Rapport de la séance 1 :

16 novembre 2023

Miri Youssef

Robotique

1-Introduction:

Une des parties les plus importantes dans notre robot, c'est ça capacité à détecter les gens surtout leurs visage, et selon une base de donnée bien défini, prendre la décision de tirer sur personnes définies comme ennemis.

C'est pour cela il était essentielle, de travailler sur cette partie en premier temps qui va être testée d'abord avec un canon Nerf et après avec le canon magnétique.

2-La reconnaissance faciale va être divisée en 3 grandes parties :

A -La reconnaissance faciale de toutes les personnes présentes devant la caméra.

B -La reconnaissance des noms des personnes ainsi que leur statut (allié ou ennemie)

C-Programmer la carte Arduino à recevoir les données de la détection et prendre une décision adéquate.

La première séance va être consacré à la détection de toutes les personnes présentes dans la salle

3- Prérequis :

La reconnaissance faciale et des objets fait toujours partie de la Vision par ordinateur ou « Computer Vision ». Du coup, c'était essentielle d'utiliser python comme langage de programmation.

Donc en premier temps, il faut télécharger python à partir du site officielle python.org, mais aussi les librairies faisant rapport avec la vison par ordinateur.

- La librairie principale utilisée est **OpenCV** (pour Open Computer Vision) est une bibliothèque libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.

- En plus il faut télécharger et ajouter plusieurs librairies qui fonctionnent avec OpenCv , citant Numpy ,cvzone et Haar Cascade.

Les contraintes :

A- Il était essentielle de faire une autoformation en Python pour que je puisse comprendre le syntaxe du langage.

B- Le téléchargement des librairies nécessaires m'a posé beaucoup de problèmes et d'erreurs. Après des recherches, la solution était de passer à la version antérieur de Python de 3.1.2 à 3.0.9

4 -Le choix de librairies :

La détection faciale peut se faire par deux librairies différentes Haar Cascade ou Cvzone .

Mon choix finale de la librairie était Cvzone et cela pour les résultats suivantes.

A -Haar cascade est un algorithme n'est pas si complexe et peut s'exécuter en temps réel. Nous pouvons entraîner un détecteur de cascade de Haar pour détecter divers objets tels que des voitures, des vélos, des bâtiments, des fruits, etc

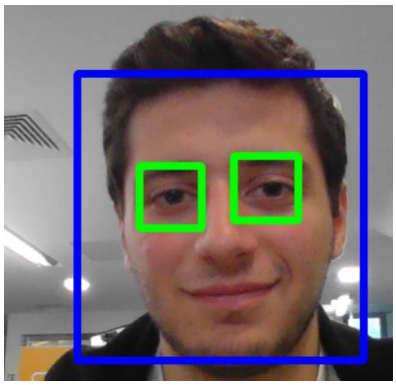
Mais un problème de précision reste toujours présent en utilisant Haar Cascade.

Je vous montre mes résultats ,vérifiant ma conclusion en exécutant le code suivant :

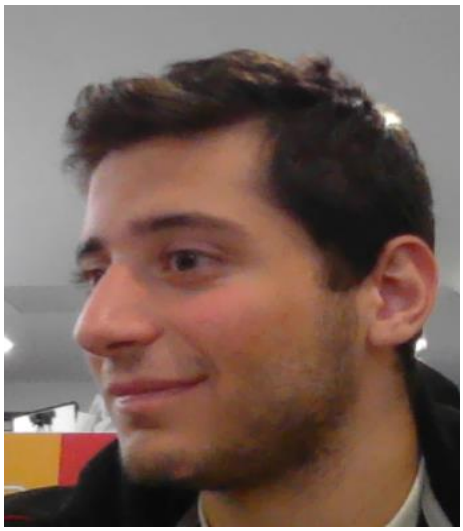
```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

while True:
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 5)
        roi_gray = gray[y:y+w, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray, 1.3, 5)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 5)

    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Avec ce code on détecte les yeux ,et le visage en utilisant face et eye classifiers.

Résultats :



Vue face



Vue côté

Ainsi on remarque que malgré des bonnes résultats dans la figure1, en tournant la tête OpenCv n'a pas pu détecter mon visage.

Alors j'ai utilisé Cvzone comme librairie principale pour la détection faciale et cela après les résultats suivantes :

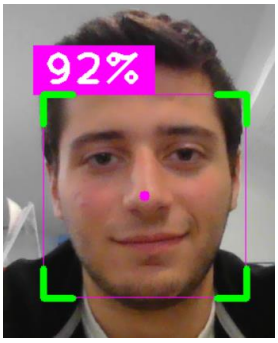
Code :

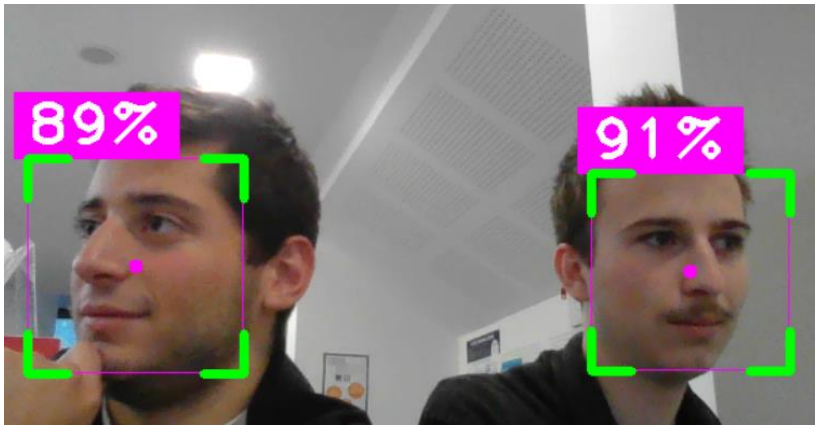
```
 import cvzone

