Detectia si Recunoasterea Fetelor

Negoita Tudor
Grupa: 1408A

Bejan Nelu-Adrian
Grupa: 1310B

Furcoi Tase-Isidor
Grupa: 1408B

***Abstract*—Acest document descrie modul de lucru pe care l-am abordat in creearea unei aplicatii in Python pentru recunoastere faciala, folosind OpenCV-Python precum si alter biblioteci**

***Index Terms*—OpenCV-Python, Python, dlib, face_recogintion**

I. Introducere

Detectia si recunoasterea faciala se bazeaza pe identificarea anumitor oameni in functie de niste trasaturi distinctive pe care acestia le prezinta.

Deoarece lumea in care traim se misca din ce in ce mai repede detectia cat mai rapida a anumitor persoane poate fi o unealta foarte utila, sau chiar o necesitate.

II. Motivatie

Dupa cum am spus exista numereoase motive pentru care o aplicatie care sa detecteze persoanele instant are o utilitate clara. In continuare vom enumera cateva dintre ele:

A. Fluidizarea structurilor birocratice

Orice om a simtit macar odata in viata cum este sa fie trimis de la un ghiseu la altul, sa fie nevoit sa semneze un numar mare de hartii intr-un proces ce poate fi destul de frustrant.

Un pas in eliminarea catorva etape din acest proces, cum ar fi completarea datelor personale pentru identificarea respectivului individ ar fi recunoasterea lui cat mai rapida dupa trasaturile fetei, in baza unei poze care poate exista in baza de date a institutiei.

Desigur intr-o astfel de situatie este foarte important ca aplicatia respectiva sa beneficieze de o acuratete cat mai mare, deoarece daca aplicatia nu ar reusi sa diferentieze doi oameni care seamana ar putea aparea diferite inconveniente, care totusi ar putea fi lamurite de angajatii institutiei.

B. Securitate si supraveghere

Detectia si recunoasterea faciala poate fi folosita cu succes si pentru a mentine siguranta cetatenilor. De exemplu, institutiile statului pot detine poze cu indivizi periculosi si se va putea afla locatia acestora, astfel incat in cazul in care ar aparea incidente fortele de ordine ar putea interveni cat mai rapid.

De asemenea, in cazul in care vreun infractor evadeaza din inchisoare acesta ar putea fi recunoscut rapid de orice camera de supraveghere care foloseste acest program si poate fi retinut in cel mai scurt timp.

C. Siguranta in trafic

Aceeasi metoda poate fi folosita si in cazul soferilor care conduc cu o mare viteza. Acestia pot fi identificati cu ajutorul camerelor de supraveghere si penalizati instant.

O astfel de metoda i-ar descuraja pe cei care depasesc viteza regulamentara, reducand astfel considerabil numarul accidentelor.

III. Metode folosite

Pentru a realiza aplicatia de recunoastere faciala am folosit limbajul de programare Python, mai exact IDE-ul PyCharm.

Am utilizat un folder in care am pus poze cu noi, pentru a observa daca aplicatia ne recunoaste pe camera.

A. Biblioteci

Am utilizat urmatoarele biblioteci:

- dlib

- OpenCV

- face_recognition

La inceputul codului am importat urmatoarele pachete:

```
import face_recognition
import os, sys
import cv2
import numpy as np
import math
```

- face_recognition - o biblioteca folosita pentru detectarea si codificarea fetelor.
- cv2 - biblioteca folosita pentru a captura imagini cu ajutorul camerei video.
- math - biblioteca folosita pentru operatii matematice.

B. Functia face_confidence

Apoi am creeat o functie numita face_confidence care ia ca parametri distanta si o valoare threshold care este folosita pentru a determina daca asemanarea este suficient de mare. Functia calculeaza un procentaj de asemanare bazat pe distanta si threshold.

```
def face_confidence(face_distance, face_match_threshold=0.6):
    range = (1.0 - face_match_threshold)
    linear_val = (1.0 - face_distance) / (range * 2.0)

    if face_distance > face_match_threshold:
        return str(round(linear_val * 100, 2)) +'%'
    else:
        value = (linear_val + ((1.0 - linear_val)
            * math.pow((linear_val - 0.5) * 2, 0.2))) * 100
        return str(round(value, 2)) +'%'
```

In cazul in care distanta este mai mare decat threshold-ul functia va returna valoarea reprezentata de variabila linear_val care reprezinta raportul dintre distanta si range-ul posibilelor distante, totul transformat in procentaj.

