored in a file (XML), We take that file and take a newer input image then check the features from the file, apply it on the input an image and if it pass through all the feature and thresholds we classify that as a face or non - face.

So, for our research project we used Haar Cascade classifier.

```
face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
```

Lastly for recognizing we used LBPH algorithm. We created some folder with specific labels and inside there are some sample images for LBPH algorithm to learn and save the image's binary patterns to compare with the input video and show result.

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

Chapter 2 Detecting a Face

2.1 Haar Cascade

Haar Cascade classification is mainly the same as hair feature except they have a distinct manner to store those characteristics and those characteristics are called hair characteristics that we feed into a sequence of Cascade or a sequence of classifiers called a cascade of classifiers and then we use that to identify basically whether that item is present in the image or not. Hair Cascade classifiers are very great at identifying a specific item as an instance of this, but we can use them in parallel to identify faces and eyes, perhaps mouths, etc.

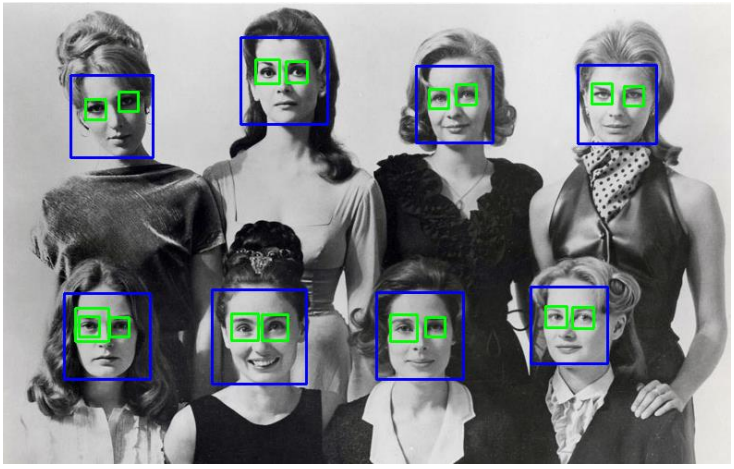


Fig 3: Parallel Identify

2.1.1 Training over Positive and Negative Images

We train the classifier by at least sometimes providing it a lot of images thousands or hundreds. We give it some positive images that can actually be less than the negative images. Maybe we're giving it 800 negative images with no faces and 400 positive images with a face. This is the first component of training gathered in those images by a Haar Cascade classifier. Once we have those images, we now have to extract our Haar features out of it.

Fig 4: **Negative Image** **Positive image**

2.1.2 Haar Features and selection

There are fixed set of types of Haar features,

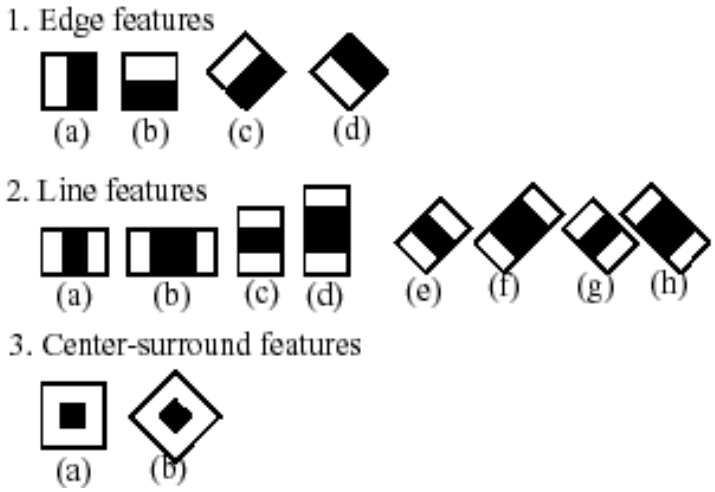


Fig 5: Haar features

Here in the edge feature (a) the white region is replaced by -1 and black region is replaced by +1. This is exactly like a combination kernel. For example if we look in edge feature (a) which is one row and two columns where left column is-1 and right column is +1.So if we apply this feature in an image we just have to sum the pixel value on the white region from the pixel value on the black region to get a specific value.

