



En place d'un Système de Détection et Reconnaissance Faciale

PROJET DE FIN D'ANNEE

Réalisé par :

Kamal ADDI

Encadré par :

Hicham LAANAYA

Sommaire :

Introduction.....	2
I-Détection des visages dans une vidéo en temps réel :.....	3
II-Création d'une base de données d'images :	4
III-Mise en place de l'algorithme LBPH :.....	6
IV- Entraînement du classifieur :.....	7
V-Détection des visages en temps réel :.....	9
Conclusion.....	11
Bibliographie :.....	12

Introduction :

La reconnaissance faciale est une technologie de plus en plus répandue, basée sur l'intelligence artificielle, permettant d'identifier une personne sur une photo ou une vidéo. un système de reconnaissance faciale dernier a de nombreuses applications en vidéosurveillance, biométrie, robotique, indexation d'images et de vidéos, recherche d'images par le contenu...

Dans ce projet nous allons découvrir les algorithmes et les techniques de détection et reconnaissance des visages, en particulier la détection en utilisent les « haar cascades classifiers ».

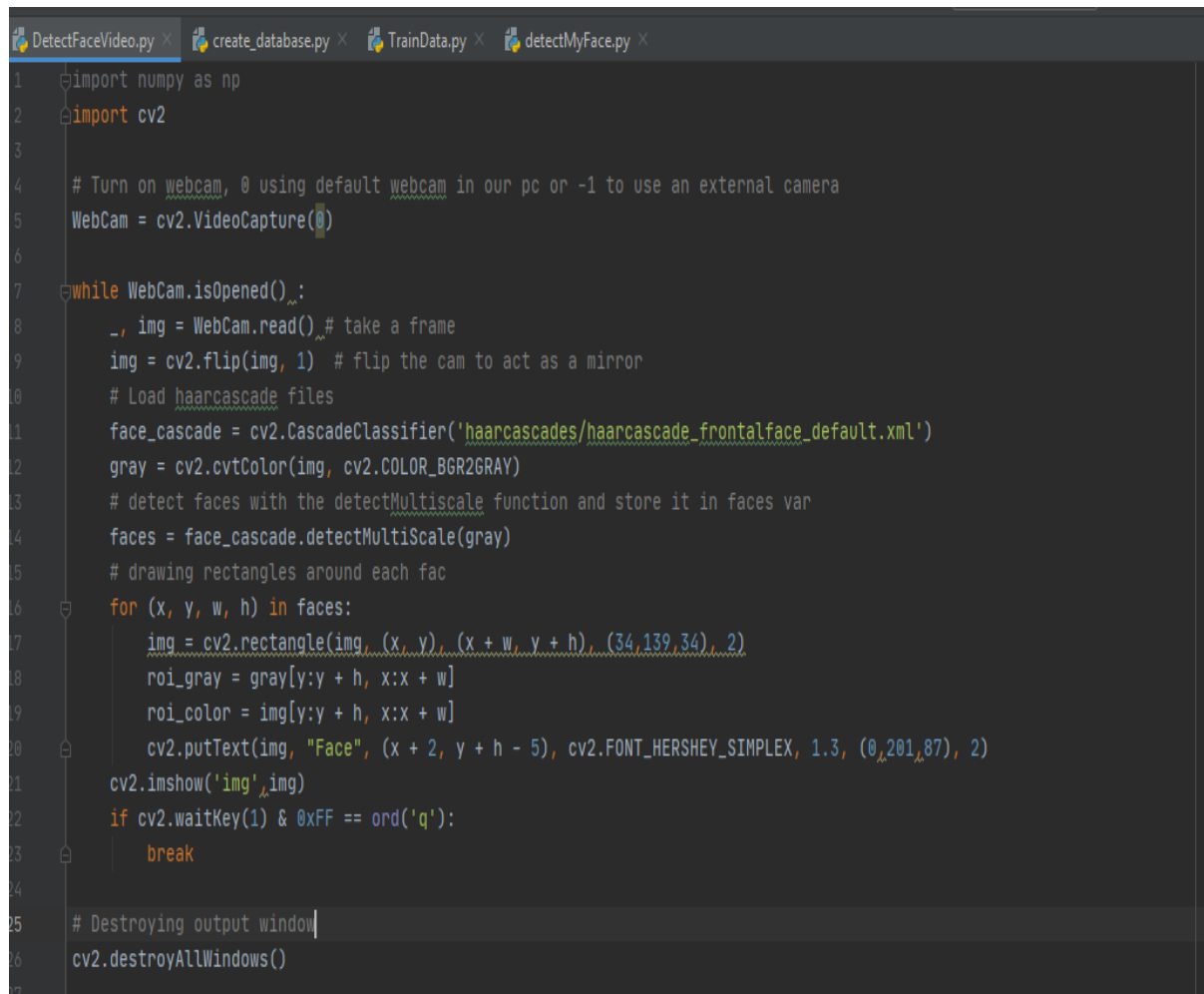
I-Détection des visages dans une vidéo en temps réel :

