**Implementation of LBPH Algorithm**

* The dataset can be created by taking images from webcam or from saved images. We will take many samples of a single person. An unique ID or a name is given to a person in the database.

# Import the necessary libraries

import numpy as np

import cv2

import os

Python

Copy

* Creating database by adding names of peopple we wanna add to database

# creating database

database = ["Tom Cruise", "Clinton"]

Py

Copy

* Face detection using Haarcascade Classifier, cropping the face and grayscaling the face .

[Read about various Haarcascade classifiers on OpenCV](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html)

def face\_detection(image):

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

haar\_classifier = cv2.CascadeClassifier('haarcascades/haarcascade\_frontalface\_default.xml')

face = haar\_classifier.detectMultiScale(image\_gray, scaleFactor=1.3, minNeighbors=7)

(x,y,w,h) = face[0]

return image\_gray[y:y+w, x:x+h], face[0]

Python

Copy

* Preparing data with labels and faces

def prepare\_data(data\_path):

folders = os.listdir(data\_path)

labels = []

faces = []

for folder in folders:

label = int(folder)

training\_images\_path = data\_path + '/' + folder

for image in os.listdir(training\_images\_path):

image\_path = training\_images\_path + '/' + image

training\_image = cv2.imread(image\_path)

face, bounding\_box = face\_detection(training\_image)

faces.append(face)

labels.append(label)

print ('Training Done')

return faces, labels

Python

Copy

faces, labels = prepare\_data('training')

print ('Total faces = ', len(faces))

print ('Total labels = ', len(labels))

Python

Copy

* Creating LBPH model and training it with the prepared data

model = cv2.face.createLBPHFaceRecognizer()

model.train(faces, np.array(labels))

Python

Copy

* Testing the trained model using a test image

def predict\_image(test\_image):

img = test\_image.copy()

face, bounding\_box = face\_detection(img)

label = model.predict(face)

label\_text = database[label-1]

print (label)

print (label\_text)

(x,y,w,h) = bounding\_box

cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)

cv2.putText(img, label\_text, (x,y), cv2.FONT\_HERSHEY\_PLAIN, 1.5, (0, 255, 0), 2)

return img

Python

Copy

test1 = cv2.imread("test/tom.jpg")

predict1 = predict\_image(test1)

cv2.imshow('Face Recognition', predict1)

cv2.waitKey(0)

cv2.destroyAllWindows()

Python

Copy

**Advantages of LBPH algorithm**

* LBPH Method is one of the best performing texture descriptors.
* The LBP operator is robust against monotonic gray scale transformations.
* FisherFaces only prevents features of a person from becoming dominant, but it still considers illumination variations as a useful feature. But light variation is not a useful feature to extract as it is not part of the actual face.
* Fisherfaces need larger storage of face data and more processing time in recognition.
* In LBPH each image is analyzed independently, while the eigenfaces and fisherfaces method looks at the dataset.
* LBPH method will probably work better than fisherfaces in different environments and light conditions. It also depends on our training and testing data sets.
* It can represent local features in the images.
* LBPH can recognize both side and front faces.