

FACE POSE ESTIMATION

INDICE

Face pose estimation	1
Introduzione	2
Librerie	3
Il codice	4
Risultati	6

1. INTRODUZIONE

Ciao, benvenuto nella documentazione del mio progetto di Face Pose Estimation.

L'idea alla base di questo progetto commissionatomi dalla Professoressa Veronica Rossano era quella di poter costruire un riconoscitore di volti che potesse anche rilevare la posizione di questo all'interno dello schermo.

Il progetto è interamente in Python 3.9, scritto mediante l'IDE Pycharm.

All'interno della cartella Face Pose Estimation si trovano: la documentazione, un file XML, utile al programma, lo script python commentato e l'eseguibile del progetto

2. LIBRERIE

Per poter realizzare il codice è stato necessario installare la libreria OperCV: il comando da eseguire per poterla installare è ' `pip install opencv-python` '.

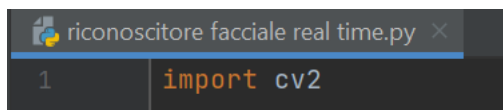
OpenCV è una libreria open source destinata alla Visione Artificiale.

Questa libreria, scritta in C e C++, sviluppata inizialmente da Intel, è disponibile su diverse piattaforme (Windows, Unix, MacOS e ultimamente anche per Android).

Le funzioni messe a disposizione da OpenCV sono circa 500 : elaborazione immagini, tracking e object detection, calibrazione dei dispositivi, estrazione delle feature da un'immagine, riconoscimento di volti e così via.

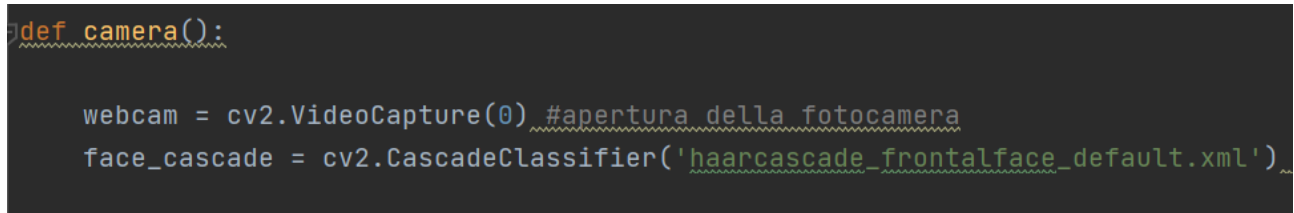
3. IL CODICE

Di seguito riporterò degli screen del codice cercando di spiegare in maniera discorsiva il funzionamento di questo:



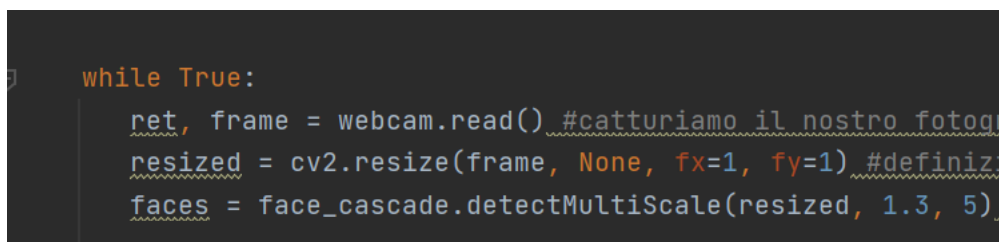
```
1 import cv2
```

Naturalmente dopo aver installato la libreria, la prima cosa da fare è importarla nel nostro codice in modo tale da poterla utilizzare.



```
def camera():  
  
    webcam = cv2.VideoCapture(0) #apertura della fotocamera  
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Andiamo dunque a definire una funzione camera, priva di parametri d'ingresso, all'interno della quale andiamo innanzitutto a creare un oggetto 'webcam', che sarà appunto la video camera che utilizzeremo per rilevare i volti, ed un 'face_cascade' che non è altro che un elemento che 'prenderà in pasto' un file XML potendo così effettuare una classificazione a cascata. Il file XML non fa altro che andare ad addestrare il classificatore per quanto riguarda i volti: verrà dunque attuata una classificazione tramite deep learning grazie all'addestramento effettuato con molti esempi di training basati sul riconoscimento di volti frontali. (N.B: è necessario avere il file XML e lo script python nella stessa cartella).



```
while True:  
    ret, frame = webcam.read() #catturiamo il nostro fotogramma  
    resized = cv2.resize(frame, None, fx=1, fy=1) #definizione della dimensione  
    faces = face_cascade.detectMultiScale(resized, 1.3, 5)
```

All'interno di un while, che idealmente rappresenta la nostra sessione di rilevamento (ovvero il tempo nel quale usiamo la fotocamera), definiamo un 'ret' e un 'frame' mediante la funzione read() della webcam: così facendo cattureremo il

nostro fotogramma dove, frame è appunto il fotogramma e return è una variabile booleana che sarà true nel caso in cui il fotogramma viene rilevato e false viceversa.

Andiamo a definire con 'resized' ciò che sarà la nostra inquadratura ridimensionata ed in faces invece ciò che dovrà essere riconosciuto come un volto (nota bene, passiamo face_cascade che contiene il file XML) e che restituirà dei rettangoli per ogni volto.

```
for (x,y,w,h) in faces: #se trovi un volto in ciò che si trova davanti alla fotocamera
    rettangolo = cv2.rectangle(resized, (x,y),(x+w,y+h),(0,255,0),2) #disegna un rettangolo
    centre_x = (x+w/2) #definizione coordinate utili per poter capire la posizione del volto riscontrato nello schermo
    if(rettangolo < centre_x).all():
        cv2.putText(resized, 'sei a sinistra', (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36, 255, 12), 2) #viceversa, 'sei a destra'
    elif(rettangolo > centre_x).all():
        cv2.putText(resized, 'sei a destra', (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36, 255, 12), 2) #viceversa, 'sei a sinistra'
```

Utilizzeremo un for che sostanzialmente non fa altro che dire: “per ogni riferimento che trovi dei volti” definisci 'rettangolo': l'elemento rettangolo non fa altro che andare a disegnare un rettangolo attorno al viso rilevato.

Definiamo dunque 'centre_x': questa andrà appunto a definire il centro dello schermo: così facendo potremo utilizzare due if per definire appunto la condizione nella quale se il rettangolo si trova a destra, verrà mostrata a schermo la stringa “sei a destra” e viceversa se questo si dovesse trovare a sinistra.

```
cv2.imshow('face pose', resized)
if cv2.waitKey(1) & 0xFF == ord('q'): #premendo q si chiude la finestra
    break
webcam.release() #rilascio della webcam
cv2.destroyAllWindows() #distruzione della finestra
```

Nella parte finale di codice possiamo trovare essenzialmente 3 elementi:

Cv2.imshow ci permette di dare un nome al riquadro dell'applicazione;

l'if ci permette di poter associare un comando per la chiusura dell'app (nel nostro caso basta premere il tasto q);

Ed infine il `webcam.release()` ed il `cv2.destroyAllWindows()` si occupano di rilasciare la webcam e distruggere la finestra dell'applicazione nel momento della chiusura.

4. RISULTATI

Il risultato è il seguente:

