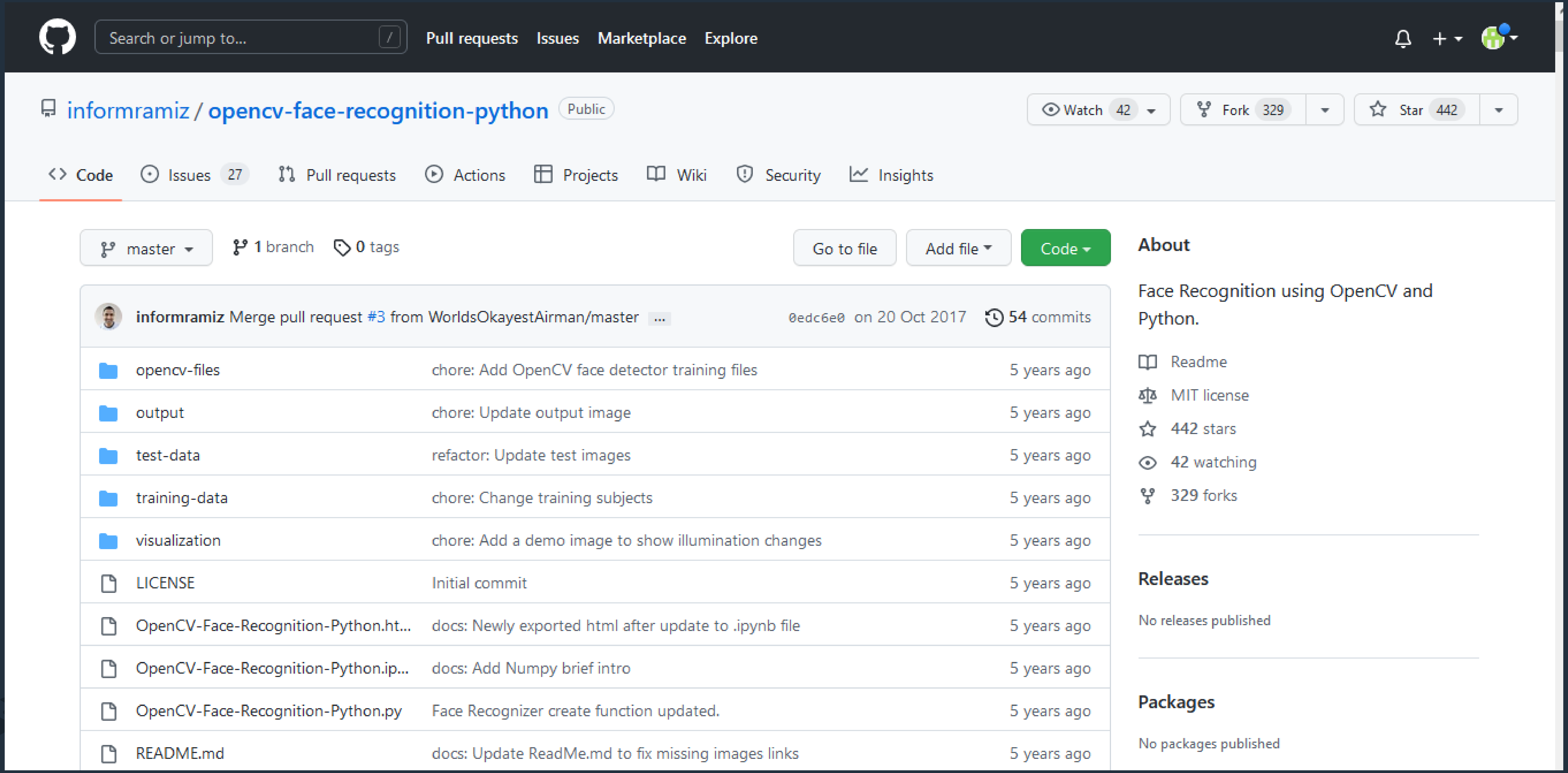
FACE RECOGNITION USING OPEN CV

Ferdinan Kurnianto – 412019007

Tikkos Marcel Hutapea – 412019036

SUMBER APLIKASI

https://github.com/informramiz/opencv-face-recognition-python



informramiz / opencv-face-recognition-python Public

informramiz Merge pull request #3 from WorldsOkayestAirman/master … 0edc6e0 on 20 Oct 2017 54 commits

opencv-files	chore: Add OpenCV face detector training files	5 years ago
output	chore: Update output image	5 years ago
test-data	refactor: Update test images	5 years ago
training-data	chore: Change training subjects	5 years ago
visualization	chore: Add a demo image to show illumination changes	5 years ago
LICENSE	Initial commit	5 years ago
OpenCV-Face-Recognition-Python.ht...	docs: Newly exported html after update to .ipynb file	5 years ago
OpenCV-Face-Recognition-Python.ip...	docs: Add Numpy brief intro	5 years ago
OpenCV-Face-Recognition-Python.py	Face Recognizer create function updated.	5 years ago
README.md	docs: Update ReadMe.md to fix missing images links	5 years ago

About

Face Recognition using OpenCV and Python.

