FACE RECOGNITION

A) Face-Rec on Input Images:

1) Introduction:

A face recognition system is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database.

It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Recently, it has also become popular as a commercial identification and marketing tool.

2) Theory of OpenCV Face Recognizer:

Thanks to OpenCV, coding face recognition is as easier as it feels. The coding steps for face recognition are similar to the steps that our brains use for recognizing faces:

- Training Data Gathering: Gather face data (face images in this case) of the persons you want to recognize (the more you meet them, the more you remember them correctly)
- Training of Recognizer: Feed that face data (and respective names of each face) to the face recognizer so that it can learn.
- Recognition: Feed new faces of the persons and see if the face recognizer you just trained recognizes them.

OpenCV has three built-in face recognizers and thanks to its clean coding, we can use any of them just by changing a single line of code. Here are the names of those face recognizers and their OpenCV calls:

- EigenFaces – cv2.face.EigenFaceRecognizer_create()
- FisherFaces – cv2.face.FisherFaceRecognizer_create()
- Local Binary Patterns Histograms (LBPH) – cv2.face.LBPHFaceRecognizer_create()

We know that Eigenfaces and Fisherfaces are both affected by light and, in real life, we can't guarantee perfect light conditions. LBPH face recognizer is an improvement to overcome this drawback.

The idea with LBPH is not to look at the image as a whole, but instead, try to find its local structure by comparing each pixel to the neighboring pixels.

3) Coding Face Recognition using Python and OpenCV:

The Face Recognition process is divided into three steps:

- Prepare training data: In this step we will read training images for each person/subject along with their labels, detect faces from each image and assign each detected face an integer label of the person it belongs to.
- Train Face Recognizer: In this step we will train OpenCV's LBPH face recognizer by feeding it the data we prepared in step 1.
- Testing: In this step we will pass some test images to face recognizer and see if it predicts them correctly.

Before starting the actual coding, we need to install the Code Dependencies and import the Required Modules:

CODE DEPENDENCIES

Install the following dependencies:

- OpenCV 3.2.0
- Python v3.5
- NumPy that makes computing in Python easy. It contains a powerful implementation of N-dimensional arrays which we will use for feeding data as input to OpenCV functions.

REQUIRED MODULES

Import the following modules:

- cv2: is OpenCV module for Python which we will use for face detection and face recognition.
- os: We will use this Python module to read our training directories and file names.
- numpy: We will use this module to convert Python lists to numpy arrays as OpenCV face recognizers accept numpy arrays.

We will now move on to the core process:

a) Prepare training data:

The premise here is simple: “The more, the better”.

Like I said: “The more images used in training, the better accuracy you gain”.

Our training data consists of total 2 persons with 12 images of each person. All training data is inside training-data folder. training-data folder contains one folder for each person and each folder is named with format sLabel (ex: s1, s2) where label is actually the integer label assigned to that person. For example folder named s1 means that this folder contains images for person 1.

The test-data folder contains images that we will use to test our face recognizer after it has been successfully trained.

As OpenCV face recognizer accepts labels as integers so we need to define a mapping between integer labels and persons actual names so we will define a mapping of persons integer labels and their respective names.

OpenCV face recognizer accepts data in a specific format. It accepts two vectors, one vector is of faces of all the persons and the second vector is of integer labels for each face so that when processing a face the face recognizer knows which person that particular face belongs too.

Preparing data step can be further divided into following sub-steps:

- Read all the folder names of subjects/persons provided in training data folder. So for example, we have folder names: s1, s2.
- For each subject, extract label number. Folder names follow the format sLabel where Label is an integer representing the label we have assigned to that subject. So for example, folder name s1 means that the subject has label 1, s2 means subject label is 2 and so on. The label extracted in this step is assigned to each face detected in the next step.
- Read all the images of the subject, detect face from each image.
- Add each face to faces vector with corresponding subject label (extracted in above step) added to labels vector.

b) Train Face Recognizer:

In this project, we are going to use LBPH face recognizer for these superior effect throughout many scientific research:

