SIGNATURE AUTHENTICATOR

TCS REMOTE INTERNSHIP PROGRAM

Mentor: Mr. Nikku Nishant (Tata Consultancy Services Private Ltd.)

Mentee: Sanyam Kapoor (DT20173748566)

Mentee: Aditya Arora (DT20173756008)

Mentee: Utkarsh (DT20173749356)

Contents

1.) Introduction

Nowadays, person identification (recognition) and verification (authentication) is very important in security and resource access control. Biometrics is the science of automatic recognition of individual depending on their physiological and behavioral attributes. For centuries, handwritten signatures have been an integral part of validating business transaction contracts and agreements. Among the different forms of biometric recognition systems such as fingerprint, iris, face, voice, palm etc., signature will be most widely used.

Signature recognition is a behavioral biometric. It is the procedure of determining to whom a particular signature belongs to. Depending on acquiring of images, there are two types of signature recognition systems:

- Online Signature Recognition
- Offline Signature Recognition

As far as this project work is concerned, it aims at recognizing the signature of a candidate by taking in an image file as input and matching the image with a predefined dataset which acts as a store for the information. The input is matched (with the help of image recognition) with the input from dataset and if it matches, the candidate will be authenticated and will be eligible to use any resource he/she requested.

2.) Technologies Used

Text Editors: Notepad++, Sublime Text

Operating System: Ubuntu, Windows

Programming Language(s): Python

Libraries: OpenCV

IDE: PyCharm (By JetBrains)

3.) Module Information

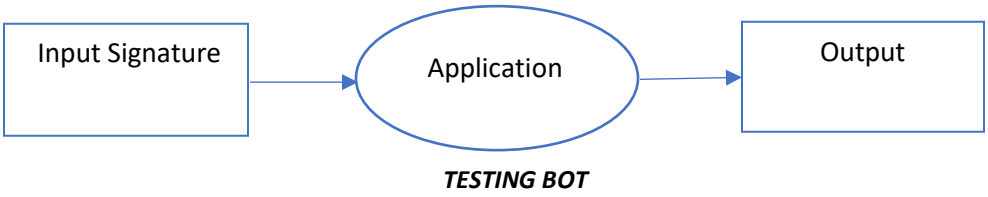
The module consists of 3 folders named as ***Correct, Incorrect*** and ***Test***. Each of these contains the corresponding signatures like ***Correct*** contains genuine signatures, ***Incorrect*** contains forged ones and ***Test*** contains signatures for testing of Bot.

Moreover, there are 2 Python(.py) files: - *train.py* for training the bot with sample images and *test.py* for testing of the signatures.

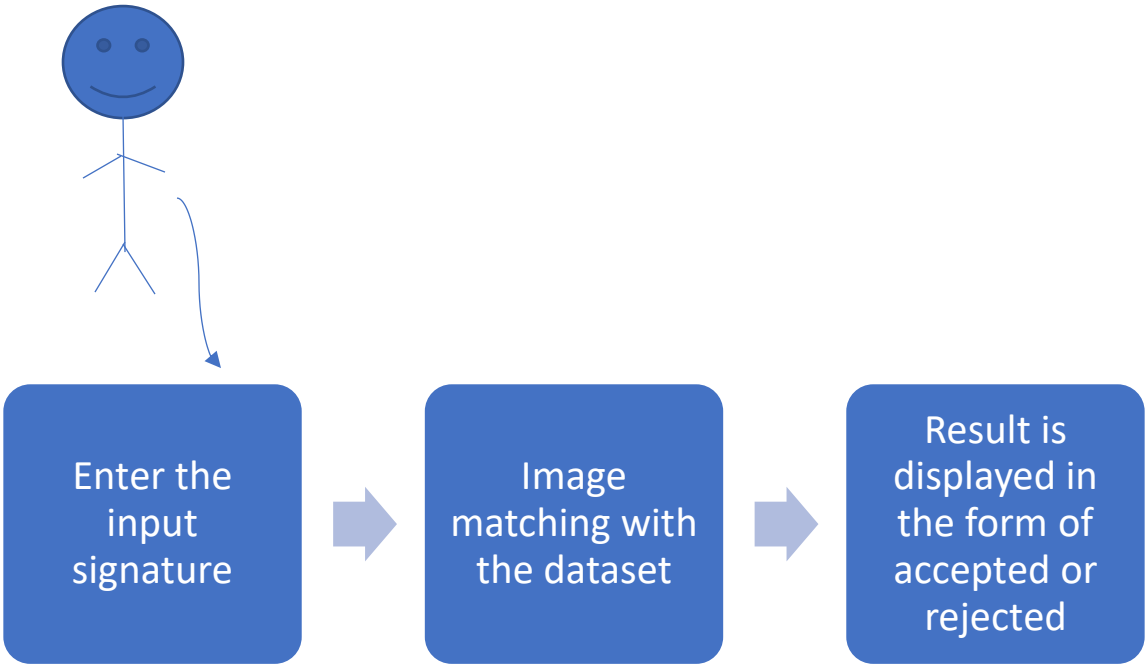
Upon execution of *train.py* 2 files will be created: - generalresponses.data and generalsamples.data which form the dataset.