Readme
MIT license
442 stars
42 watching
329 forks

Releases

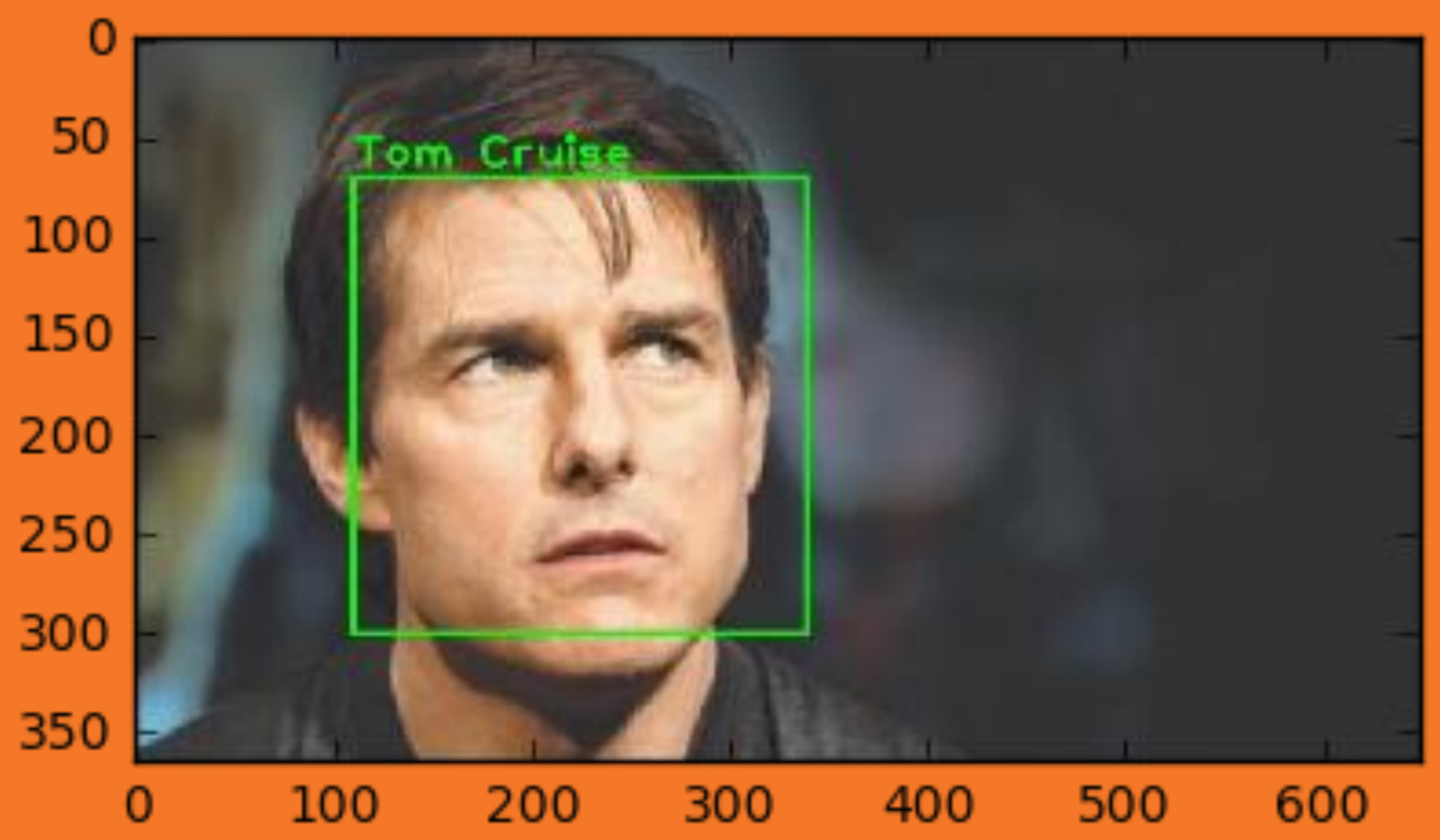
No releases published

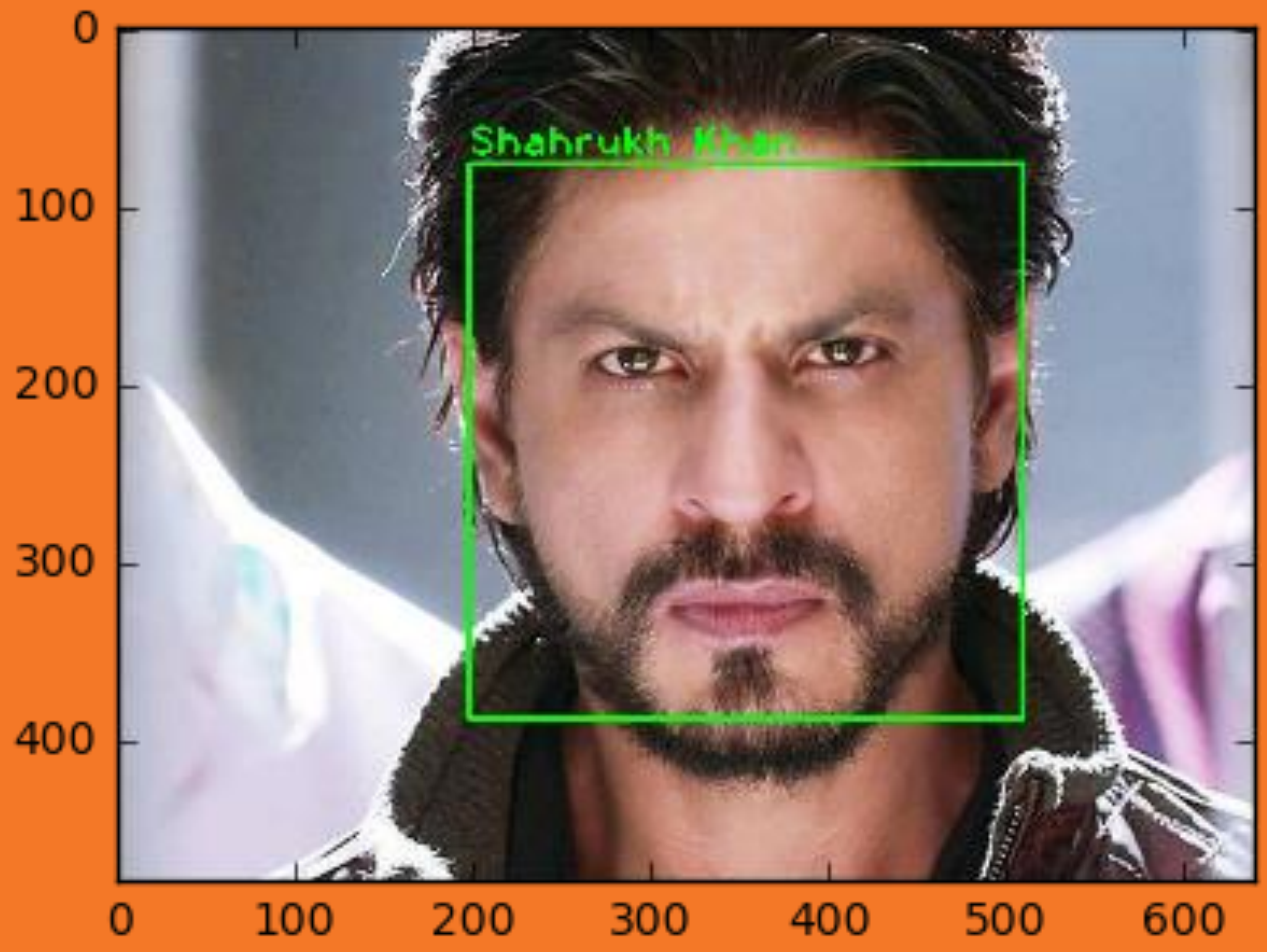
Packages

No packages published

POKOK PERSOALAN

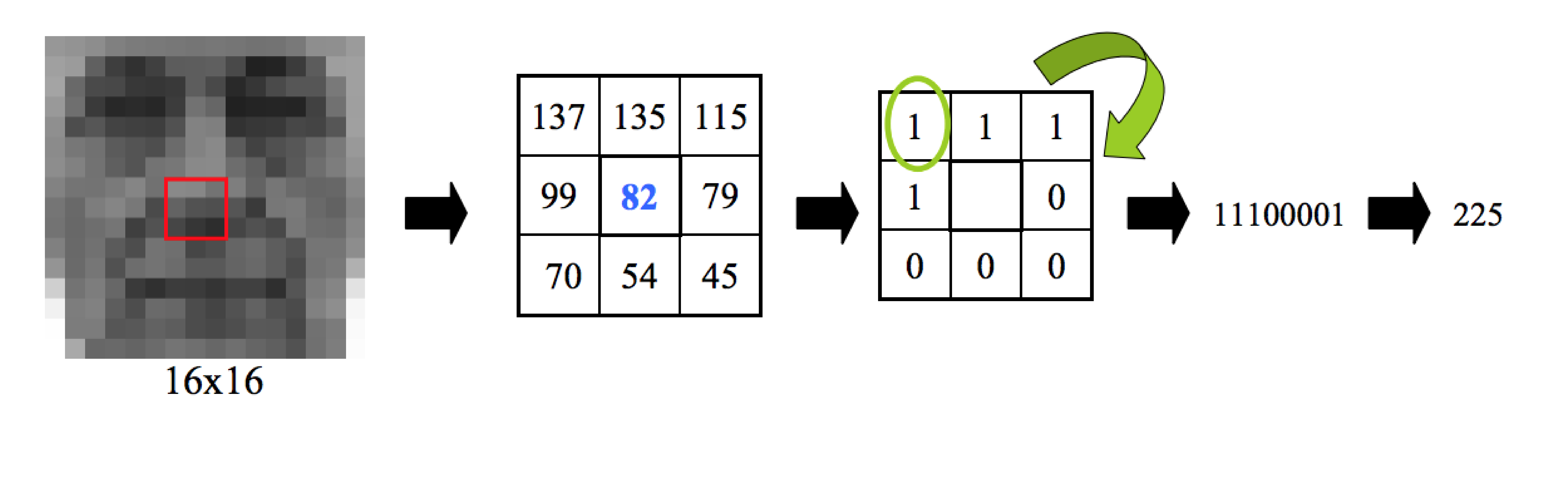
Pengenalan wajah menggunakan library OpenCV





METODE YANG DIGUNAKAN

Local Binary Patterns Histograms (LBPH) Face Recognizer untuk mengenal wajah yang diinput.

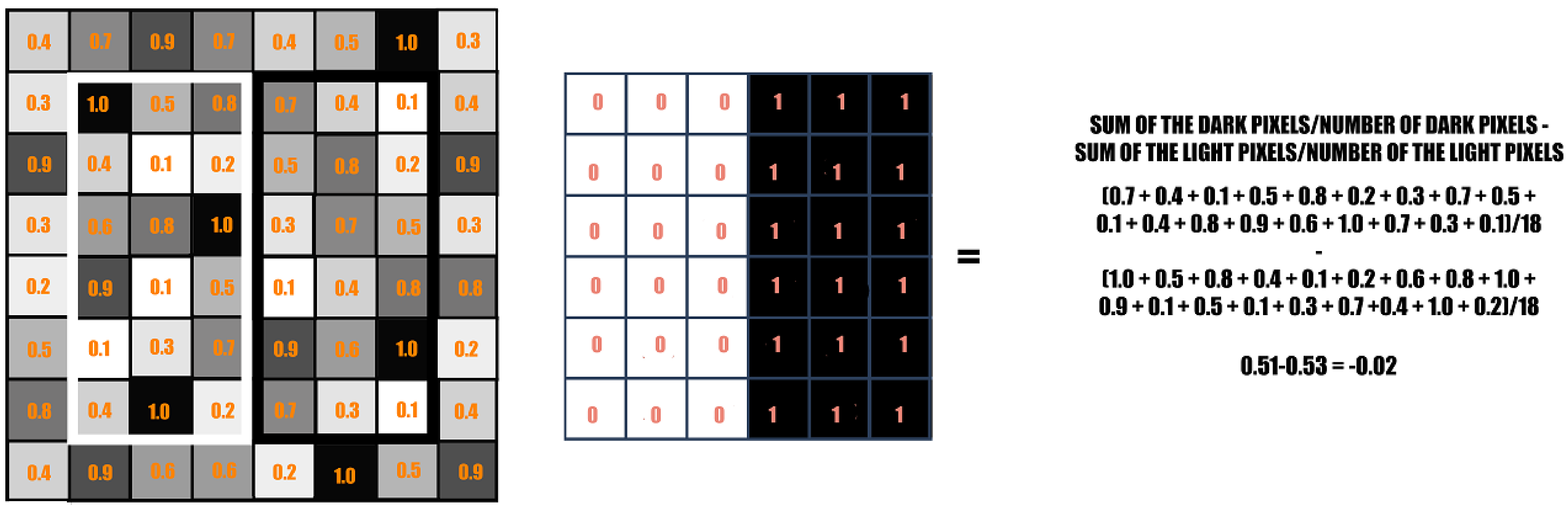


137	135	115
99	82	79
70	54	45

1	1	1
1		0
0	0	0

METODE YANG DIGUNAKAN

Haar Cascade. Digunakan untuk mendeteksi wajah yang diinput.