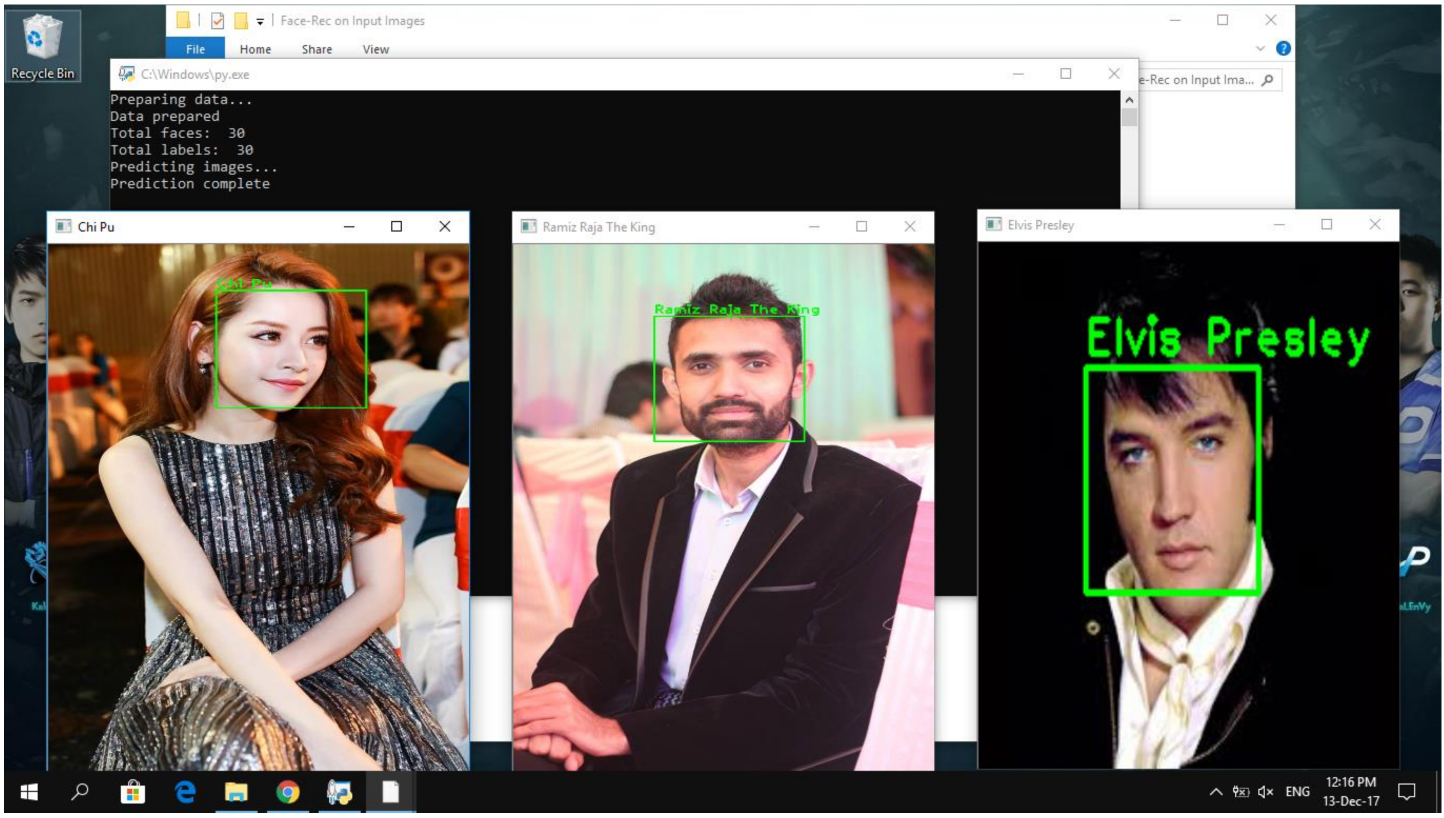
- Speed
- Smooth process on image with bad conditions: Lighting, Trash pixels, Noise ,…

c) Prediction:

This is where we actually get to see if our algorithm is actually recognizing our trained subjects's faces or not. We will take three test images of our celebrities, detect faces from each of them and then pass those faces to our trained face recognizer to see if it recognizes them correctly.

Finally, we will use functions for drawing bounding box (rectangle) around detected face and putting their corresponding subject name near the face bounding box.

Here is the wonderful result:



```
Preparing data...
Data prepared
Total faces:  30
Total labels:  30
Predicting images...
Prediction complete
```

Now that we have recognized people on Input Images successfully. However, in real life, face recognition concerns bigger problem: Real-Time Recognition, which is face recognizer system on Webcam or Security Camera mainly. Thus, we are going to explore the next aspect.

B) Face-Rec on Webcam:

First of all, you will need to install the corresponding “Opencv_contrib” modules for the face recognizer to work.

As for Face-Rec on Input Images above, we only focus on LBPH algorithm with built-in function “cv2.face.LBPHFaceRecognizer_create()”.