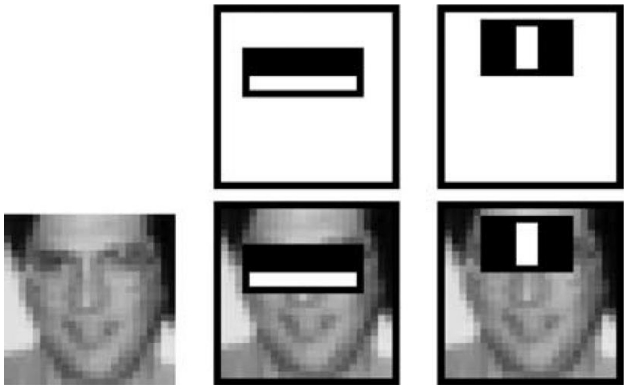


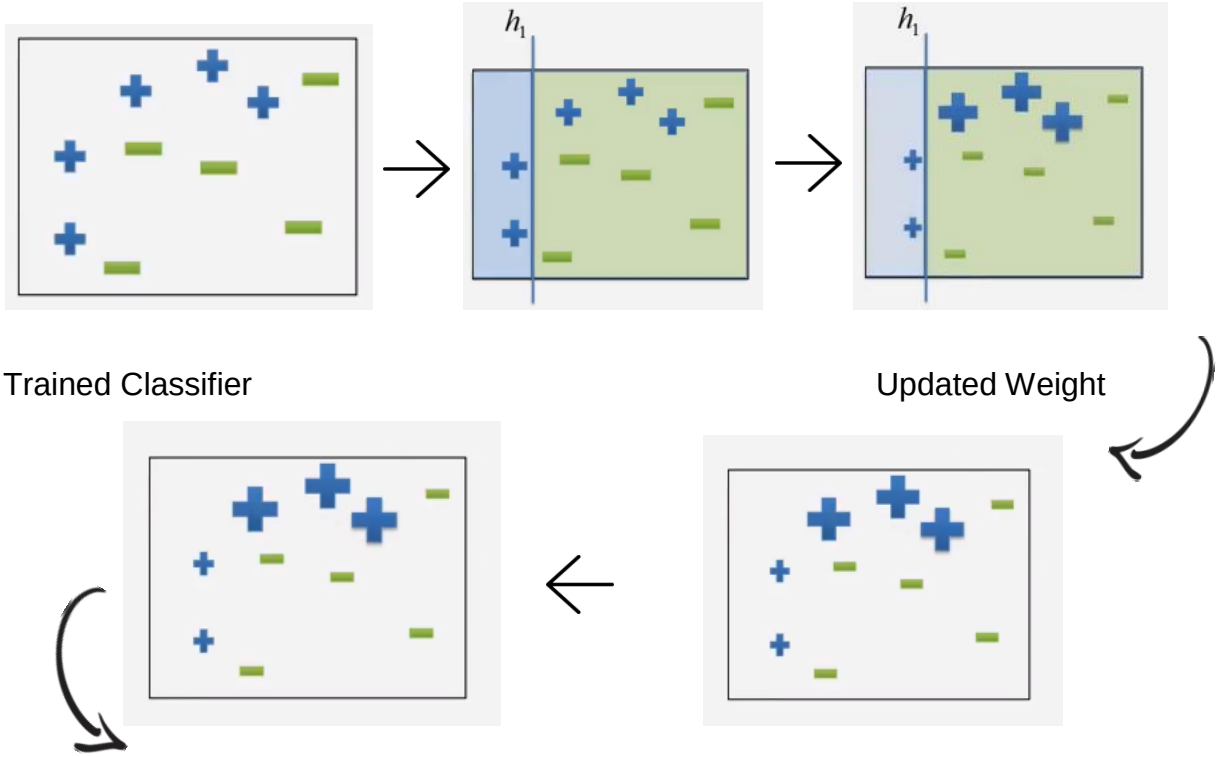
Fig 6: Haar Feature Applied

Viola Jones algorithm uses 24 x 24 pixel as the base window size to start evaluating these features in any given input source. If we consider all the possible parameters like scale, position there is almost 180,000 features generated. **Integral images** are used to make this procedure quicker. So to evaluate this huge set of features is practically impossible. So to eliminate the redundant features Boosting can be implemented with Freund and Schapire's **AdaBoost** Algorithm which selects the best features among them and from 180000 features now by Adaboost training the amount of features narrowed down to 6000 features but that is still not efficient enough for real time use because a large number of features are still left to process at once.

2.1.3 Boosting

Boosting uses ensemble techniques to create a sequence of increasingly complex predictors out of building blocks made out of very simple predictors boosting trains models by sequentially training a new simple model based on the errors of the previous model.