NOTE: Remember that *train.py* file has to be executed before than *test.py* file.

4.) Data Flow Diagrams



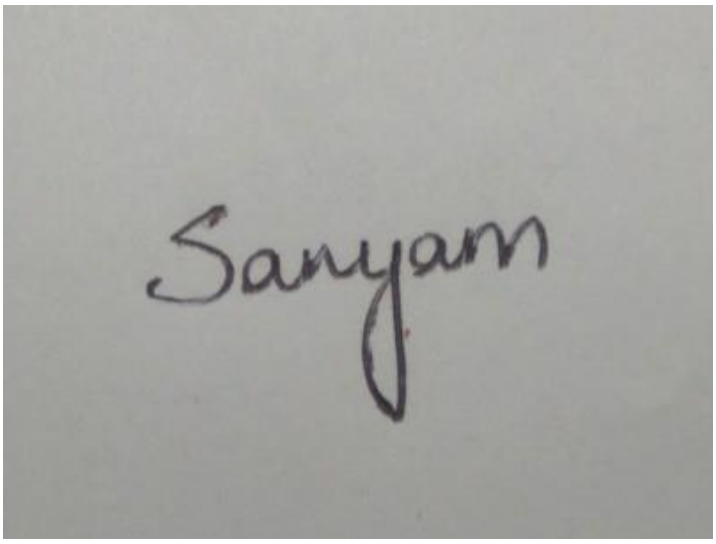
Level 0 Data Flow Diagram

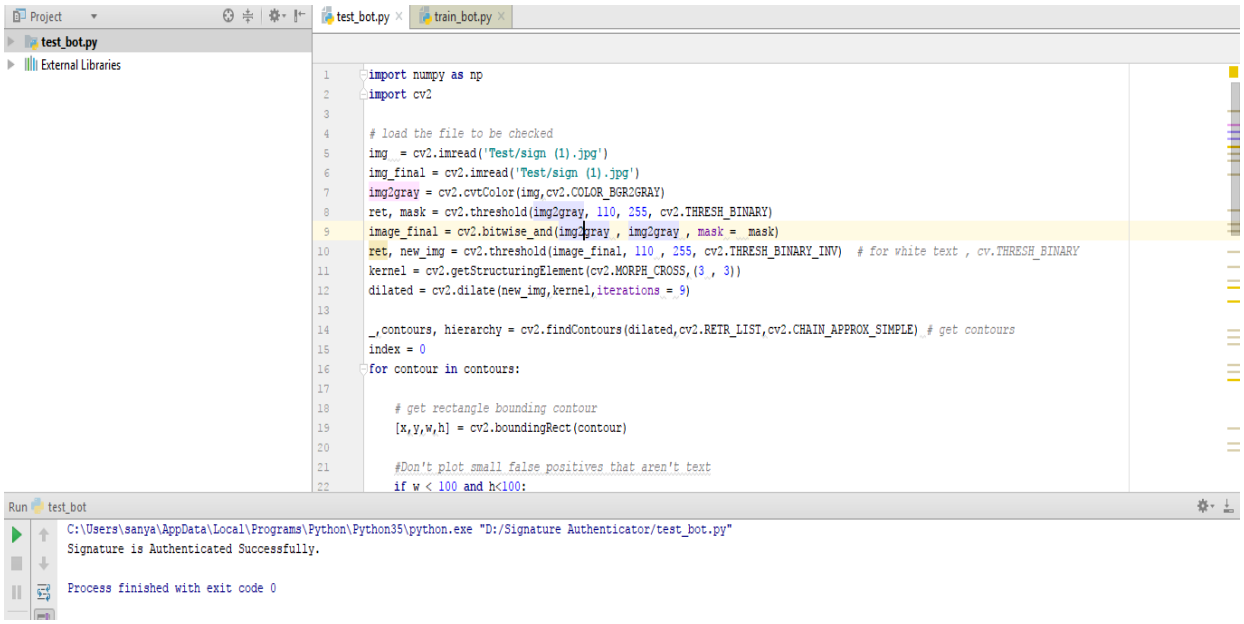


Level 1 Data Flow Diagram

5.) Test cases

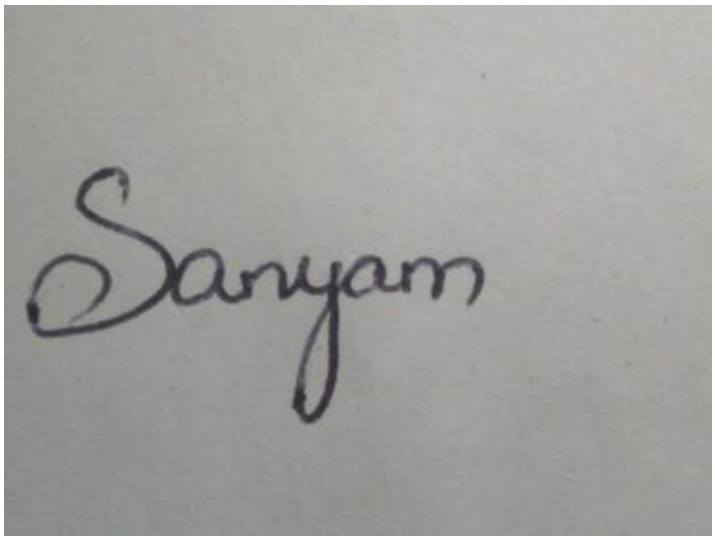
Test case #1: When signature is genuine





"Signature is Authenticated Successfully" is printed on the output screen

Test case #2: When signature is of same person but not genuine



Project

test_bot.py

External Libraries

test_bot.py | train_bot.py

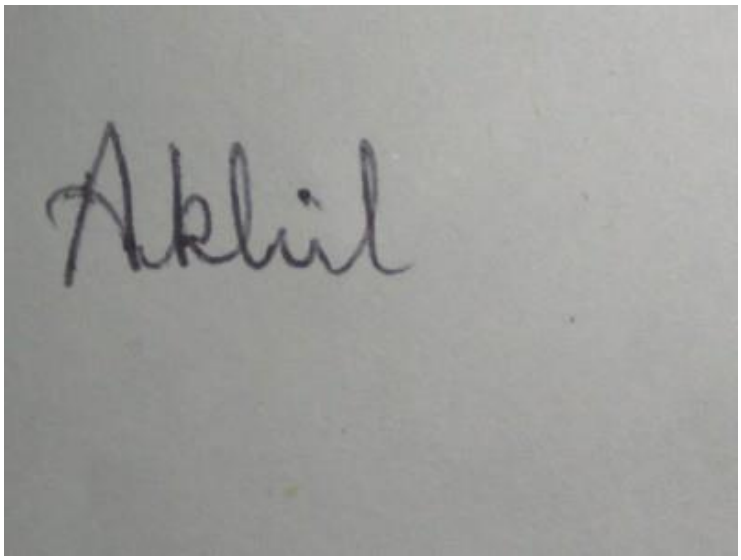
```python
import numpy as np
import cv2

# load the file to be checked
img = cv2.imread('Test/sign (5).jpg')
img_final = cv2.imread('Test/sign (5).jpg')
img2gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 110, 255, cv2.THRESH_BINARY)
image_final = cv2.bitwise_and(img2gray , img2gray , mask = mask)
ret, new_img = cv2.threshold(image_final, 110 , 255, cv2.THRESH_BINARY_INV)  # for white text , cv.THRESH_BINARY