0.4	0.7	0.9	0.7	0.4	0.5	1.0	0.3
0.3	1.0	0.5	0.8	0.7	0.4	0.1	0.4
0.9	0.4	0.1	0.2	0.5	0.8	0.2	0.9
0.3	0.6	0.8	1.0	0.3	0.7	0.5	0.3
0.2	0.9	0.1	0.5	0.1	0.4	0.8	0.8
0.5	0.1	0.3	0.7	0.9	0.6	1.0	0.2
0.8	0.4	1.0	0.2	0.7	0.3	0.1	0.4
0.4	0.9	0.6	0.6	0.2	1.0	0.5	0.9

0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1

=

SUM OF THE DARK PIXELS/NUMBER OF DARK PIXELS -
SUM OF THE LIGHT PIXELS/NUMBER OF THE LIGHT PIXELS

(0.7 + 0.4 + 0.1 + 0.5 + 0.8 + 0.2 + 0.3 + 0.7 + 0.5 +
0.1 + 0.4 + 0.8 + 0.9 + 0.6 + 1.0 + 0.7 + 0.3 + 0.1)/18

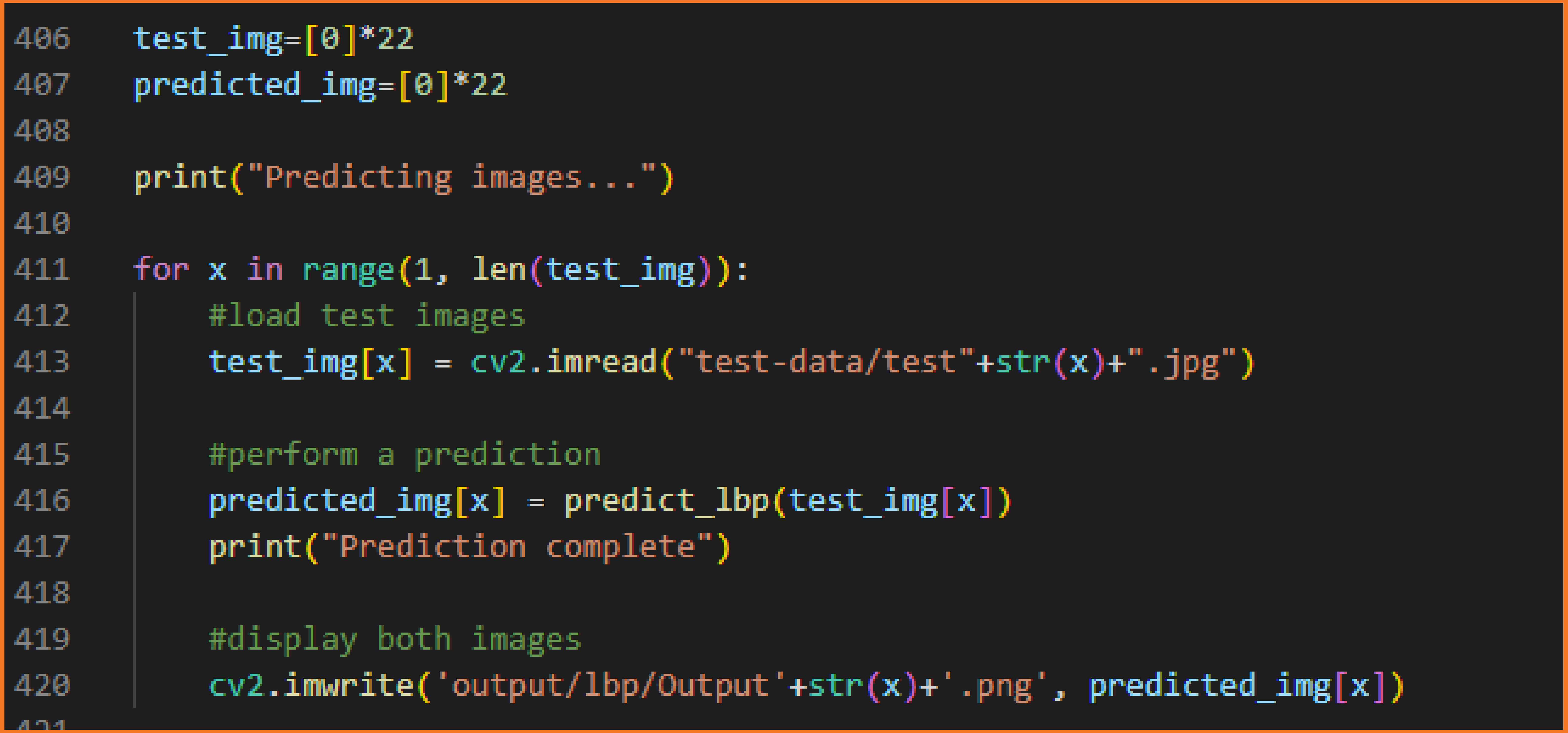
-

(1.0 + 0.5 + 0.8 + 0.4 + 0.1 + 0.2 + 0.6 + 0.8 + 1.0 +
0.9 + 0.1 + 0.5 + 0.1 + 0.3 + 0.7 +0.4 + 1.0 + 0.2)/18

0.51-0.53 = -0.02

PROSES PENGEMBANGAN 1

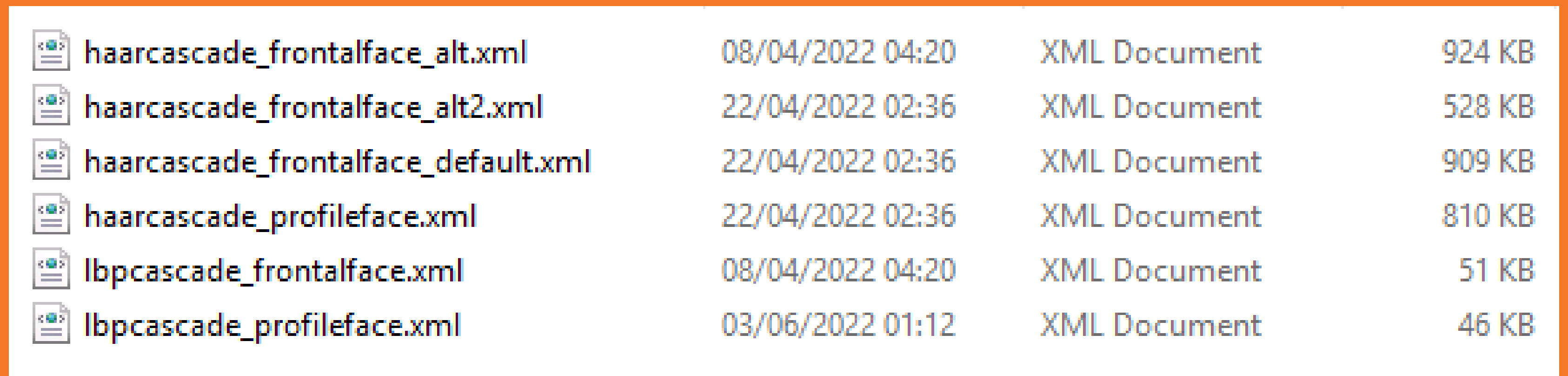
Dari repository awal, diubah bagian pengujian image yang diinput menjadi loop.



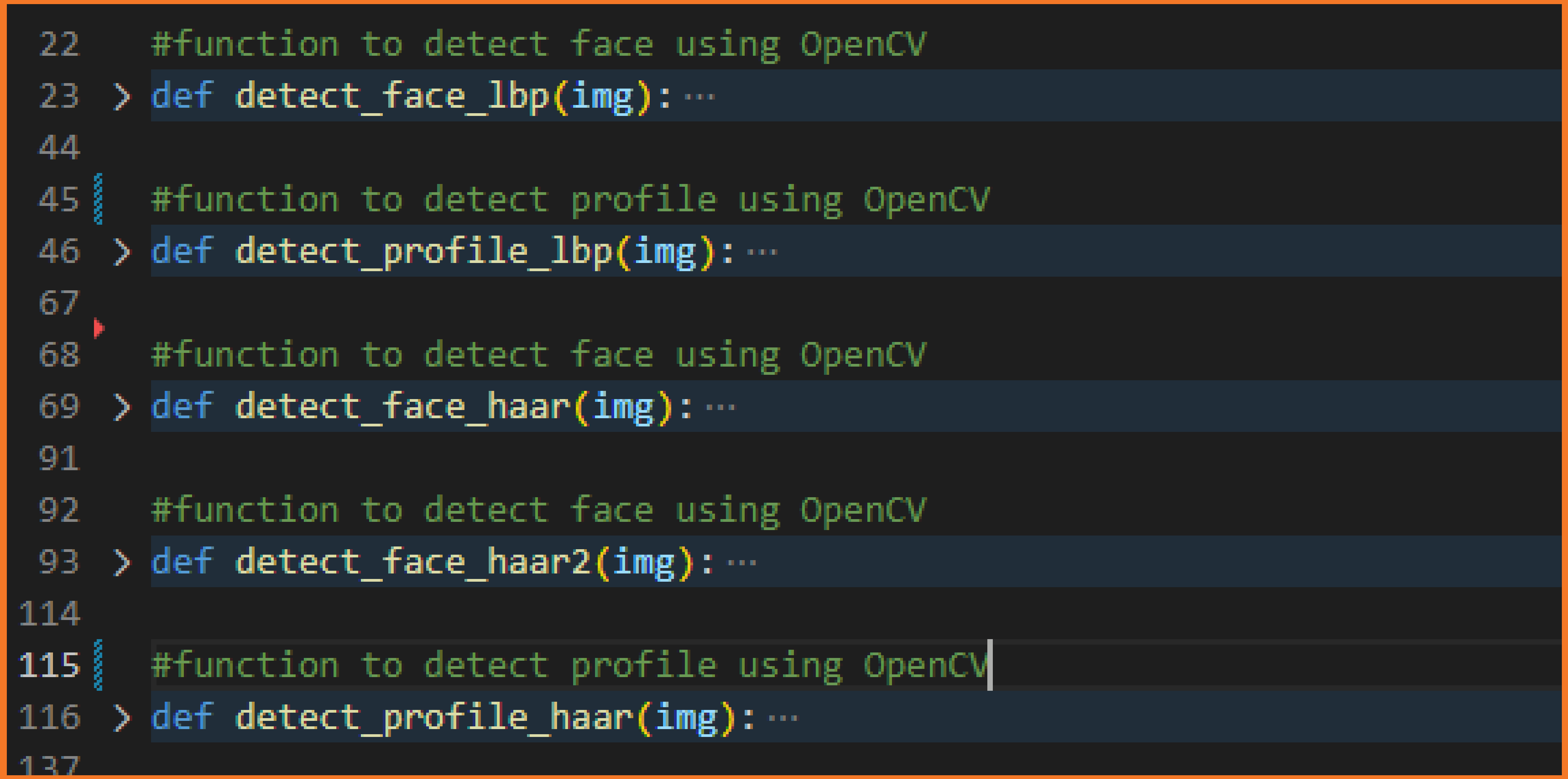
```python
test_img=[0]*22
predicted_img=[0]*22

print("Predicting images...")

for x in range(1, len(test_img)):
    #load test images
    test_img[x] = cv2.imread("test-data/test"+str(x)+".jpg")

    #perform a prediction
    predicted_img[x] = predict_lbp(test_img[x])
    print("Prediction complete")

    #display both images
    cv2.imwrite('output/lbp/Output'+str(x)+'.png', predicted_img[x])
```

PROSES PENGEMBANGAN 2

Mencari algoritma lain seperti haar cascade yang dapat digunakan untuk meningkatkan akurasi deteksi wajah input.