Original Dataset



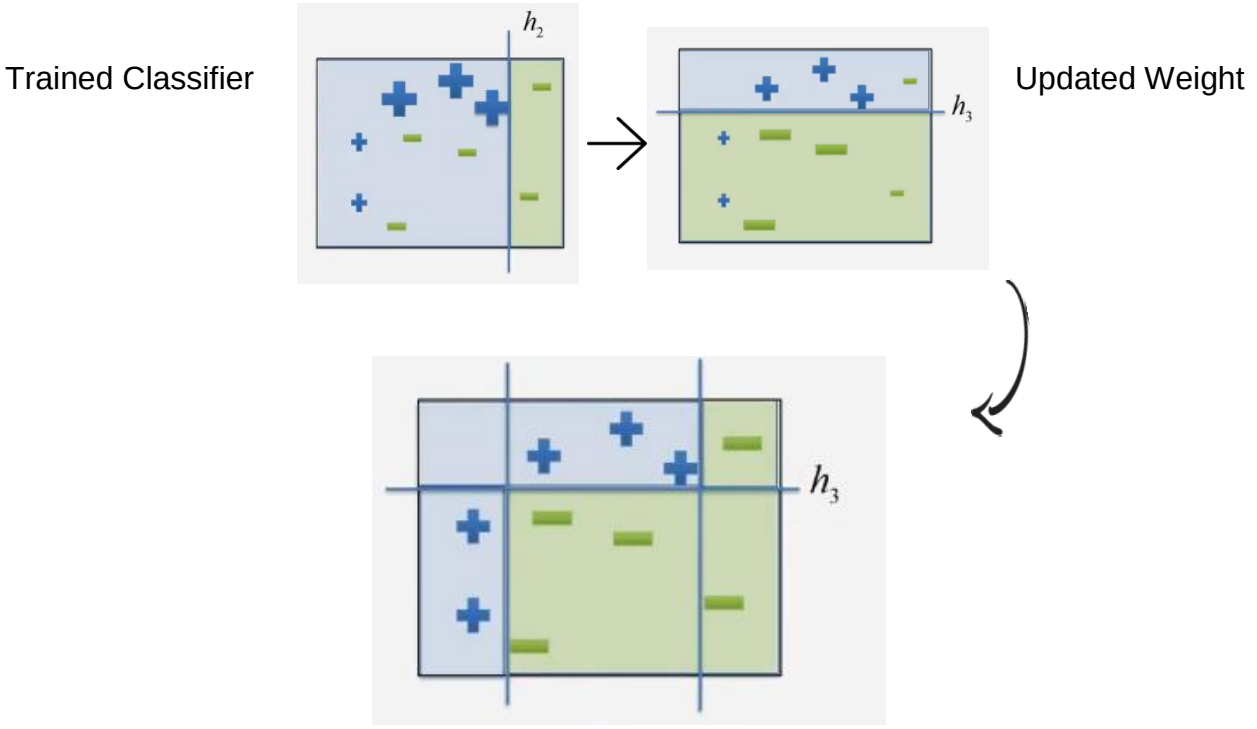


Fig 6: Boosting

We start off by learning a very simple predictor and then evaluate its errors and focus the next predictor on getting these examples right. This procedure tends to discover the examples and data points that are hard to predict and focuses later classifiers on predicting these examples better. In the end we combine the whole set uses some weighted combination and this is a way of scaling up complexity. Each individual predictor tends to be very simple and by combining many of these weak learners that are not able to learn complex functions we can convert them all into an overall much more complex classifier here for classification. Here this weighted final combination has managed to carve out a shape a region to predict plus one and minus one. The classic example of boosting for classification is the adaboost or adaptive boosting algorithm adaboost trains a series of models say n boost models sequentially using a weighted trainer so we can use any black-box machine learning algorithm that we would like as long as it weighted collection of training points and try to minimize the weighted error.

Viola Jones face detector the idea here is that we can well again combine a collection of many very weak and easy to compute classifiers in particular Viola Jones uses decision stumps with threshold some single feature or lots and lots of features but using very simple classifiers of those features so the number of features will make us prone to over fitting the choice of an extremely simple classifier function will make us less prone to over fitting and then we build up a more complex function and fit well so the idea of face detection is looks at a single patch of an image and decides whether or not that image patch has a face.

2.1.4 Haar Cascade Classifier

So, to show result quickly and real time, we use the system called a cascade of classifiers for face detection. A Haar Cascade is actually a classifier used to identify from the source the object for which it was trained. The Haar Cascade is trained by overlaying the positive image over a set of negative images. Training is typically performed on a server and at various stages. The Viola Jones approach used 38 stages.