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3 , 3))
dilated = cv2.dilate(new_img,kernel,iterations = 9)

_,contours, hierarchy = cv2.findContours(dilated,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE) # get contours
index = 0
for contour in contours:

    # get rectangle bounding contour
    [x,y,w,h] = cv2.boundingRect(contour)

    #Don't plot small false positives that aren't text
    if w < 100 and h<100:
```

Run test_bot

```
C:\Users\sanya\AppData\Local\Programs\Python\Python35\python.exe "D:/Signature Authenticator/test_bot.py"
Signature Authentication Failed.

Process finished with exit code 0
```

"Signature Authentication Failed" is printed on the output screen.

Test case #3: When the signature is of some other random person

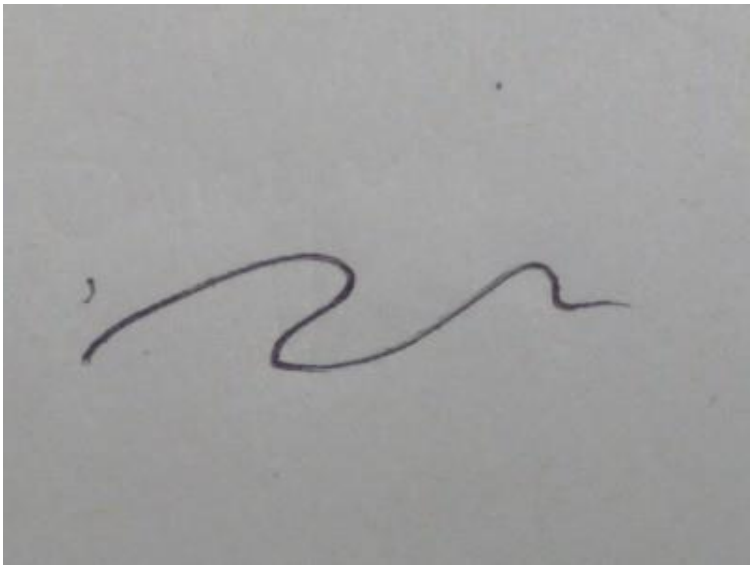


```
import numpy as np
import cv2

#Load the file to be checked
img = cv2.imread('Test/sign (22).jpg')
img_final = cv2.imread('Test/sign (22).jpg')
img2gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 110, 255, cv2.THRESH_BINARY)
image_final = cv2.bitwise_and(img2gray , img2gray , mask = mask)
ret, new_img = cv2.threshold(image_final, 110 , 255, cv2.THRESH_BINARY_INV)  # for white text , cv.THRESH_BINARY
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3 , 3))
dilated = cv2.dilate(new_img,kernel,iterations = 9)

_,contours, hierarchy = cv2.findContours(dilated,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE) # get contours
index = 0
for contour in contours:

    # get rectangle bounding contour
    [x,y,w,h] = cv2.boundingRect(contour)

    #Don't plot small false positives that aren't text
    if w < 100 and h<100:
```

```
C:\Users\sanya\AppData\Local\Programs\Python\Python35\python.exe "D:/Signature Authenticator/test_bot.py"
Signature Authentication Failed.

Process finished with exit code 0
```

"Signature Authentication Failed" is printed on the output screen.