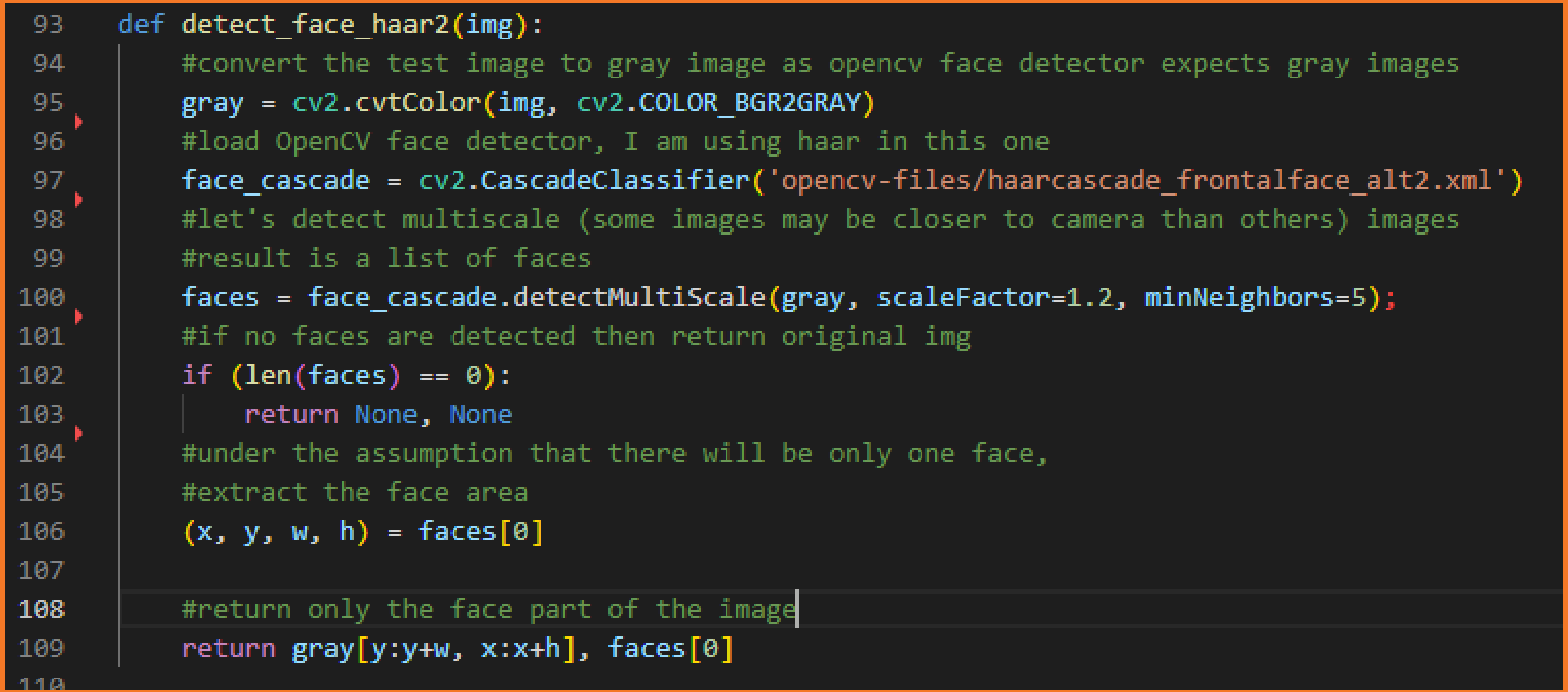


haarcascade_frontalface_alt.xml	08/04/2022 04:20	XML Document	924 KB
haarcascade_frontalface_alt2.xml	22/04/2022 02:36	XML Document	528 KB
haarcascade_frontalface_default.xml	22/04/2022 02:36	XML Document	909 KB
haarcascade_profileface.xml	22/04/2022 02:36	XML Document	810 KB
lbpcascade_frontalface.xml	08/04/2022 04:20	XML Document	51 KB
lbpcascade_profileface.xml	03/06/2022 01:12	XML Document	46 KB



```
#function to detect face using OpenCV
def detect_face_lbp(img):···

#function to detect profile using OpenCV
def detect_profile_lbp(img):···

#function to detect face using OpenCV
def detect_face_haar(img):···

#function to detect face using OpenCV
def detect_face_haar2(img):···

#function to detect profile using OpenCV
def detect_profile_haar(img):···
```

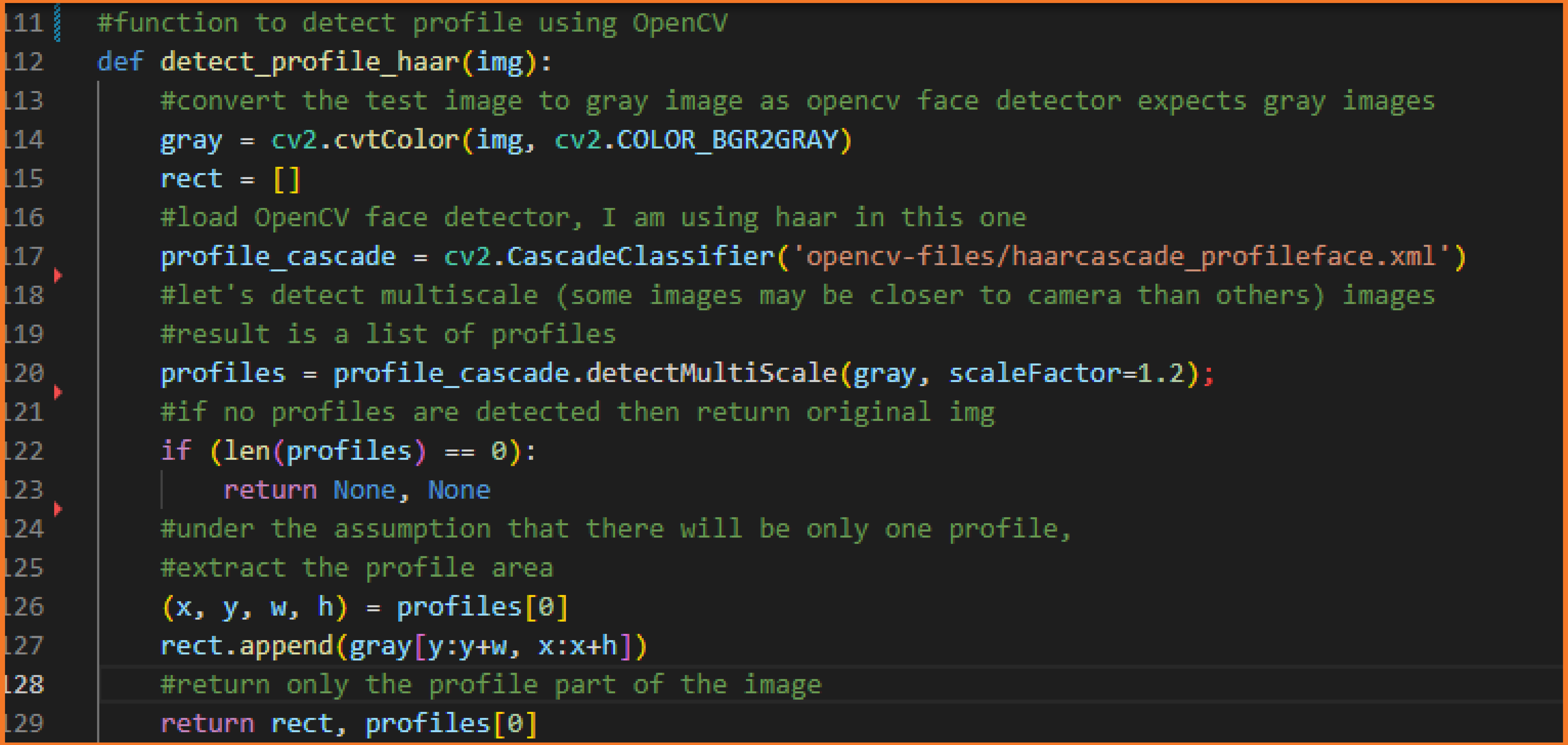
PROSES PENGEMBANGAN 2



```python
def detect_face_haar2(img):
    #convert the test image to gray image as opencv face detector expects gray images
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #load OpenCV face detector, I am using haar in this one
    face_cascade = cv2.CascadeClassifier('opencv-files/haarcascade_frontalface_alt2.xml')
    #let's detect multiscale (some images may be closer to camera than others) images
    #result is a list of faces
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5);
    #if no faces are detected then return original img
    if (len(faces) == 0):
        return None, None
    #under the assumption that there will be only one face,
    #extract the face area
    (x, y, w, h) = faces[0]

    #return only the face part of the image
    return gray[y:y+w, x:x+h], faces[0]
```