En utilisant des classificateurs déjà entraînés (disponibles sur GitHub dans les fichiers d'OpenCV, on peut facilement détecter un visage dans une image ou dans une vidéo. Dans ce projet je vais travailler avec le classificateur `haarcascade_frontalface_default.xml`.

Tout d'abord on allume notre caméra avec la méthode `VideoCapture()` avec 0 en paramètre pour utiliser la caméra par défaut dans notre PC.

Après qu'on détecte les visages existant dans une image avec la méthode `detectMultiScale()`, on dessine des rectangles autour de chaque visage :

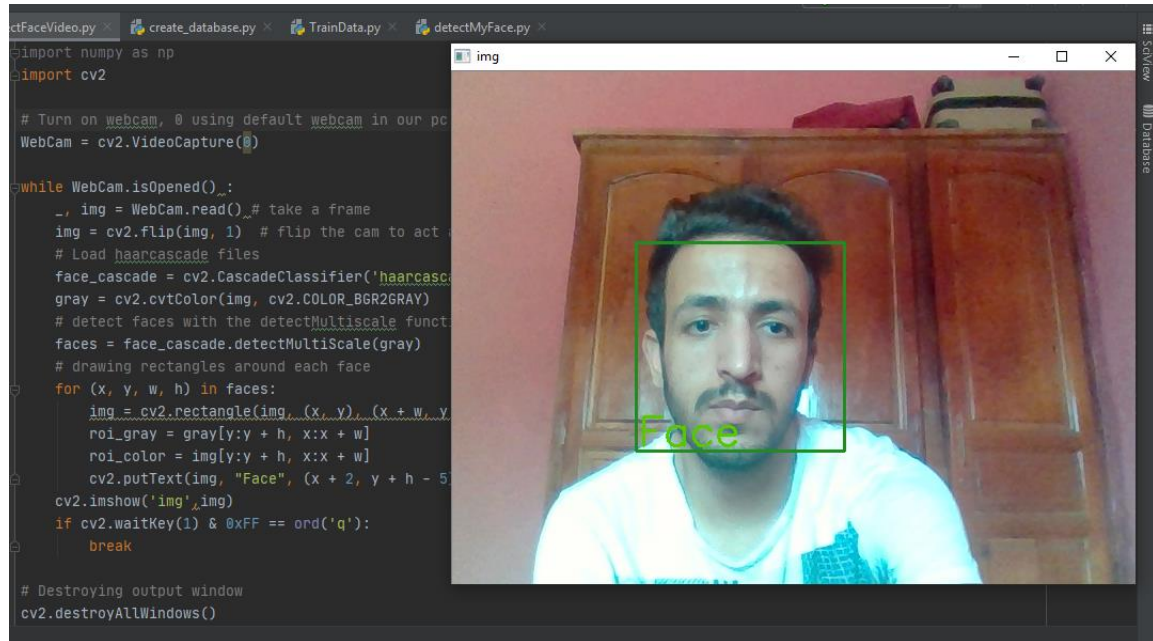
Cette screen montre le programme complet :