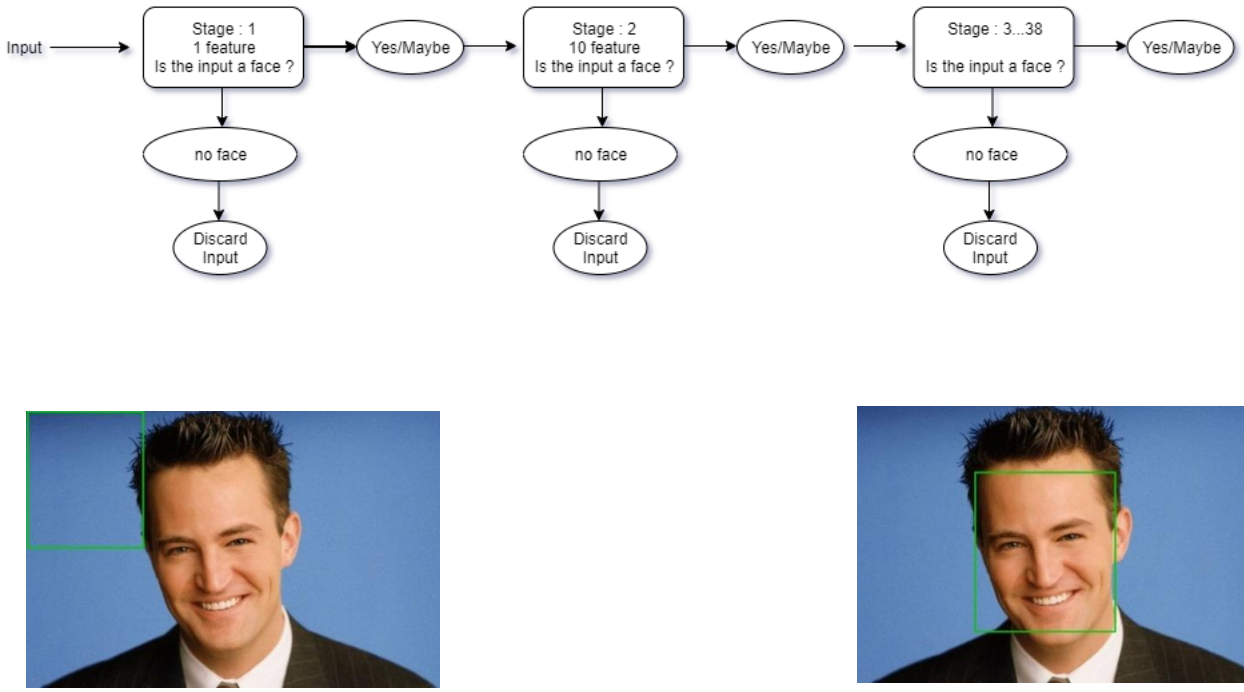


Fig 7: Haar Cascade working stages

Chapter 3 Face Recognizer

For recognition we used the LBPHFaceRecognizerCreate () which is class reference in OpenCV. Local Binary Pattern (LPB) defines a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. LBPH stands for Local Binary Pattern Histogram which gathers statistics of LBP event in a form of Histogram.

3.1 LBP (Local Binary Pattern)

Local Binary Pattern looks at 9 pixels at a time. So, it is a block of 3x3, 9 pixels and it particularly interested in the central pixel. The LBP turns this set of blocks in a single value. It will do this by comparing every neighboring pixel with the central pixel and if the neighbor value is greater than or equal to the center, we will assign 1 and if less than or equal we will assign 0. Example is as below,

13	12	18
8	**9**	5
10	2	1

1	1	1
0		0
1	0	0

So now all these bits we got are now going to turn into a byte

1	1	1	0	0	0	1	0

The decimal of this Binary Pattern 11100010 is 226.

LBP has to level of encoding

1) 256-level encoding

2) 59-level encoding

The decimal value we got is 256 level encoding that needs to encode in 59-level encoding to reduce encoding redundancy. The way to reduce the encoding redundancy is by assigning uniform LBP with unique index

LBP	Uniform?	Bin Index
0	Yes	0
1	Yes	1
2	Yes	2
3	Yes	3
4	Yes	4
5	No	58
6	Yes	5
7	Yes	6

LBP	Uniform?	Bin Index
8	Yes	7
9	No	58
10	No	58
11	No	58
12	Yes	8
13	No	58
14	Yes	9
15	Yes	10

LBP	Uniform?	Bin Index
248	Yes	51
249	Yes	52
250	No	58
251	Yes	53
252	Yes	54
253	Yes	55
254	Yes	56
255	Yes	57

Fig 8: Non-Uniform Pattern Handling

Now let's take an image to encode every point as a pattern which is the 1st stage,

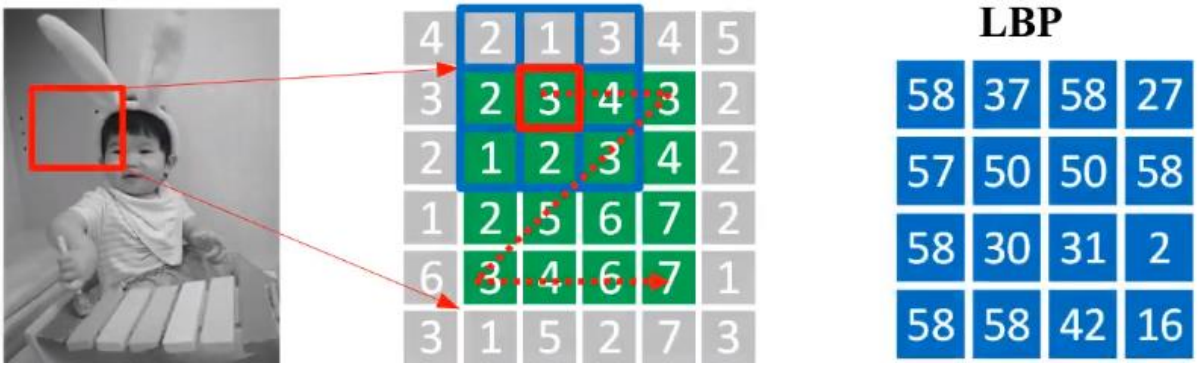


Fig 9: Encoding points as LBP

In stage two we gather the LBP in the form of Histogram.