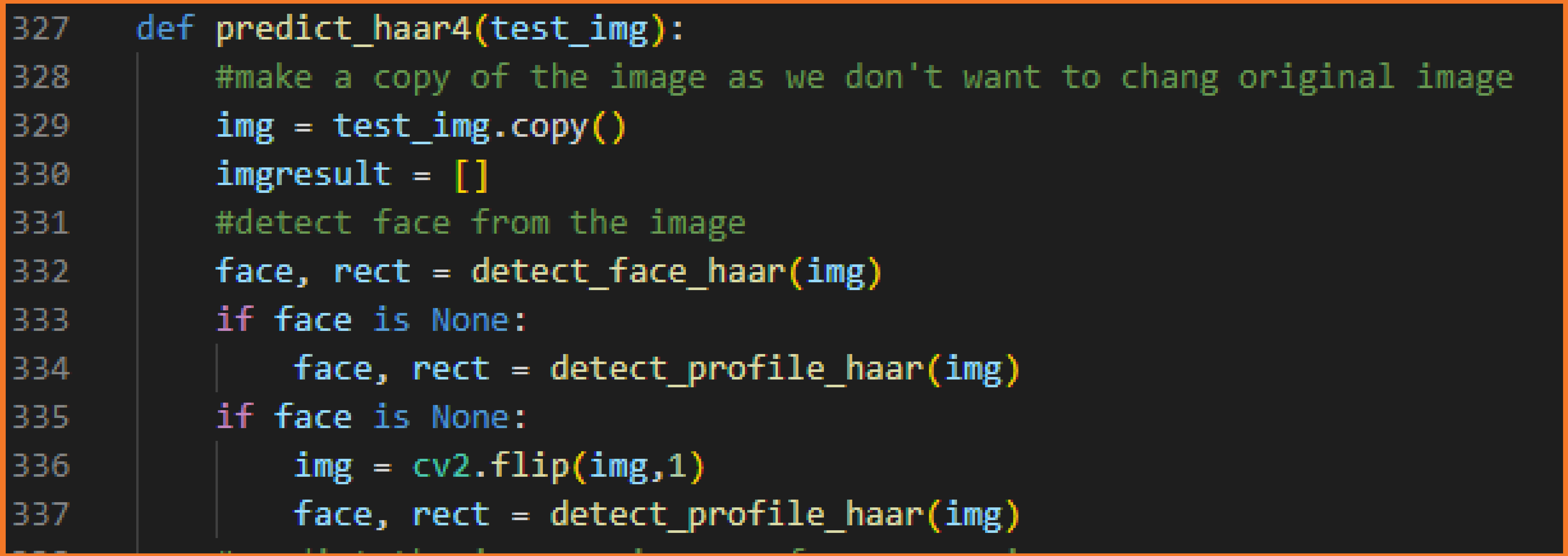
PROGRES PENGEMBANGAN 3

Mencari algoritma yang dapat digunakan untuk mendeteksi sisi wajah yang belum bisa terdeteksi.



```python
#function to detect profile using OpenCV
def detect_profile_haar(img):
    #convert the test image to gray image as opencv face detector expects gray images
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    rect = []
    #load OpenCV face detector, I am using haar in this one
    profile_cascade = cv2.CascadeClassifier('opencv-files/haarcascade_profileface.xml')
    #let's detect multiscale (some images may be closer to camera than others) images
    #result is a list of profiles
    profiles = profile_cascade.detectMultiScale(gray, scaleFactor=1.2);
    #if no profiles are detected then return original img
    if (len(profiles) == 0):
        return None, None
    #under the assumption that there will be only one profile,
    #extract the profile area
    (x, y, w, h) = profiles[0]
    rect.append(gray[y:y+w, x:x+h])
    #return only the profile part of the image
    return rect, profiles[0]
```

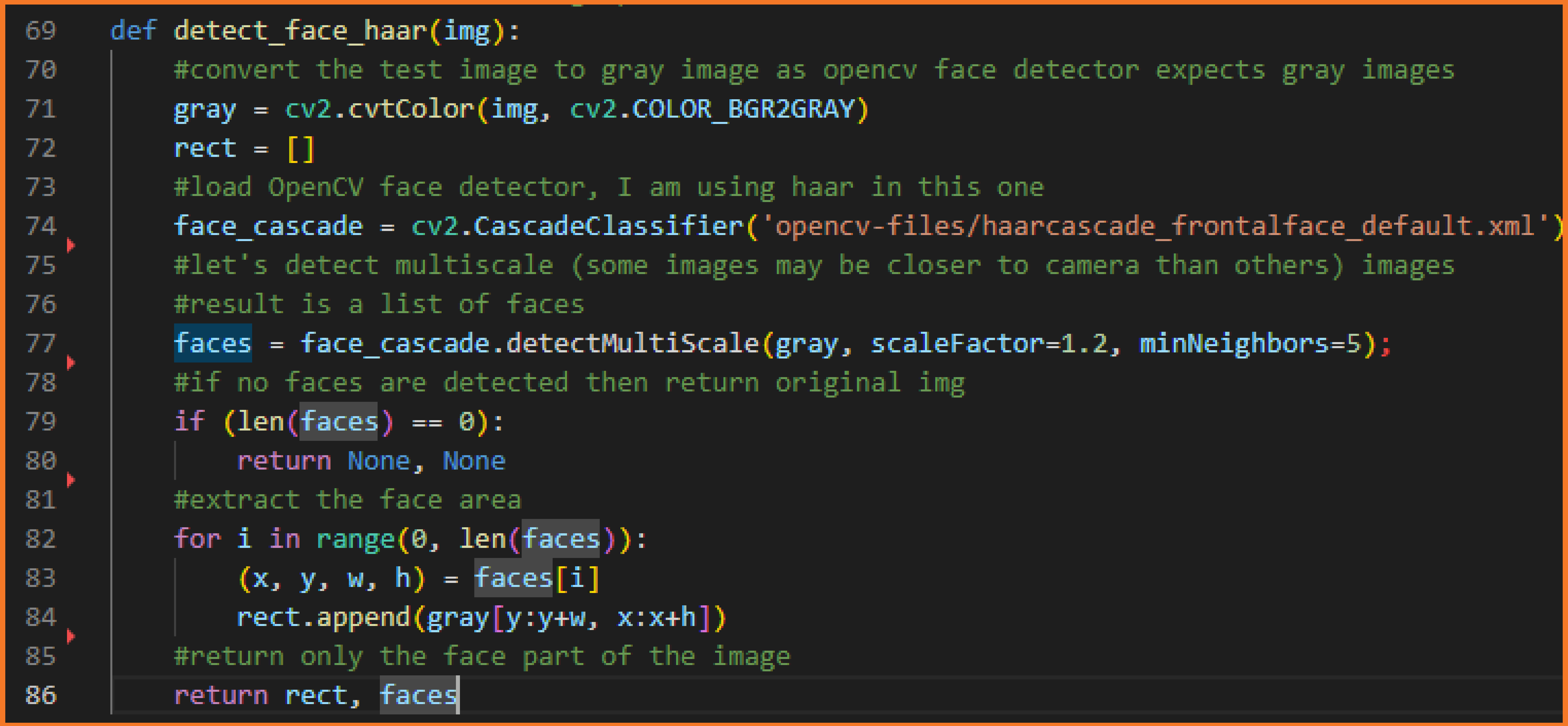
PROGRES PENGEMBANGAN 4



```
def predict_haar4(test_img):
    #make a copy of the image as we don't want to chang original image
    img = test_img.copy()
    imgresult = []
    #detect face from the image
    face, rect = detect_face_haar(img)
    if face is None:
        face, rect = detect_profile_haar(img)
    if face is None:
        img = cv2.flip(img,1)
        face, rect = detect_profile_haar(img)
```