For Face-Rec on Webcam, we need to track and detect both faces and eyes by using the OpenCV pre-trained classifiers for face / eyes detection:

“Config.py” will contain:

- FACE_CASCADE_FILE = "opencv-files\haarcascade_frontalface_default.xml"
- EYE_CASCADE_FILE = "opencv-files\haarcascade_eye.xml"

And we will output trained dataset so as to reuse the recognizer on them if they haven’t been modified:

- RECOGNIZER_OUTPUT_FILE = "train_result.out"

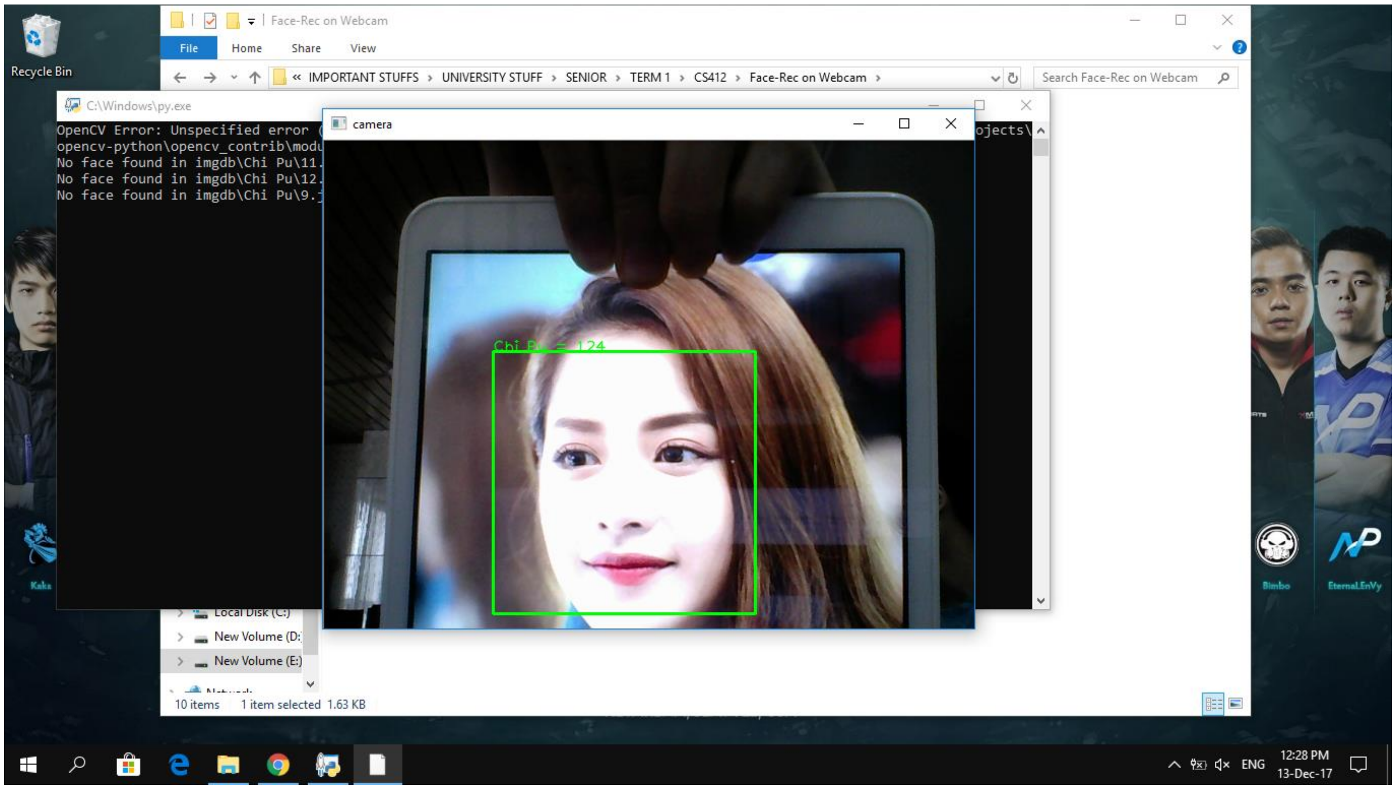
“Detect.py” contains sample on running face detection on webcam attached to the computer. The levelFace function tries to rotate the image so that the face detected is level (based on the eye positions).

“Train.py” trains the face recognizer by extracting faces from images provided under a given folder. Images for each individual should be organized in corresponding sub-folders with the folder name used by face recognizer as the labels.