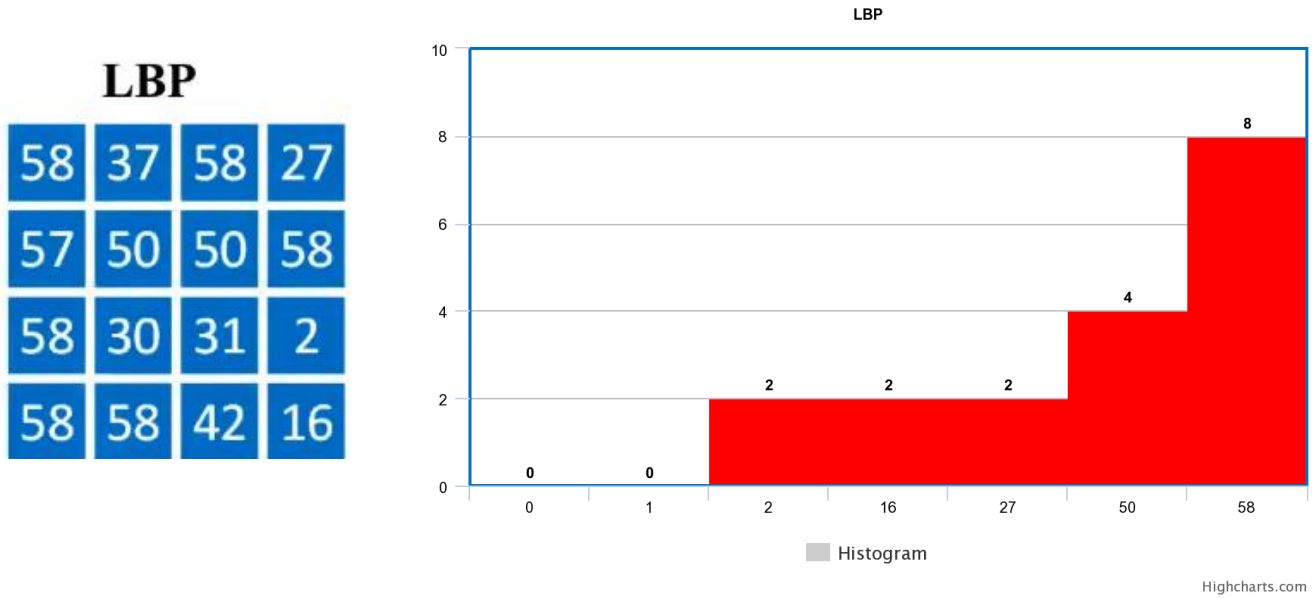


Fig 10: Histogram

3.2 Confidence

In the shape of a probability or confidence rating, both face detection and face recognition systems can provide an assessment of the forecast confidence level. A face detection scheme, for instance, can estimate that a picture area is a face with a 90% confidence score, and another picture area is a face with a 60% confidence score. It should be more probable that the region with the greater confidence rating will have a face. If a face detection scheme does not correctly detect a face or gives a forecast of poor confidence of a real face, it is regarded as a missed detection or false negative. Confidence ratings are a critical element of facial recognition and detection schemes. These schemes predict whether a face occurs in a picture or matches a face in a different picture with a corresponding degree of confidence in the forecast.

Chapter 4 Tools

4.1 Hardware Requirement

a) A PC with Windows (preferred) operating system.

b) RAM is greater than or equivalent to 4 GB

c) Intel® Core™ 2 Duo processor E8400 or higher

d) Secondary Memory at least 10 GB

e) A camera

4.2 Software Requirement

a) Python (3.4 or higher)

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

b) OpenCV

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

c) Windows Power Shell

PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language.

Chapter 5 Implementation

5.1 Copy Haar Cascade

We copied the data folder that is in OpenCV source folder to our Project folder. In the data folder there is different Haar Cascades and the one we use is haarcascade_frontalface_alt2

5.2 Using a face classifier

Here the face cascade that we copied now we use it to detect the faces in the frame (ret, frame = cap.read) Then we converted the frame to gray in order to make Haar cascade work. After that declare the scaleFactor (to improve predictor accuracy) and min neighbors.

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    gray  = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    # Display the resulting frame
    cv2.imshow('frame',frame)
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

5.3 Drawing a rectangle over face region

This step we draw a rectangle over the face region that we detected earlier. We declared the color and stroke value of the rectangle. Cord x = x+w, Cord y = y+h means how big the x and y co-ordinate will be for the rectangle (w = weight h = Height).