A screenshot of a code editor showing a Python script for real-time face detection. The script uses OpenCV and NumPy. It opens a webcam, reads frames, flips them horizontally, converts them to grayscale, and uses a Haar cascade classifier to detect faces. Rectangles are drawn around detected faces, and the word 'Face' is printed on each. The script includes a loop to keep running until the 'q' key is pressed, followed by window destruction.

```
1 import numpy as np
2 import cv2
3
4 # Turn on webcam, 0 using default webcam in our pc or -1 to use an external camera
5 WebCam = cv2.VideoCapture(0)
6
7 while WebCam.isOpened():
8     _, img = WebCam.read() # take a frame
9     img = cv2.flip(img, 1) # flip the cam to act as a mirror
10    # Load haarcascade files
11    face_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
12    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
13    # detect faces with the detectMultiScale function and store it in faces var
14    faces = face_cascade.detectMultiScale(gray)
15    # drawing rectangles around each fac
16    for (x, y, w, h) in faces:
17        img = cv2.rectangle(img, (x, y), (x + w, y + h), (34, 139, 34), 2)
18        roi_gray = gray[y:y + h, x:x + w]
19        roi_color = img[y:y + h, x:x + w]
20        cv2.putText(img, "Face", (x + 2, y + h - 5), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 201, 87), 2)
21    cv2.imshow('img', img)
22    if cv2.waitKey(1) & 0xFF == ord('q'):
23        break
24
25 # Destroying output window
26 cv2.destroyAllWindows()
```

Si on veut quitter on clic la lettre « q ».

On exécute notre programme et voici le résultat :



Maintenant lorsqu'on a vu comment repérer un visage dans une image, on va essayer de reconnaître l'identité de ce visage.

Pour ce faire il nous faut une base de données d'images.

II-Création d'une base de données d'images :

J'ai créé mon base de données d'images à partir d'un vidéo que j'ai filmé avec mon téléphone.

Le but c'est d'extraire un ensemble de visages et les enregistrer dans un dossier qu'on nomme par le nom de la personne correspondant.

On commence à lire notre vidéo dans le répertoire actuel, puis on prend des images et on détecte les visages existant dans ces images avec toujours la méthode `detectMultiScale()`. Et enfin on enregistre le visage dans une image avec l'extension « .png ».

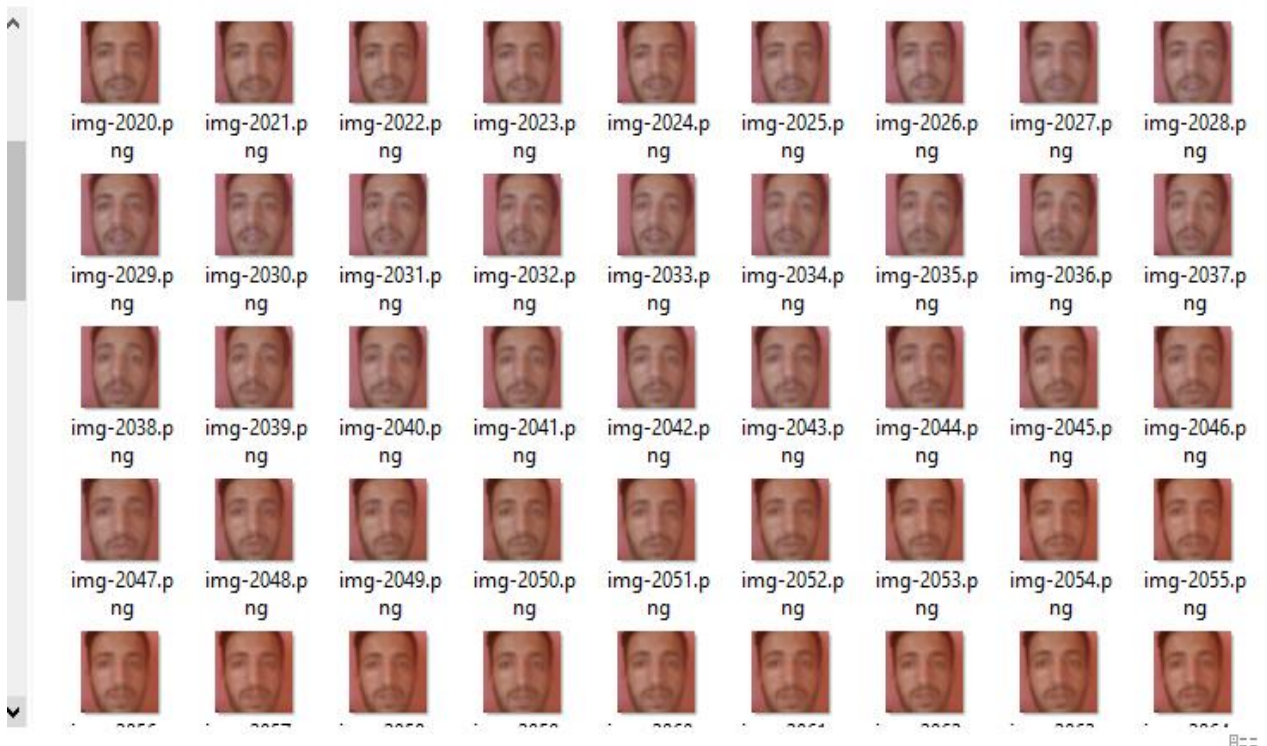
```

ectFaceVideo.py × create_database.py × TrainData.py × detectMyFace.py ×
# import libraries
import cv2

# Turn on my video in the current directory
video = cv2.VideoCapture("MyVideo.mp4")
# Load haarcascade file
face_cascade = cv2.CascadeClassifier("haarcascades/haarcascade_frontalface_alt2.xml")
id = 2020 # initilize id to identify image
while True:
    # take a frame from the video
    _, frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # detect face in our frame
    face = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=4)
    for x, y, w, h in face:
        # store the face with extention .png, here we are going to extract juste the face with frame[y:y+h, x:x+w]
        cv2.imwrite("database/Kamal/img-{}.png".format(id), frame[y:y+h, x:x+w])
        # draw boundary around the face
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
        id+=1
    key = cv2.waitKey(1)
    if key == ord("q"): # press q to finish
        break
    cv2.imshow("video", frame)
video.release()
cv2.destroyAllWindows()