from cvzone.FaceDetectionModule import FaceDetector
import cv2
import os
import numpy as np


# Initialize the webcam
# '2' means the third camera connected to the computer, usually 0 refers to the
built-in webcam
cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')  # codec


fps = cap.get(cv2.CAP_PROP_FPS)
```

```
width  = cap.get(cv2.CAP_PROP_FRAME_WIDTH)   # float
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)  # float
out = cv2.VideoWriter('output7500.avi',fourcc, 30, (640, 480)) #Bonus:saving
video captured in a folder.




# Initialize the FaceDetector object
# minDetectionCon: Minimum detection confidence threshold
# modelSelection: 0 for short-range detection (2 meters), 1 for long-range
detection (5 meters)
detector = FaceDetector(minDetectionCon=0.5, modelSelection=1)

# Run the loop to continually get frames from the webcam
while True:
    # Read the current frame from the webcam
    # success: Boolean, whether the frame was successfully grabbed
    # img: the captured frame

    success, img = cap.read()
    out.write(img)

    # Detect faces in the image
    # img: Updated image
    # bboxs: List of bounding boxes around detected faces
    img, bboxs = detector.findFaces(img, draw=False)

    # Check if any face is detected
    if bboxs:
        # Loop through each bounding box
        for bbox in bboxs:
            # bbox contains 'id', 'bbox', 'score', 'center'

            # ---- Get Data  ---- #
            center = bbox["center"]
            x, y, w, h = bbox['bbox']
            score = int(bbox['score'][0] * 100)

            # ---- Draw Data  ---- #
            cv2.circle(img, center, 5, (255, 0, 255), cv2.FILLED)
            cvzone.putTextRect(img, f'{score}%', (x, y - 10))
            cvzone.cornerRect(img, (x, y, w, h))

    # Display the image in a window named 'Image'
```

```
    cv2.imshow("Image", img)
    # Wait for 1 millisecond, and keep the window open
    if cv2.waitKey(1) == ord('q'):
     break


# Assuming 'frame' is the frame you want to save
# Write out the frame


# Release the VideoWriter when finished
out.release()
cap.release()
cv2.destroyAllWindows()
```

Résultats:





Vue face vue cÔté

On remarque bien que si on tourne la tête, Opencv continue à détecter le visage.

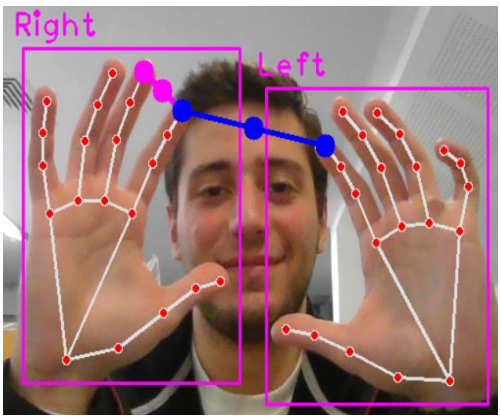
Le pourcentage nous indique le degré de certitude de la détection augmentant ainsi la précision qui est généralement plus que 70%.

*** L'explication du code est fait sous forme de commentaire pour faciliter la compréhension.

Bonus :

J'ai profité de cette séance pour entrer plus en détails dans la compréhension de la vision ordinateur et j'ai ajouté la possibilité de sauvegarder tous ce qui a été filmer dans un fichier bien précis.

J’ai essayé aussi de détecter le mouvement des mains, peut servir plus tard pour le contrôle du mouvement du robot.



Conclusion :

Cette séance était très importante, la première partie de la détection faciale est complète.

La séance suivante va être consacré à la création de la base de donnée.

Références :

Haar Cascade documentation:
https://github.com/opencv/opencv/tree/master/data/haarcascades

Cvzone documentation :

https://github.com/techwithtim/OpenCV-Tutorials/blob/main/tutorial8.py