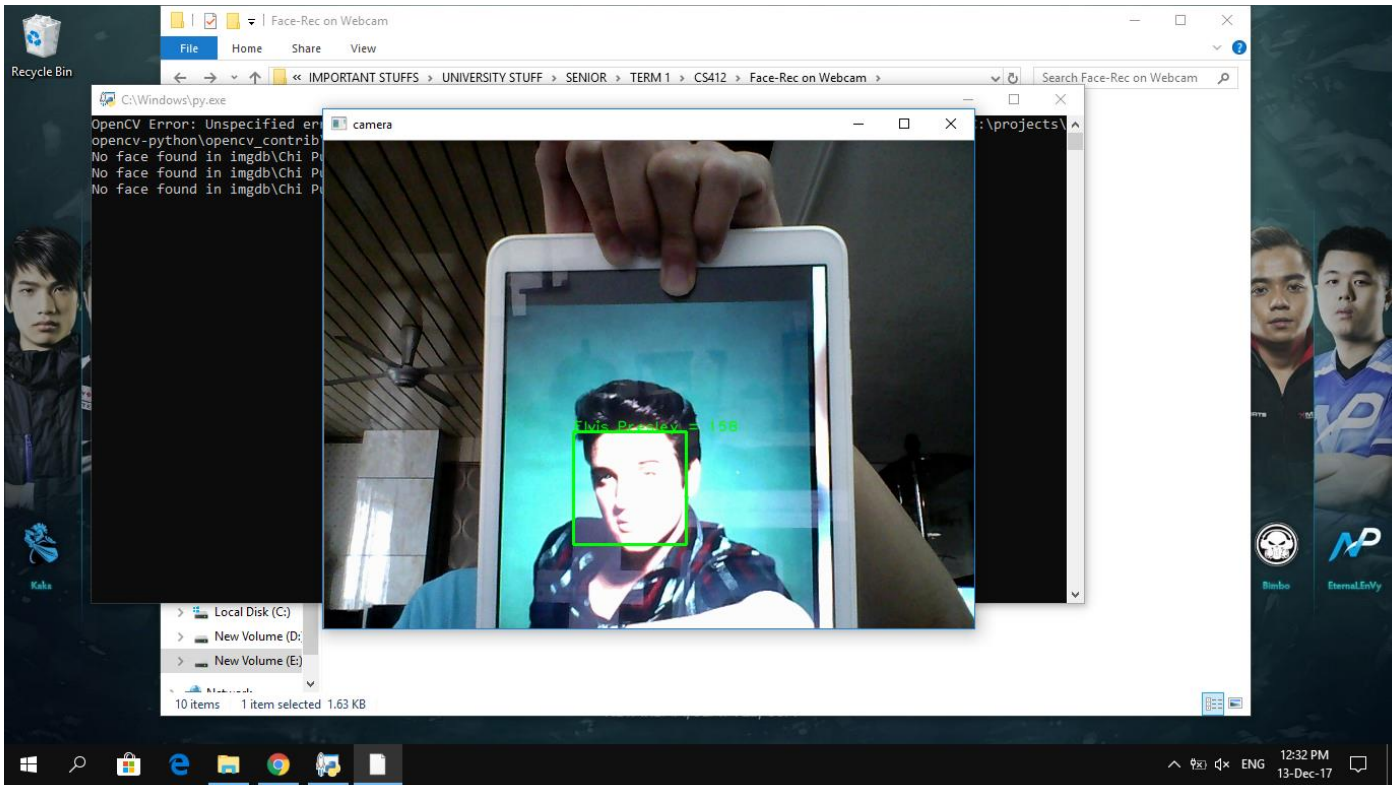
“Recognize.py” puts everything together. It demonstrates on training the face recognizer and feeding webcam images to recognize detected faces.

In order to speed up the process, the training result should run once and saved, which is the “train_result_out” process on “Config.py” already. Subsequent running of the program can load the recognizer immediately instead of training the recognizer again.

Here are the marvelous results:



```
OpenCV Error: Unspecified error (
opencv-python\opencv_contrib\modu
No face found in imgdb\Chi Pu\11.
No face found in imgdb\Chi Pu\12.
No face found in imgdb\Chi Pu\9.j
```



```
OpenCV Error: Unspecified er
opencv-python\opencv_contrib
No face found in imgdb\Chi Pu
No face found in imgdb\Chi Pu
No face found in imgdb\Chi Pu
```

Conclusion:

Face Recognition is fascinating and OpenCV has supported incredibly straightforward path for us to implement it.

Although EigenFaces, FisherFaces, and LBPH face recognizers are smooth and accurate, there are even better ways to perform face recognition like using Histogram of Oriented Gradients (HOGs) and Neural Networks.

More advanced face recognition algorithms are implemented using a combination of OpenCV and Machine Learning such as: Tensorflow, Support Vector Machine (SVM), Deep Neural Networks.

References:

http://www.face-rec.org/

https://www.superdatascience.com/opencv-face-recognition/

https://github.com/opencv/opencv/blob/master/samples/python/facedetect.py

https://docs.opencv.org/3.0-beta/modules/objdetect/doc/cascade_classification.html

https://docs.opencv.org/3.0-beta/modules/face/doc/facerec/facerec_tutorial.html

https://github.com/kitsook/face

https://github.com/habrman/FaceRecognition

https://thecodacus.com/opencv-python-face-detection/#.Wi6end-WbIU

https://github.com/cmusatyalab/openface

https://github.com/ageitgey/face_recognition

https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-in-tensorflow-66e7645015b8

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78