```

L'exécution du programme permet de construire une base de données d'images situé au répertoire « ./database/Kamal » :



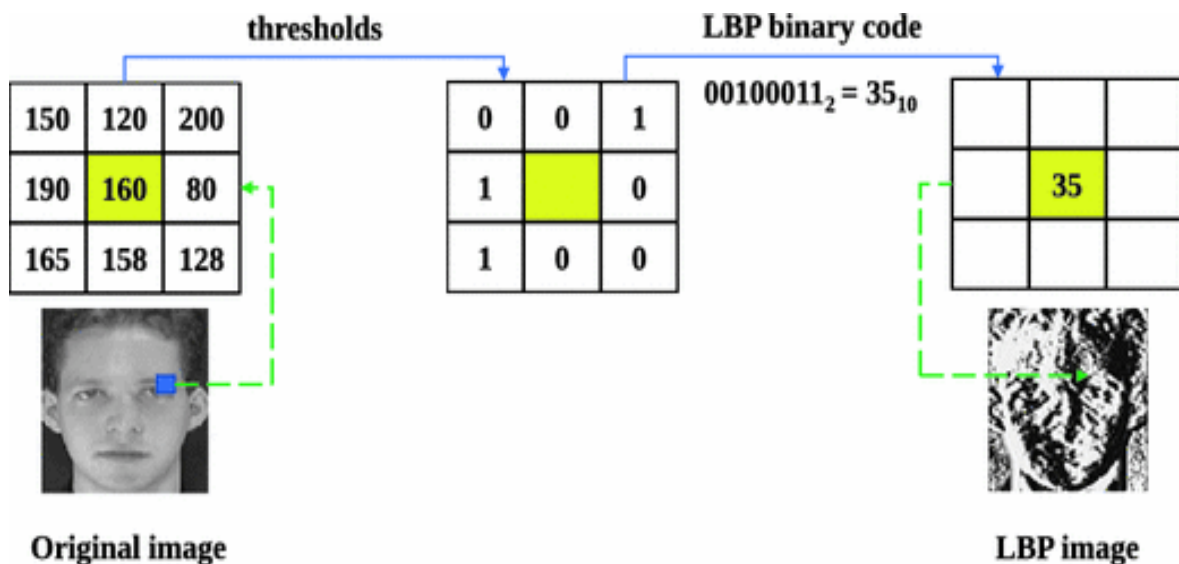
III-Mise en place de l'algorithme LBPH :

Local Binary Patterns Histograms (LBPH) est un algorithme d'apprentissage pour la reconnaissance faciale. Celui-ci se base sur l'analyse de petites régions dans une image et de les caractériser en établissant la relation entre les pixels voisins.