Test case #4: When there is no signature/ random arrows or curls in image



```
import numpy as np
import cv2

# load the file to be checked
img  = cv2.imread('Test/sign (10).jpg')
img_final = cv2.imread('Test/sign (10).jpg')
img2gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 110, 255, cv2.THRESH_BINARY)
image_final = cv2.bitwise_and(img2gray , img2gray , mask = mask)
ret, new_img = cv2.threshold(image_final, 110 , 255, cv2.THRESH_BINARY_INV)  # for white text , cv.THRESH_BINARY
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(3 , 3))
dilated = cv2.dilate(new_img,kernel,iterations = 9)

_,contours, hierarchy = cv2.findContours(dilated,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE) # get contours
index = 0
for contour in contours:

    # get rectangle bounding contour
    [x,y,w,h] = cv2.boundingRect(contour)

    #Don't plot small false positives that aren't text
    if w < 100 and h<100:
```

```
C:\Users\sanya\AppData\Local\Programs\Python\Python35\python.exe "D:/Signature Authenticator/test_bot.py"
Signature Authentication Failed.

Process finished with exit code 0
```

"Signature Authentication Failed" is printed on the output screen.

6.) Screenshots

```
index = 0
for contour in contours:
    # get rectangle bounding contour
    [x, y, w, h] = cv2.boundingRect(contour)

    # Don't plot small false positives that aren't text
    if w < 100 and h < 100:
        continue

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

    cropped = img_final[y:y + h, x: x + w]
    s = 'final_' + 'crop_' + str(index) + '.jpg'
    cropped = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
    ret, mask = cv2.threshold(cropped, 130, 255, cv2.THRESH_BINARY)
    image_final = cv2.bitwise_and(cropped, cropped, mask=mask)
    ret, new_img = cv2.threshold(image_final, 130, 255, cv2.THRESH_BINARY)

    index = index + 1

# setting up samples and corresponding responses
responses = []
samples = np.empty((0, 625))
roismall = cv2.resize(cropped, (25, 25))

responses.append(0)
sample = roismall.reshape((1, 625))
samples = np.append(samples, sample, 0)

responses = np.array(responses, np.float32)
responses = responses.reshape((responses.size, 1))

np.savetxt(f1, samples)
np.savetxt(f2, responses)
```

Training of Bot

The image above shows the making and filling of the entries in a data file to store the image.

```python
# loading the trained data files
samples=np.loadtxt('generalsamples.data',np.float32)
responses=np.loadtxt('generalresponses.data',np.float32)
responses=responses.reshape((responses.size,1))

# applying K-Nearest Neighbours algorithm
model= cv2.ml.KNearest_create()
model.train(samples,cv2.ml.ROW_SAMPLE,responses)
roi = cropped
roismall = cv2.resize(roi, (25, 25))
roismall = roismall.reshape((1, 625))
roismall = np.float32(roismall)
retval, results, neigh_resp, dists = model.findNearest(roismall, k=1)
text = str(int((results[0][0])))

if text=='1':
    print('Signature is Authenticated Successfully.')
elif text=='0':
    print('Signature Authentication Failed.')
```

Testing Of signatures

The image above shows testing of the input signature whether it is genuine or not. In case of genuine signature "Signature is Authenticated Successfully" is printed else "Signature Authentication Failed" is printed.