In cazul in care distanta este mai mica deca threshold-ul functia va returna rezultatul unei functii mai complexe, de asemenea transformat in procentaj.

C. Clasa FaceRecognition

Am creeat o clasa FaceRecognition:

```
class FaceRecognition:
    face_locations = []
    face_encodings = []
    face_names = []
    known_face_encodings = []
    known_face_names = []
    process_current_frame = True
```

1) Field-uri:

- face_locations - locatiile fetei
- face_encodings - codificarile fetei de pe camera
- face_names - numele fetei de pe camera
- known_face_encodings - codificarile fetelor din folder
- known_face_names - numele fetelor din corner
- process_current_frame - un flag care indica daca cadrul curent trebuie procesat

2) Functii:

- Functia encode_faces

  Apoi cream functia ”encode_faces” care foloseste functia face_encodings apartinand bibliotecii face_recognition pentru a asigna codificarile si numele persoanelor din pozele din folder in variabilele known_face_encoding si known_face_names.
  Asignarea se poate observa intre liniile 39 si 43.

```
def encode_faces(self):
    for image in os.listdir('persoane'):
        face_image = face_recognition\
            .load_image_file(f"persoane/{image}")
        face_encoding = face_recognition\
            .face_encodings(face_image)[0]

        self.known_face_encodings\
            .append(face_encoding)
        self.known_face_names\
            .append(image)
    print(self.known_face_names)
```

- Fuctia run_recognition

  Functia run_recognition este functia principala din scriptul nostru, ea ocupandu-se practic de recunoasterea faciala. Deoarece functia este foarte mare o vom aborda pe bucati:
  Cream o instanta a clasei VideoCapture din biblioteca cv2 in variabila video_capture.
  Cu ajutorul ei verificam daca camera este pornita. Daca nu este vom returna mesajul ”Sunt probleme cu camera”.

```
45      def run_recognition(self):
46          video_capture = cv2.VideoCapture(0)
47
48          if not video_capture.isOpened():
49              sys.exit('Sunt probleme cu camera')
```

  Apoi la linia 52 ”ret” este o variabila booleana care ne indica daca metoda ”read()” a putut citi frame-ul, iar ”frame” va contine urmatorul frame care apare la camera.

```
51          while True:
52              ret, frame = video_capture.read()
```

  Vom folosi functia cv2.resize() pentru a micsora frame-ul la un sfert din marimea sa initiala pentru a parcurge operatiile mai rapid.

```
53          if self.process_current_frame:
54
55              small_frame = cv2.resize(frame, (0, 0),
56                                       fx=0.25, fy=0.25)
```

  Cand am folosit biblioteca OpenCV pentru a captura frame-ul, deoarece OpenCV foloseste formatul BGR, acesta nu este compatibil cu biblioteca face_recognition care foloseste formatul RGB. Astfel prin linia 58 inversam imaginea small_frame si o salvam in variabila rgb_small_frame.

```
58              rgb_small_frame = small_frame[:, :, ::-1]
```

  Apoi folosim functiile ”face_locations()” si ”face_encodings()”, aferente bibliotecii face recognition pentru a stoca coordonatele, respectiv codificarile fetelor detectate in frame in field-urile clasei.

```
60              # cautam toate fetele din cadru
61              self.face_locations = face_recognition\
62                  .face_locations(rgb_small_frame)
63              self.face_encodings = face_recognition\
64                  .face_encodings(rgb_small_frame, self.face_locations)
```

  In continuare vom creea variabila booleana ”matches”, in care vom stoca rezultatul aplicarii functiei ”compare_faces”. Aceasta functie din bibiloteca face_recognition va returna o lista de valori True sau False, fiecare atribuita fiecarei fete. True inseamna ca fata din frame se potriveste unei fete din folder, False ca nu.
  In cazul in care nu se potriveste cu nicio fata din folder, variabila name va lua valoarea ”Necunoscut”.

```
66              self.face_names = []
67              for face_encoding in self.face_encodings:
68
69                  matches = face_recognition\
70                      .compare_faces(self.known_face_encodings, face_encoding)
71                  name = "Necunoscut"
72                  confidence = '???'
```

  Folosim functia ”face_distance()” din biblioteca face_recognition pentru a compara codificarile pozelor in folder si cele ale pozelor din frame. Aceasta metoda returneaza un sir de ”distante”, cu cat ”distanta” este mai mica, cu atat sunt mai asemanatoare respectivele codificari.
  Apoi cu ajutorul functiei ”argmin()” stocam distanta minima in variabila ”best_match_index” si asignam numele pozei din folder fetei respective.