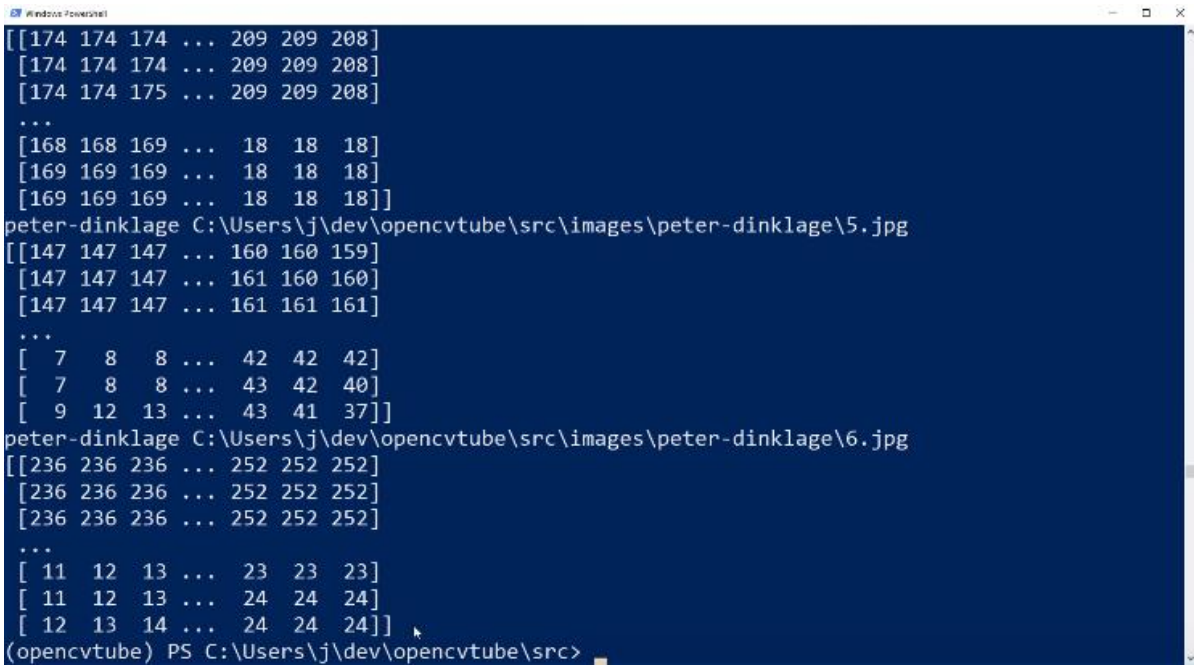
```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    gray  = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    for (x, y, w, h) in faces:
        print(x,y,w,h)
        roi_gray = gray[y:y+h, x:x+w] #(ycord_start, ycord_end)
        roi_color = frame[y:y+h, x:x+w]
        img_item = "my-image.png"
        cv2.imwrite(img_item, roi_gray)

        color = (255, 0, 0) #BGR 0-255
        stroke = 2
        end_cord_x = x + w
        end_cord_y = y + h
        cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke)
```

5.4 Training image into numpy arrays

Pillow or PIL is the python image library. Converting th image into gray by ‘Convert (“L”) ‘. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. With numpy we are turning the images into a numpy array, uint8 is a data type which is used unsigned 8-bit integer. And that is the range of pixel.

```
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()
            print(label, path)
            #y_labels.append(label) # some number
            #x_train.append(path) # verify this image, turn into a NUMPY arrray, GRAY
            pil_image = Image.open(path).convert("L") # grayscale
            image_array = np.array(pil_image, "uint8")
            print(image_array)
```

This is the numPy array now. This is taking every pixel value and turning it into numpy array. Every pixel here is now numbers. Why we did this because later we need to implement a recognizer class and in order to train that recognizer, we need to convert the images pixel into numbers.