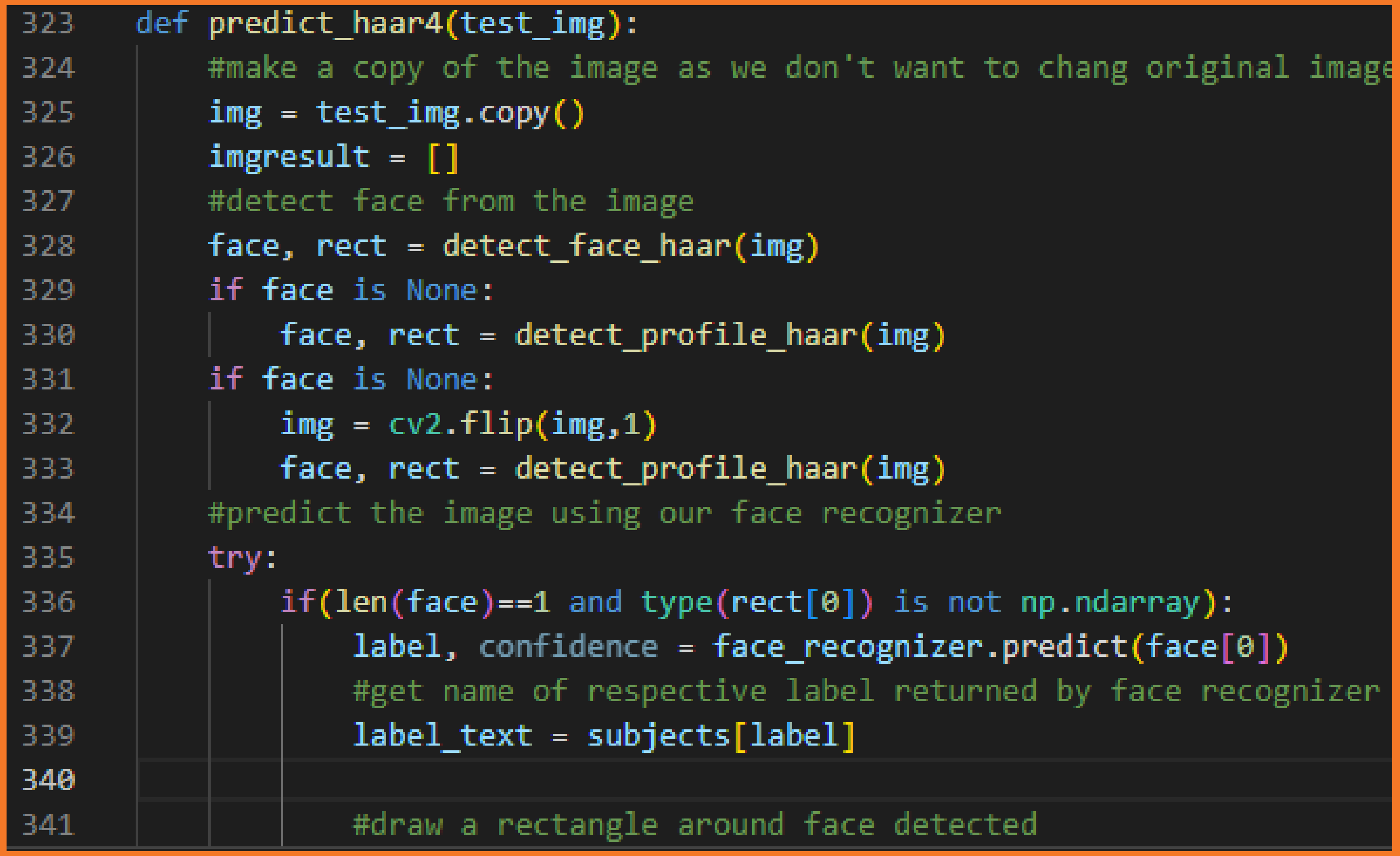
PROGRES PENGEMBANGAN 4

Mengembangkan algoritma sehingga dapat mendeteksi dan mengenali dua atau lebih wajah dan melakukan crop sekitar wajah yang dideteksi.

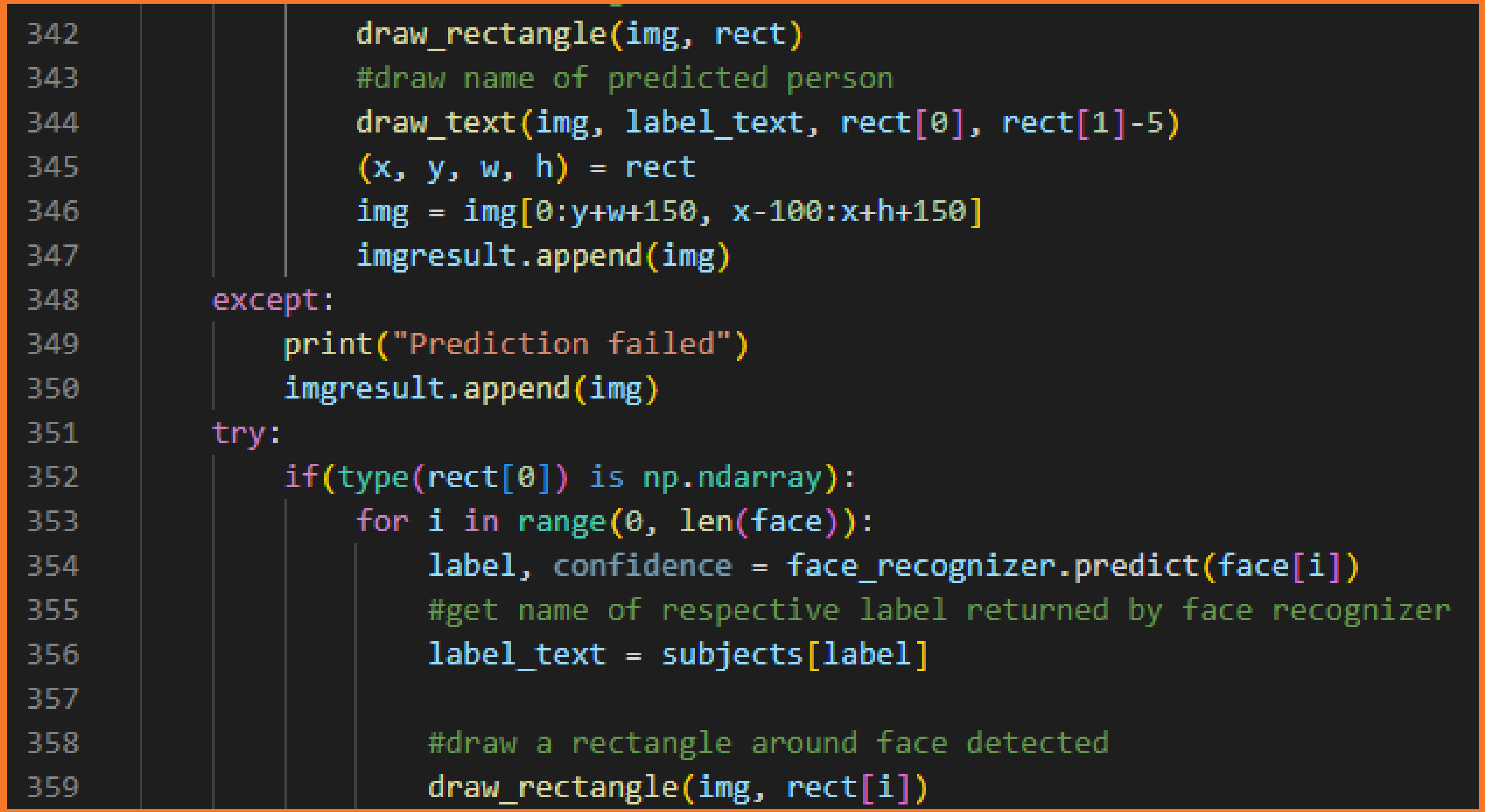


```python
def detect_face_haar(img):
    #convert the test image to gray image as opencv face detector expects gray images
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    rect = []
    #load OpenCV face detector, I am using haar in this one
    face_cascade = cv2.CascadeClassifier('opencv-files/haarcascade_frontalface_default.xml')
    #let's detect multiscale (some images may be closer to camera than others) images
    #result is a list of faces
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5);
    #if no faces are detected then return original img
    if (len(faces) == 0):
        return None, None
    #extract the face area
    for i in range(0, len(faces)):
        (x, y, w, h) = faces[i]
        rect.append(gray[y:y+w, x:x+h])
    #return only the face part of the image
    return rect, faces
```

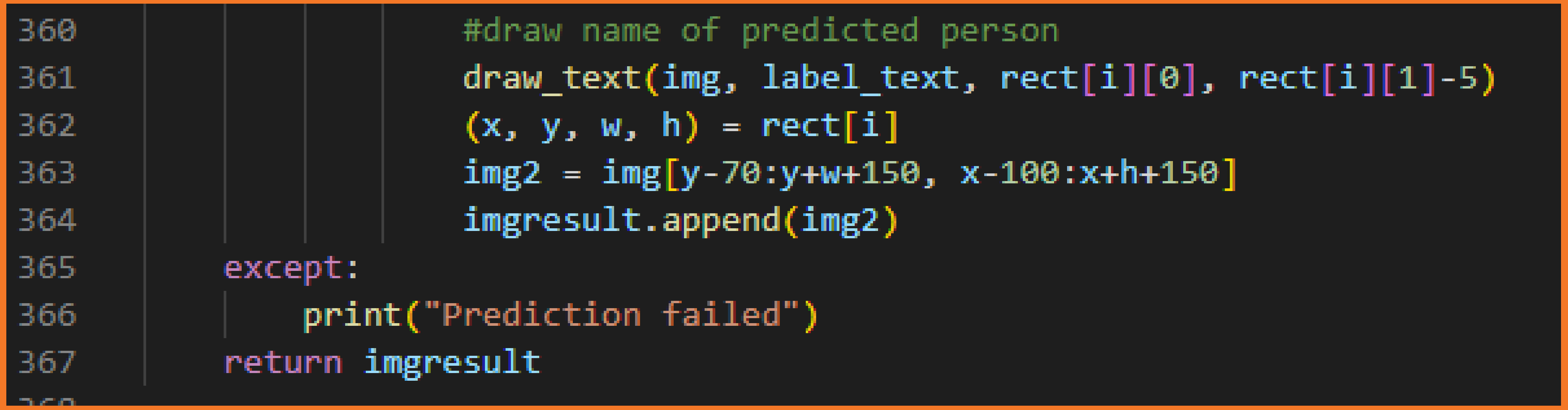
PROGRES PENGEMBANGAN 4



```python
def predict_haar4(test_img):
    #make a copy of the image as we don't want to chang original image
    img = test_img.copy()
    imgresult = []
    #detect face from the image
    face, rect = detect_face_haar(img)
    if face is None:
        face, rect = detect_profile_haar(img)
    if face is None:
        img = cv2.flip(img,1)
        face, rect = detect_profile_haar(img)
    #predict the image using our face recognizer
    try:
        if(len(face)==1 and type(rect[0]) is not np.ndarray):
            label, confidence = face_recognizer.predict(face[0])
            #get name of respective label returned by face recognizer
            label_text = subjects[label]

            #draw a rectangle around face detected
```