generalsamples - Notepad

File Edit Format View Help

```
1.78000000000000000e+02 1.80000000000000000e+02 1.76000000000000000e+02 1.78000000000000000e+02 1.76000000000000000e+0
1.77000000000000000e+02 1.75000000000000000e+02 1.78000000000000000e+02 1.73000000000000000e+02 1.75000000000000000e+0
1.72000000000000000e+02 1.75000000000000000e+02 1.73000000000000000e+02 1.73000000000000000e+02 1.76000000000000000e+0
1.68000000000000000e+02 1.71000000000000000e+02 1.71000000000000000e+02 1.72000000000000000e+02 7.00000000000000000e+0
1.44000000000000000e+02 1.63000000000000000e+02 1.64000000000000000e+02 1.63000000000000000e+02 1.65000000000000000e+0
1.59000000000000000e+02 1.57000000000000000e+02 8.50000000000000000e+01 1.63000000000000000e+02 6.00000000000000000e+0
8.20000000000000000e+01 1.64000000000000000e+02 1.30000000000000000e+02 1.65000000000000000e+02 1.62000000000000000e+0
1.62000000000000000e+02 1.61000000000000000e+02 9.80000000000000000e+01 1.52000000000000000e+02 1.61000000000000000e+0
1.62000000000000000e+02 1.63000000000000000e+02 1.61000000000000000e+02 1.62000000000000000e+02 1.63000000000000000e+0
1.58000000000000000e+02 1.61000000000000000e+02 1.58000000000000000e+02 1.58000000000000000e+02 1.56000000000000000e+0
1.56000000000000000e+02 1.54000000000000000e+02 1.53000000000000000e+02 1.48000000000000000e+02 1.52000000000000000e+0
1.52000000000000000e+02 1.52000000000000000e+02 1.51000000000000000e+02 1.51000000000000000e+02 1.51000000000000000e+0
1.51000000000000000e+02 1.49000000000000000e+02 1.47000000000000000e+02 1.49000000000000000e+02 1.48000000000000000e+0
1.47000000000000000e+02 1.51000000000000000e+02 1.48000000000000000e+02 1.50000000000000000e+02 1.50000000000000000e+0
1.43000000000000000e+02 1.45000000000000000e+02 1.46000000000000000e+02 1.46000000000000000e+02 1.44000000000000000e+0
1.39000000000000000e+02 1.44000000000000000e+02 1.40000000000000000e+02 1.42000000000000000e+02 1.44000000000000000e+0
1.77000000000000000e+02 1.73000000000000000e+02 1.76000000000000000e+02 1.76000000000000000e+02 1.75000000000000000e+0
1.67000000000000000e+02 1.67000000000000000e+02 1.64000000000000000e+02 1.77000000000000000e+02 1.74000000000000000e+0
1.67000000000000000e+02 1.66000000000000000e+02 1.68000000000000000e+02 1.66000000000000000e+02 1.66000000000000000e+0
1.62000000000000000e+02 1.64000000000000000e+02 1.64000000000000000e+02 1.66000000000000000e+02 1.66000000000000000e+0
1.62000000000000000e+02 4.60000000000000000e+01 1.62000000000000000e+02 1.57000000000000000e+02 1.61000000000000000e+0
5.70000000000000000e+01 1.59000000000000000e+02 5.40000000000000000e+01 9.60000000000000000e+01 1.55000000000000000e+0
1.58000000000000000e+02 1.59000000000000000e+02 1.28000000000000000e+02 1.59000000000000000e+02 1.11000000000000000e+0
1.61000000000000000e+02 1.58000000000000000e+02 1.61000000000000000e+02 1.56000000000000000e+02 1.58000000000000000e+0
```

Datasets

The data file contains the details of the image in binary format in form of a 1-D array.

7.) Demonstration

The demonstration has been carried out with the mentor through screen sharing. The installation of the Python modules and OpenCV libraries was walked through.

The working software was presented to the mentor and several nuances of the working was explained in brief. The interaction was a two-way street. Inputs from the mentor were taken into account and were accommodated in the project wherever possible.

The communication is made time to time with the mentor and working of the project is discussed thoroughly through various channels such as WhatsApp, Mail and Phone calls.

8.) Future Enhancements

1.) Future enhancements can take place in the form of adding a user interface such as adding an uploading section where the user can browse and upload a signature image for verification.

2.) Elevating from the signature recognition, we can extend the service for fingerprint authentication as well.

3.) The service, still being in its early build can be enhanced and a full-fledged application can be made applicable in the form of an executable file (.exe).

9.) Sources

1.) pythonprogramming.net for learning how to incorporate Image Processing into our project.
2.) thenewboston.com for learning Python programming language.