Real time Face Recognition with opencv :

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source.

Human beings perform face recognition automatically every day and practically with no effort.

It seems simple to us but in practice it has proven to be complex task for a computer as many factors tend to decrease the accuracy of methods.

It has become very popular in the last two decades, mainly because of the new methods developed and the high quality of current videos/cameras.

face recognition is different of face detection:

- Face Detection: it has the objective of finding the faces (location and size) in an image and probably extract them to be used by the face recognition algorithm.
- Face Recognition: with the facial images already extracted, cropped, resized and usually converted to grayscale, the face recognition algorithm is responsible for finding characteristics which best describe the image.

There are different types of face recognition algorithms, for example:

- Eigenfaces (1991)
- Local Binary Patterns Histograms (LBPH) (1996)
- Fisherfaces (1997)
- Scale Invariant Feature Transform (SIFT) (1999)
- Speed Up Robust Features (SURF) (2006)

In this we are going to use Local Binary Patterns Histograms(LBPH) as it is one of the simplest and robust algorithms and students can understand it without major difficulties.

Introduction

Local Binary Pattern (LBP) is 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.

LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

Recognition tasks are performed with LBP as it is a visual descriptor.

There are 4 parameters in LBPH:

- Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2. creating the database :

A SQLite3 is used because SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. It is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

A connection conn object is created with the connect method of sqlite. The sql query is executed and committed for maintaining the durability of the database. Close() closes the connection at the end.

The database created is used to store the data of the names associated with faces in record_face.py .

In this file the connection is created with the connection object.

OS is used to fetch the path of directories and dataset folder is created to store the frames of the faces recorded with the corresponding name in the database.

Opencv has cascade classifiers which uses the haarcascade_frontalface_default xml file.

A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

This difference is then used to categorize subsections of an image.

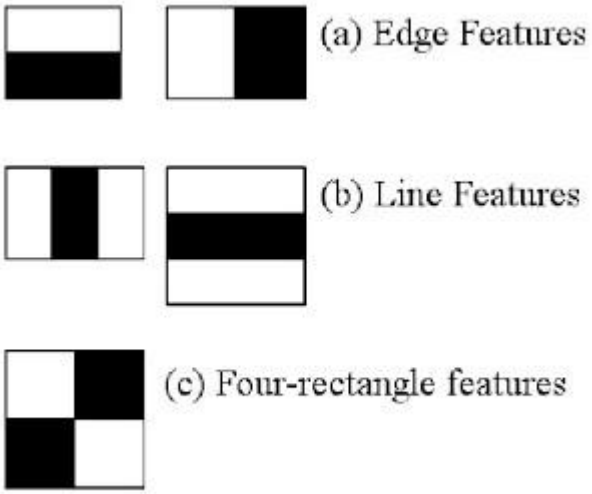
Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

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.
The algorithm has four stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object.

A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.



Query is executed to store the userid and sample number of the faces detected.
Cap is the object of opencv videoCapture method to open the camera .
Read method of cap returns two parameters

- ➔ Ret is the return parameter which determines if the camera is working or not.(it is Boolean either true or false).
- ➔ Img is the image associated with the different frames of the video being captured.

Now we find the faces in the video using the detect_multiscale method. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face.
The detected faces are stored in the database with user name and sample number with cv2.imwrite.
The data is stored in the database and comitted. the window displaying the video is closed with destroyAllWindow method of the opencv.

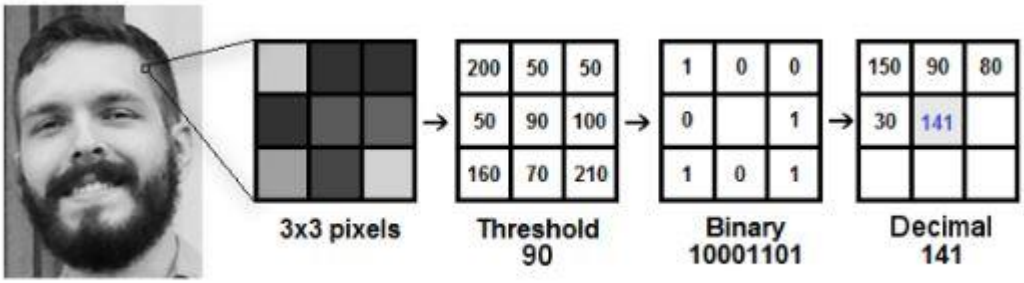
The trainer.py file is used to train the LBPH algorithm from the dataset recorded in record_face.py file.
When you execute the trainer.py file you will see the computational steps of LBPH algorithm.
The Image module provides a class with the same name which is used to represent a PIL image. The module also provides a number of factory

functions, including functions to load images from files, and to create new images.

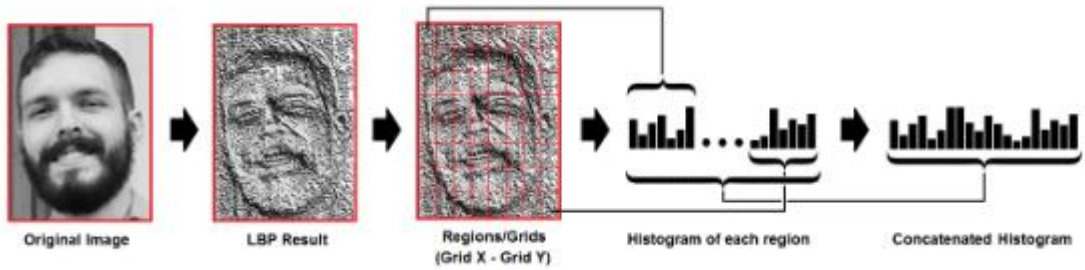
The path is set using the **os** module to get the image with id.
Image.convert returns a converted copy of this image.
When translating a color image to black and white(mode “L”) the library uses the ITU-R 601-2 luma transform:

L = R* 299/1000 +G * 587/1000 + B * 114/1000

A numpy array of the converted images is then analysed by the LBPH by using the radius and neighbor parameters as shown below.



In the next step the histogram is extracted form the images using the Grid-x and Grid-Y parameters.



the LBPH is trained at this step .

trainingData.yml is the trained classifier that is used in the face recognition phase.

In this last step of face recognition the histograms are compared and the closest match name is displayed on the screen.

We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

The LBPH classifier predicts the ids which match and the name corresponding to the id is extracted and using sql query and the result is displayed with the name in the rectangle which detects the face.