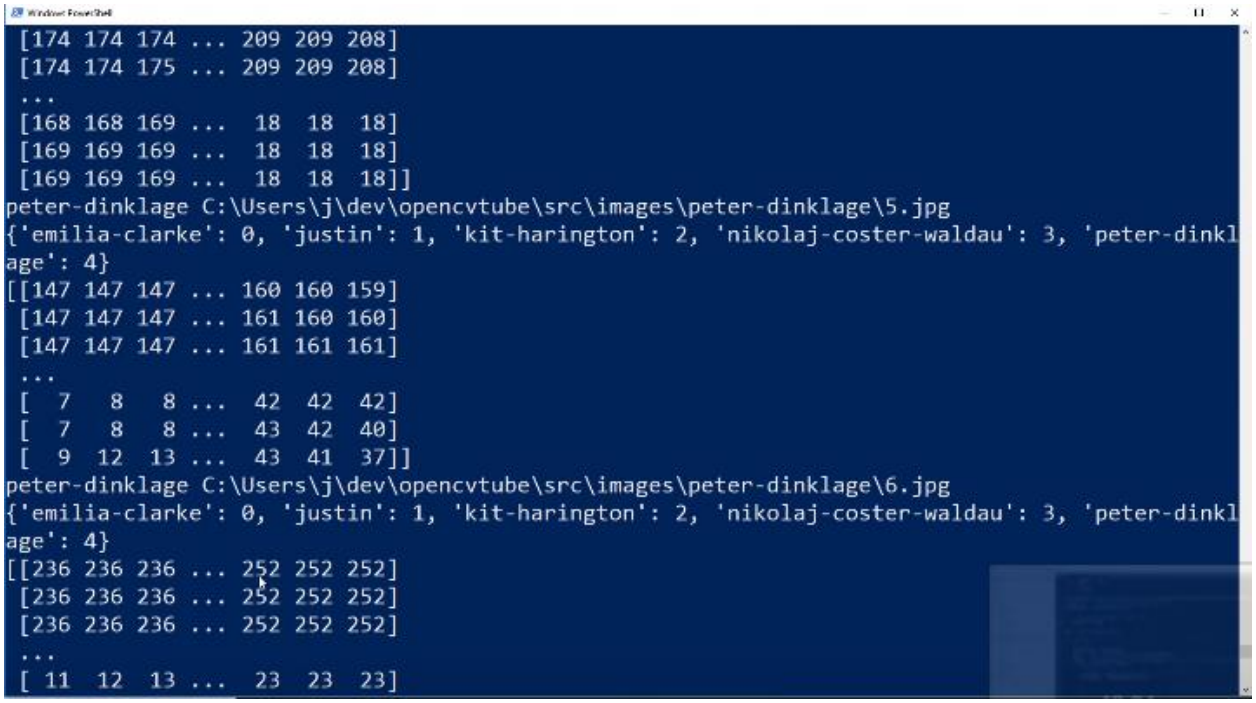


5.5 Region of interest in training data

```
y_labels = []
x_train = []

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()
            print(label, path)
            #y_labels.append(label) # some number
            #x_train.append(path) # verify this image, turn into a NUMPY arrray, GRAY
            pil_image = Image.open(path).convert("L") # grayscale
            image_array = np.array(pil_image, "uint8")
            print(image_array)
            faces = face_cascade.detectMultiScale(image_array, scaleFactor=1.5, minNei

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

5.6 Creating training labels

Here we associate number values with label ids.



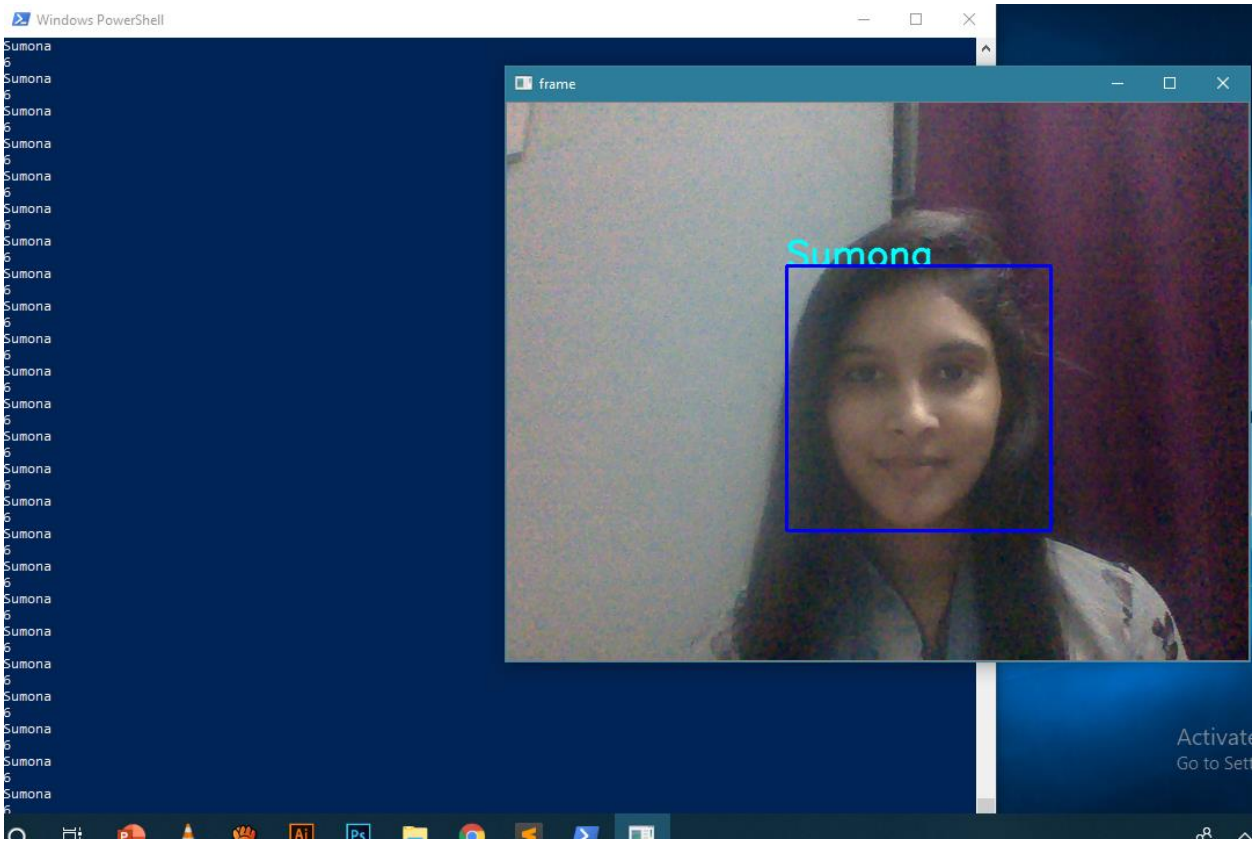
5.7 Training & implementing Recognizer

We used LBPH face recognizer to recognize the faces that we detected earlier. Here the. yml is for using trained file to predict.So, we used the roi and labels for the training here.

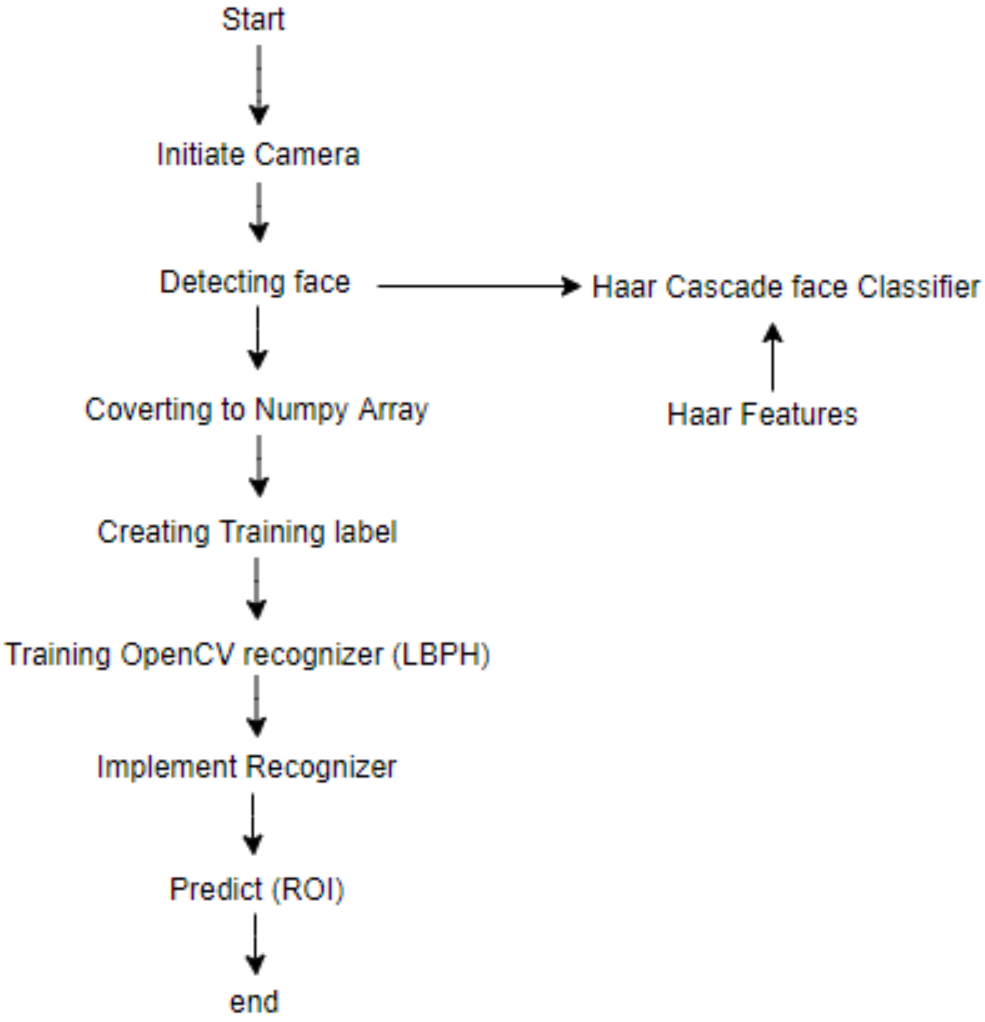
```
recognizer.train(x_train, np.array(y_labels))
recognizer.save("trainner.yml")
```

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

5.8 Result



5.9 Working Procedure



Chapter 6 Conclusion & Limitation

Conclusion

From designing to choosing and implementing the algorithm for this research project was a real challenge because there is lots of option to choose from. We didn't want to use any framework or any libraries so we choose the simplest and basic way of working which is by Using Viola Jones Algorithm based Cascade Classifier for faces. Overall this was a great experience because what we learnt here was all the basic steps of how a computer vision works and learns.