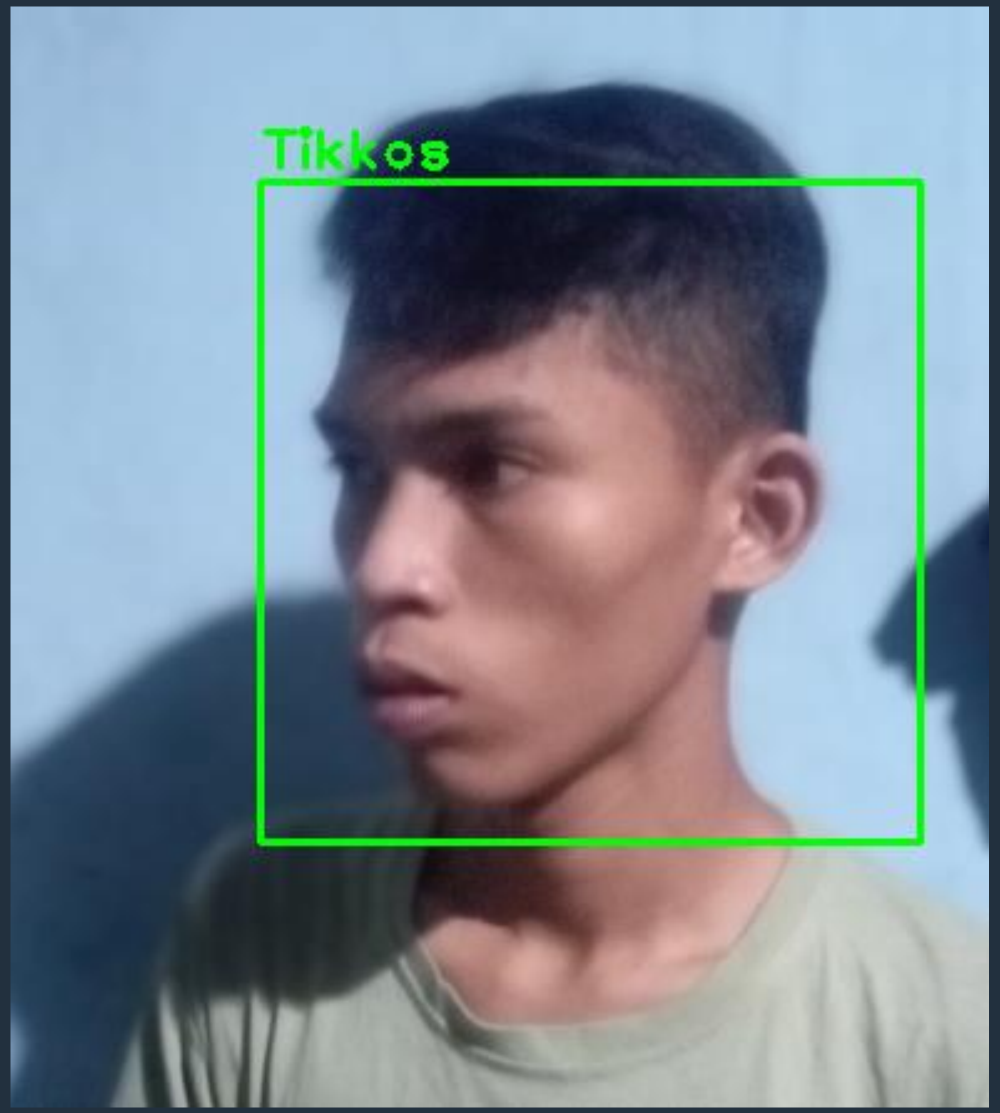


```python
            draw_rectangle(img, rect)
            #draw name of predicted person
            draw_text(img, label_text, rect[0], rect[1]-5)
            (x, y, w, h) = rect
            img = img[0:y+w+150, x-100:x+h+150]
            imgresult.append(img)
    except:
        print("Prediction failed")
        imgresult.append(img)
    try:
        if(type(rect[0]) is np.ndarray):
            for i in range(0, len(face)):
                label, confidence = face_recognizer.predict(face[i])
                #get name of respective label returned by face recognizer
                label_text = subjects[label]

                #draw a rectangle around face detected
                draw_rectangle(img, rect[i])
```

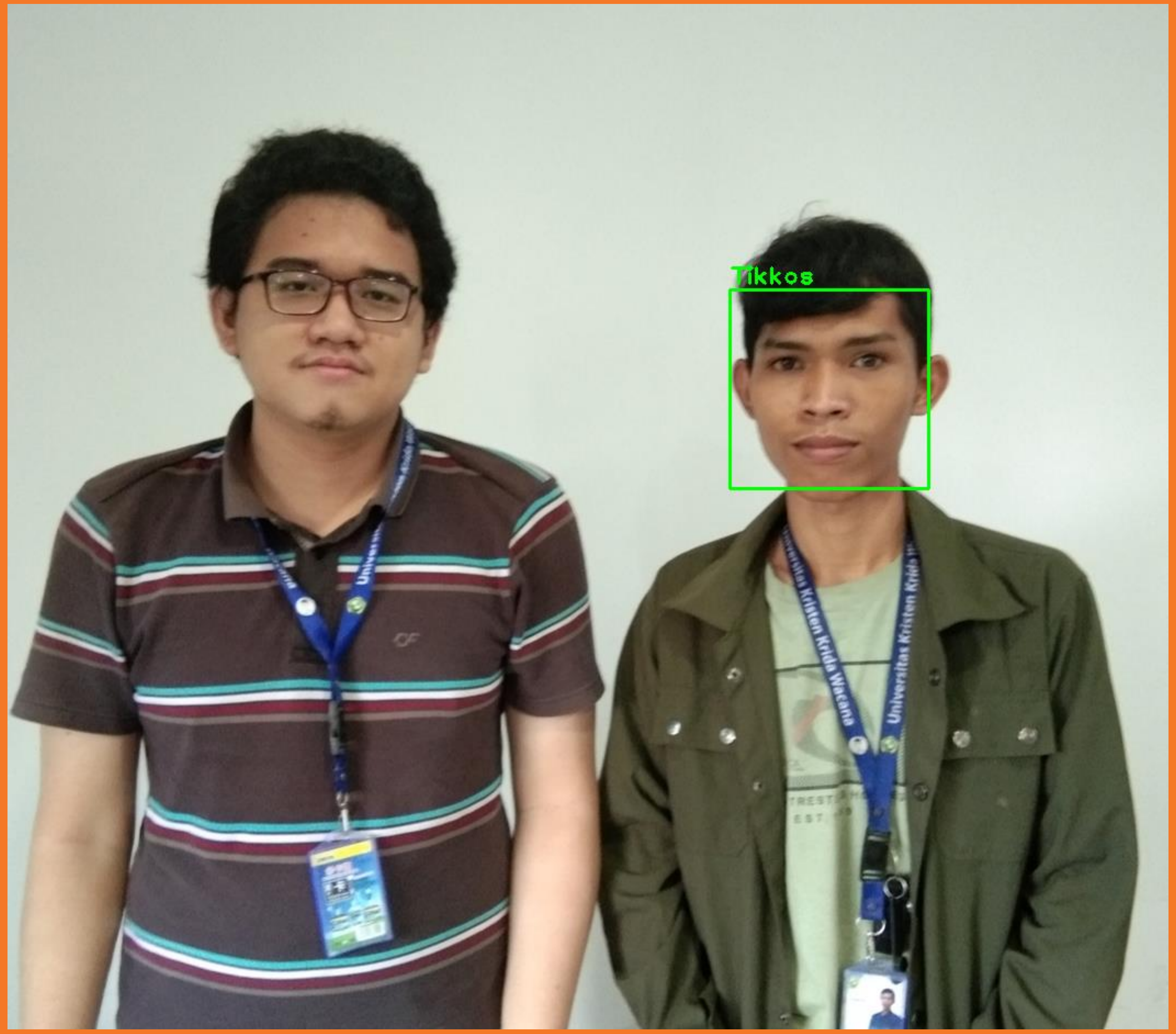


```python
                #draw name of predicted person
                draw_text(img, label_text, rect[i][0], rect[i][1]-5)
                (x, y, w, h) = rect[i]
                img2 = img[y-70:y+w+150, x-100:x+h+150]
                imgresult.append(img2)
    except:
        print("Prediction failed")
    return imgresult
```