✓ Conception du vecteur caractéristique de LBP :

Pour chaque pixel dans une cellule, comparer le pixel à chacun de ses 8 voisins (sur sa gauche en haut, à gauche du milieu, en bas à gauche, à droite en haut, etc.).

Lorsque la valeur du pixel central est supérieure à la valeur du voisin, écrire « 0 ». Sinon, écrire "1". Cela donne un nombre binaire à 8 chiffres (qui est habituellement converti en décimal pour plus de commodité).



L'opérateur LBP en un point (x_c, y_c) est défini par :

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p * \delta(i_p - i_c)$$

avec i_c, i_p respectivement, le niveau de gris du pixel central et celui de ses voisins et $\delta(.)$ est la fonction de Heaviside.

Et son LH(Local Histograms) défini par :

$$LH_{(x,y)}(l) = \sum_{(x',y') \in S(x,y)} I\{LBP(x',y') = l\}$$

Avec $I(.) \in \{0,1\}$

L'histogramme calculé est pour but de former le descripteur de notre image qui peut être traitée avec les algorithmes d'apprentissage automatique.

Le LBPH a été implémenté sur OpenCV et on peut l'appeler facilement avec la fonction `LBPHFaceRecognizer_create()` sous Python.

IV- Entraînement du classifieur :

Tout d'abord on importe les librairies nécessaires :

Image : pour convertir l'image en greyscale.

Cv2 : pour faire l'appel aux algorithmes `LBPHFaceRecognizer`.

Numpy : pour les opérations sur les tableaux de type array.

Os : pour parcourir le répertoire de notre base de données.

```
TrainData.py
1 # import libraries
2 from PIL import Image
3 import cv2
4 import numpy as np
5 import os
```

Pour la 2ème partie du code nous avons créé une fonction **train_data()** qui prends comme argument le répertoire d'une base de données d'images et génère un fichier d'extension « .yaml » qui contient les caractéristiques des différents visages.

La fonction **LBPHFaceRecognizer_create()** est dédiée à la reconnaissance des visages, et la fonction qui permet de commencer l'entraînement est la fonction **train()** qui prends en paramètres les deux tableaux numpy **list_faces** et **list_ids**.


```

7 def train_data(data_directory): # this function take in parameter the directory of data
8     path = [os.path.join(data_directory, file) for file in os.listdir(data_directory)]
9     list_faces = []
10    list_ids = []
11
12    for image in path:
13        img = Image.open(image).convert("L")
14        imgArray = np.array(img, "uint8")
15        id = int(os.path.split(image)[1].split("-")[1][:4])
16        list_faces.append(imgArray)
17        list_ids.append(id)
18    list_ids = np.array(list_ids)
19    classifier = cv2.face.LBPHFaceRecognizer_create()
20    classifier.train(list_faces, list_ids)
21    classifier.write("MyClassifier.yml")
22    train_data("database/Kamal")

```

Après le lancement de ce programme on va avoir un fichier « MyClassifier.yml » qui contient des caractéristiques des visages, qu'on va pouvoir lire avec des fonctions et identifier des personnes avec ce fichier.

Le fichier MyClassifier.yml contient des informations sur chaque personne dans les images de notre base de données :

```

%YAML:1.0
---
threshold: 1.7976931348623157e+308
radius: 1
neighbors: 8
grid_x: 8
grid_y: 8
histograms:
- !!opencv-matrix
  rows: 1
  cols: 16384
  dt: f
  data: [ 6.66389009e-03, 4.58142441e-03, 0., 8.32986261e-04,
    6.66389009e-03, 8.32986261e-04, 4.16493131e-04, 7.08038313e-03,
    0., 0., 0., 0., 2.91545200e-03, 4.16493131e-04, 4.16493131e-04,
    2.08246568e-03, 5.83090400e-03, 4.16493131e-04, 0., 0.,
    2.49895873e-03, 0., 0., 1.66597252e-03, 6.24739705e-03, 0., 0.,
    0., 3.95668484e-02, 3.74843809e-03, 1.95751768e-02,
    3.54019143e-02, 0., 0., 0., 0., 0., 0., 4.16493131e-04, 0.,
    0., 0., 4.16493131e-04, 0., 0., 8.32986261e-04,
    2.49895873e-03, 0., 0., 0., 0., 0., 0., 9.16284882e-03,
    4.16493131e-04, 0., 0., 9.41274464e-02, 4.16493131e-04,
    2.58225743e-02, 1.16618080e-02, 6.66389009e-03, 0., 0.,
    4.16493131e-04, 1.87421907e-02, 4.16493131e-04, 0.,
    4.16493131e-04, 4.16493131e-04, 0., 0., 0., 1.24947936e-03, 0.,
    0., 4.16493131e-04, 1.66597252e-03, 4.16493131e-04, 0., 0.,

```

Ce fichier va nous permettre par la suite d'identifier des personnes dans une image ou dans une vidéo.