Limitations

Training the recognizer was our first challenge because we have to give a lot of different photos in different lighting. We found that different lighting causes different prediction which is a big deal when it comes to trust part. In future we will use a more powerful machine learning approach to train our recognizer such as Keras, PyTorch and we have an idea to implement this system over a real-life scenario.

Reference

1) https://docs.opencv.org/3.4/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html#ac33ba992b16f29f2824761cea5cd5fc5
2) https://en.wikipedia.org/wiki/Local_binary_patterns
3) https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b
4) http://www.willberger.org/cascade-haar-explained/?fbclid=IwAR2tj_CB_-LlbOqS-dwX6GHGA0qFPTllg71A4GpjImL4pzSur7RMNQU3SO8

5) ShervinEmami, Valentin PetrutSuciu,Facial Recognition using OpenCV
6) Yi-Qing Wang, An Analysis of the Viola-Jones Face Detection Algorithm, Published in Image Processing On Line on 2014–06–26
7) Computer Vision and Image Understanding, Volume 187, October 2019

8) S. Zafeiriou, C. Zhang, Z. Zhang, A Survey on Face Detection in the wild: past, present and future, Computer Vision and Image Understanding (2015), doi:http://dx.doi.org/10.1016/j.cviu.2015.03.015

9) Paul Viola & Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001