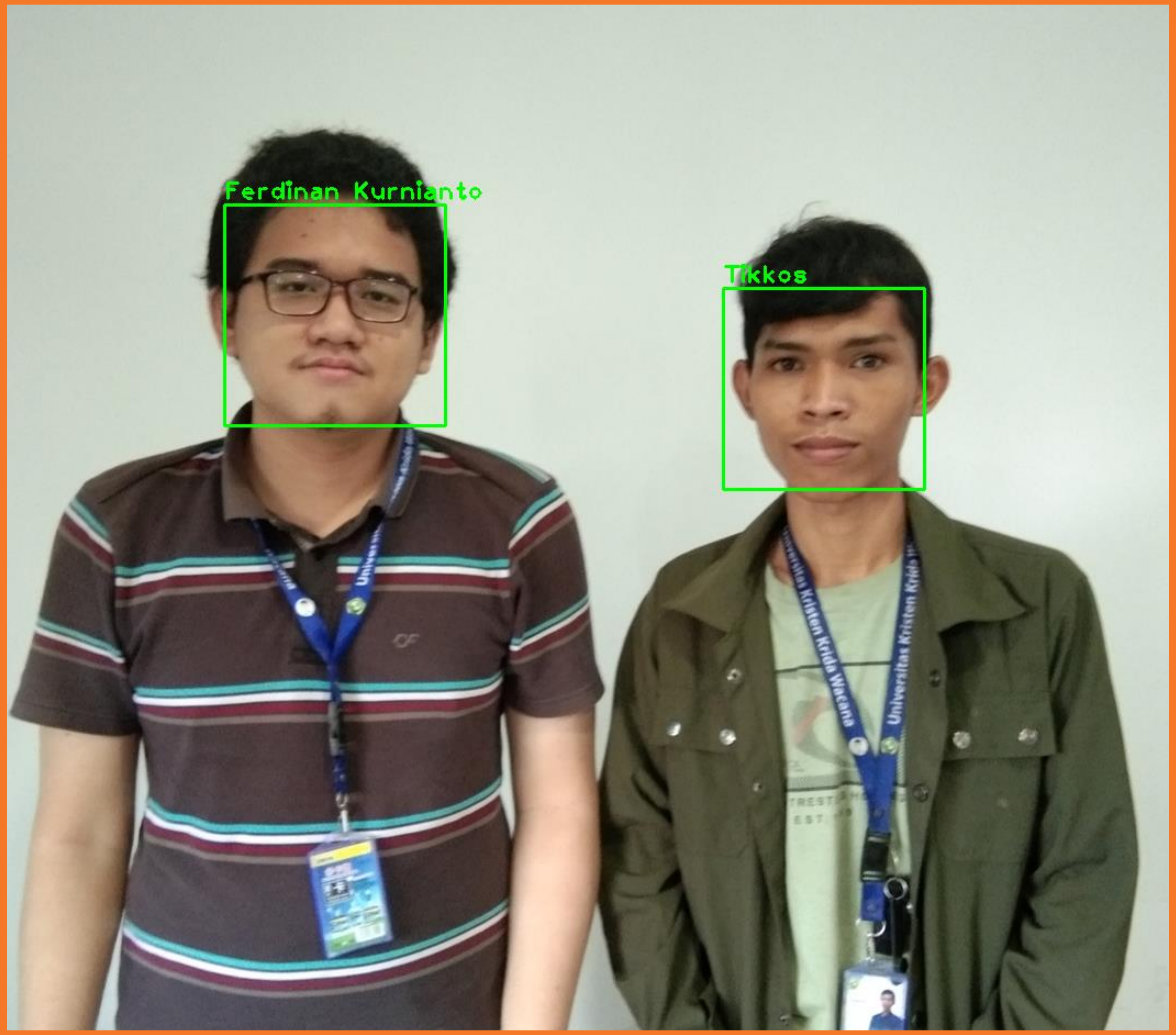
HASIL PENGEMBANGAN

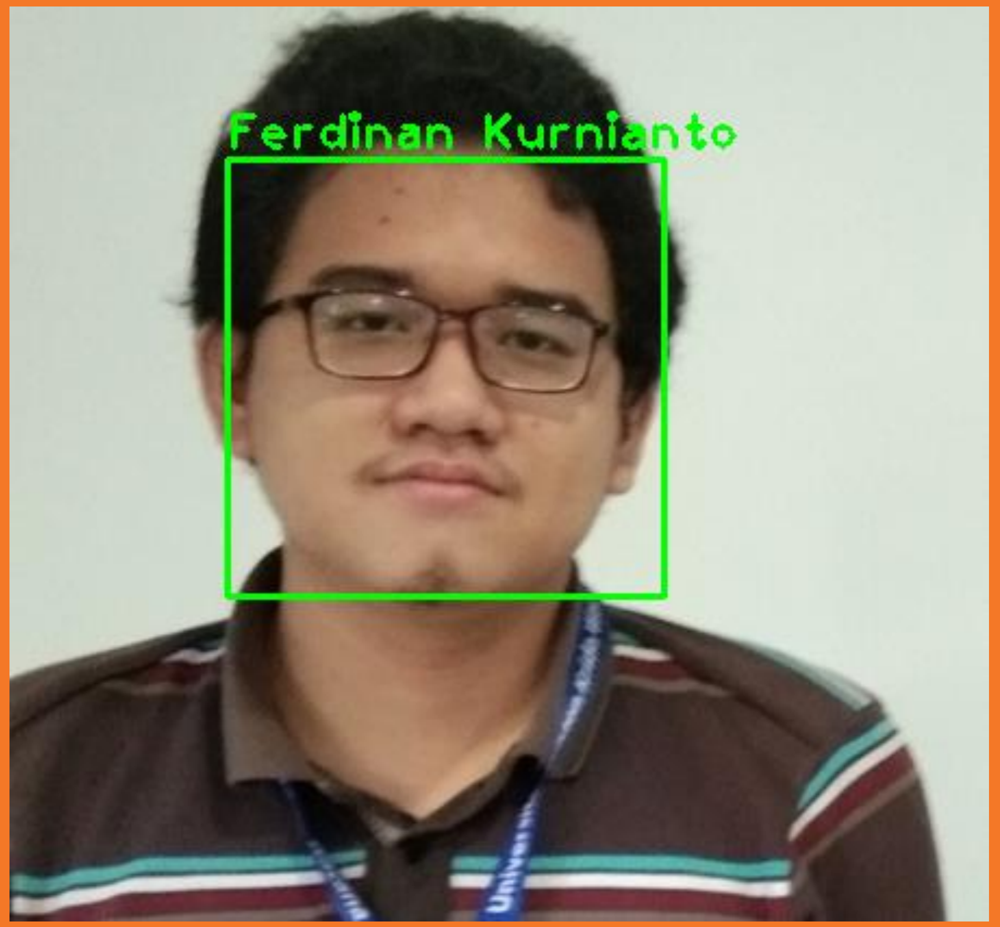


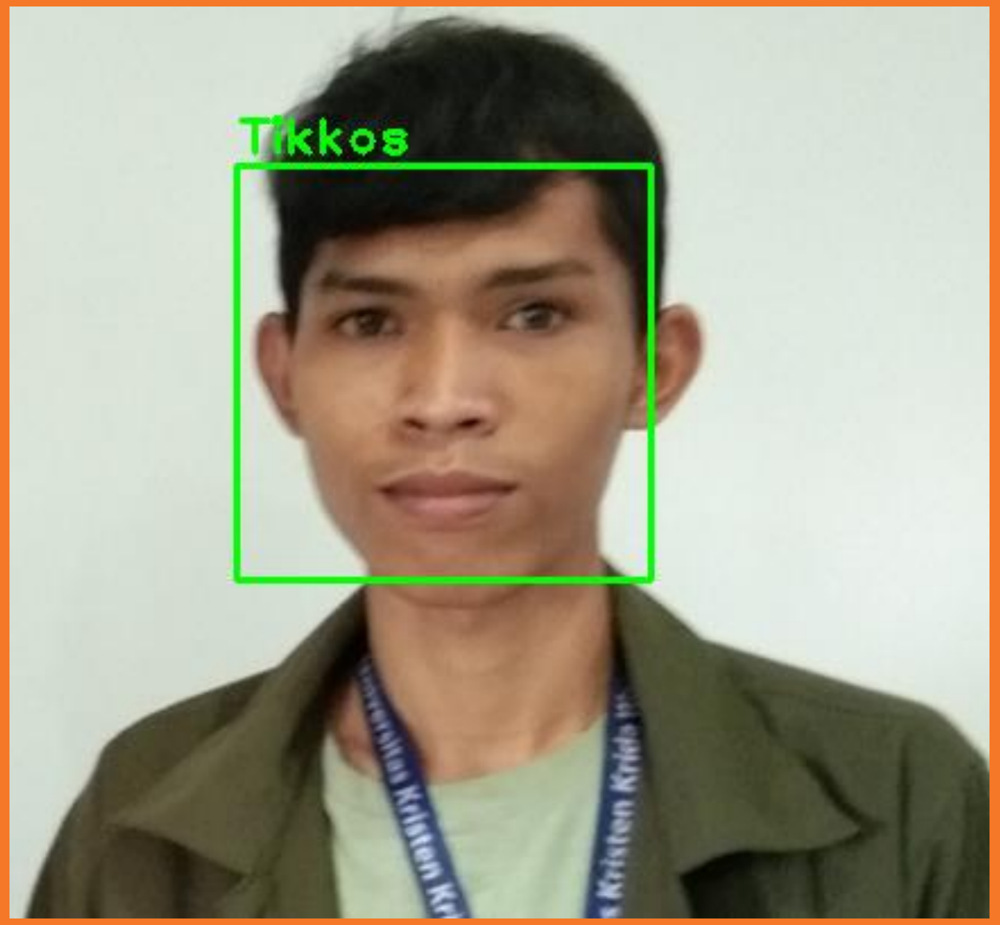
HASIL PENGEMBANGAN



HASIL PENGEMBANGAN







KELEBIHAN DAN KEKURANGAN

+

Kelebihan dari algoritma yang telah kami kembangi adalah dapat mendeteksi dan mengenali sisi wajah dan juga dua atau lebih wajah. Lalu, juga dapat mengcrop hasil dari wajah yang dikenali

−

Kekurangan dari algoritma yang telah kami kembangi adalah kurangnya dataset training yang kami gunakan dan dataset bersifat lokal. Selain itu, pengenalan wajah belum diimplementasi ke dalam sebuah aplikasi dan belum bisa mengenali wajah secara real-time.

IDE-IDE PENGEMBANGAN SELANJUTNYA

1. Yang pertama dapat dilakukan adalah mengimplementasi algoritma ini ke dalam sebuah aplikasi.
2. Selanjutnya, dapat menggantikan dataset dan data test yang digunakan agar lebih besar dan real.
3. Terakhir, dapat dikembangkan pengenalan wajah secara real time melalui video atau face cam.

LINK PROJECT

https://github.com/Ferdinan412019007/PengolahanCitra

THANK YOU