V-Détection des visages en temps réel :

On va importer les deux fichiers `haarcascades_frontaleface_alt2.xml` et « **MyClassifier.yml** » à partir de son répertoire. Puis on démarre la camera de notre pc avec la méthode `VideoCapture(0)` (si on utilise une autre camera externe on va juste prendre en argument le « -1 » pour notre méthode `VideoCapture(-1)`).

```
face_cascade = cv2.CascadeClassifier("haarcascades/haarcascade_frontalface_alt2.xml")
classifier = cv2.face.LBPHFaceRecognizer_create()
classifier.read("MyClassifier.yml")
image_id = 0
video = cv2.VideoCapture(0)
```

Et ensuite on entre dans une boucle infinie, dans laquelle on va prendre des photos et détecter les personnes existant dans ces images et les identifier et mettre un nom et un niveau de confiance sous le visage si il le reconnu et « inconnu » sinon :

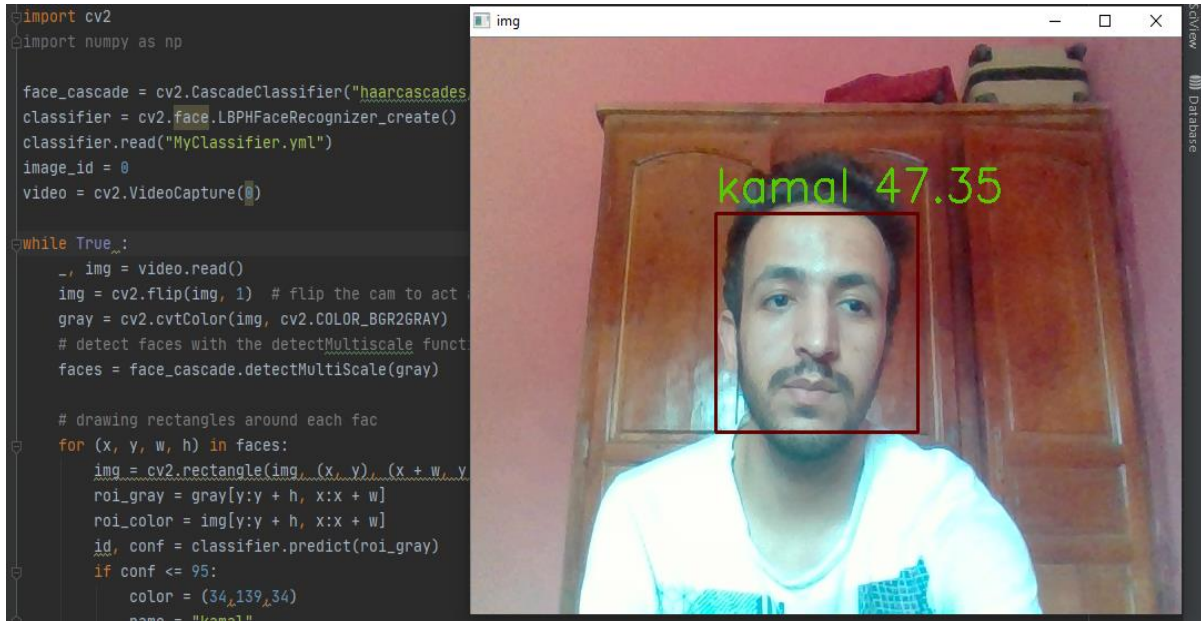
```
while True:
    _, img = video.read()
    img = cv2.flip(img, 1) # flip the cam to act as a mirror
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # detect faces with the detectMultiScale function and store it in faces var
    faces = face_cascade.detectMultiScale(gray)

    # drawing rectangles around each fac
    for (x, y, w, h) in faces:
        img = cv2.rectangle(img, (x, y), (x + w, y + h), (34, 139, 34), 2)
        roi_gray = gray[y:y + h, x:x + w]
        roi_color = img[y:y + h, x:x + w]
        id, conf = classifier.predict(roi_gray)
        if conf <= 95:
            color = (34, 139, 34)
            name = "kamal"
        else:
            color = (100, 154, 244)
            name = "inconnu"
```

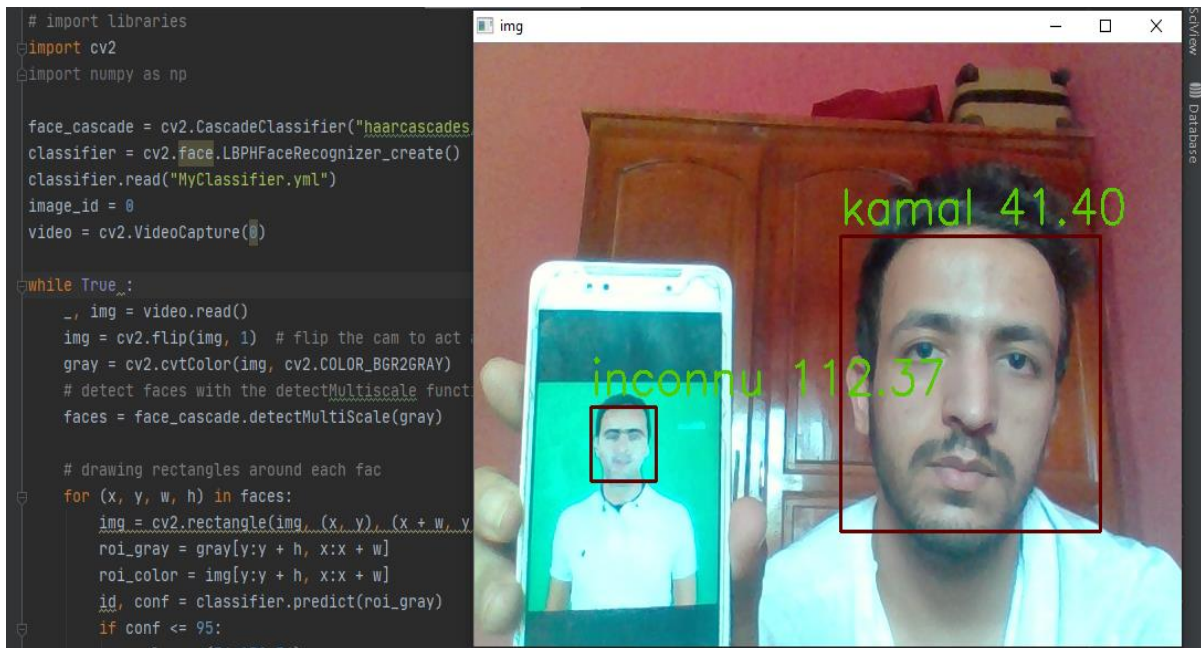
```
label = name + " {:.2f}".format(conf)
cv2.putText(img, label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 201, 87), 2)
cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 100), 2)
cv2.imshow('img', img)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cv2.destroyAllWindows()
```

La méthode **predict()** retourne deux paramètres le id de la photo et le niveau de confiance. Plus le niveau de confiance s'approche à 0 plus

Le système a reconnu mon visage avec une niveau de confiance de 47.35.



S'il détecte une autre visage qui n'existe pas dans la base de données il met inconnu sous ce visage :



Conclusion :

L'objectif de ce projet est d'implémenter un système de reconnaissance faciale en temps réel capable, de reconnaître les visages.

Il existe une variété de techniques consacrées à la détection de visage. Les systèmes de suivi de visage se sont beaucoup développés ces dernières années grâce à l'évolution du domaine de l'IA et du machine learning. Parmi les méthodes de base les plus utilisées on cite : la reconnaissance en utilisant les haar cascade classifier et l'algorithme LBPHFaceRecognizer.

Bibliographie :

[1] : livre : Analysis and Modeling of Faces and Gestures Editors: **Zhou**, S.K., **Zhao**, W.-Y., **Tang**, X., **Gong**, S. (Eds.)

[2] : OpenCV : <https://github.com/opencv/opencv>