De asemenea folosim functia face_confidence pentru a stoca in variabila ”confidence” un procentaj de similitudine intre cele doua poze pe care de asemenea il asignam fetei din frame.

```
76                  face_distances = face_recognition\
77                      .face_distance(self.known_face_encodings, face_encoding)
78                  best_match_index = np.argmin(face_distances)
79
80                  if matches[best_match_index]:
81                      name = self.known_face_names[best_match_index]
82                      confidence = face_confidence(face_distances[best_match_index])
83
84                  self.face_names.append(f'{name} ({confidence})')
```

Apoi la linia 86 negam variabila ”process_current_frame” pentru a alterna frame-urile procesate

```
86              self.process_current_frame = not self.process_current_frame
```

Apoi cream un loop pentru a parcurge in simultan listele face_locations si face_names si pentru a construi un cadru in forma de patrat in jurul fiecarei fete din frame.
Functia ”zip()” este folosita pentru a creea perechi formate din elementele celor doua liste care au aceeasi indecsi.
”top”, ”right”, ”bottom” si ”left” reprezinta niste coordonate returnate de functia face_locations() sub forma de tuple de cate patru elemente.

```
88              for (top, right, bottom, left), name in\
89                      zip(self.face_locations, self.face_names):
```

Apoi dam resize la frame inmultind fiecare coordonata cu 4.

```
91                  top *= 4
92                  right *= 4
93                  bottom *= 4
94                  left *= 4
```

Construim patratele cu ajutorul functiei ”rectangle()” din biblioteca cv2, care ia ca argumente frame-ul, coordonatele coltului din stanga sus si a celui din dreapta jos, culoarea si grosimea laturilor patratului ca argumente.
Numele persoanei localizate in poza, precum si procentajul de similitudine sunt afisate cu ajutorul functiei ”putText()” din biblioteca cv2.

```
97                  cv2.rectangle(frame, (left, top),
98                                (right, bottom), (102,205,170), 2)
99                  cv2.rectangle(frame, (left, bottom +20),
100                               (right, bottom), (102,205,170), cv2.FILLED)
101                 cv2.putText(frame, name, (left + 3, bottom +14),
102                             cv2.FONT_HERSHEY_DUPLEX, 0.4, (0,0,0), 1)
```

Apoi afisam frame-urile capturate prin camera video cu ajutorul functiei imshow(). Pentru a opri programul va trebui sa apasam pe litera q pe tastatura. Pentru a utiliza camera video folosim functia video_capture.release().

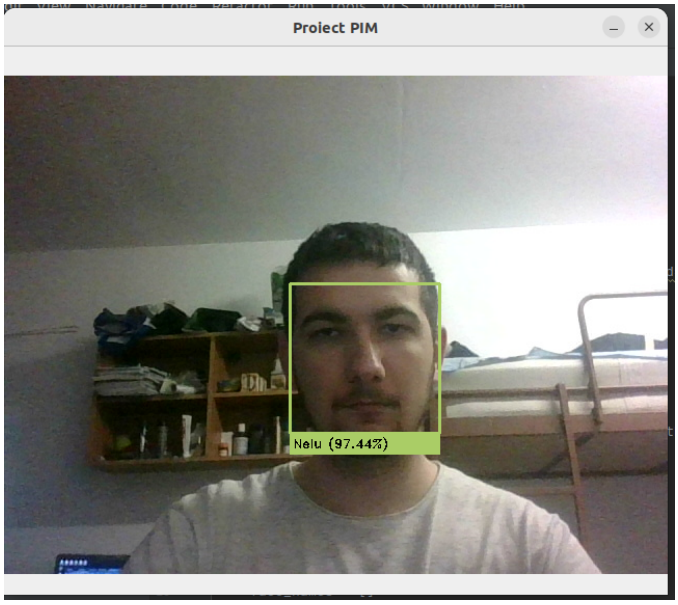
```
105             cv2.imshow('Proiect PIM', frame)
106             if cv2.waitKey(1) == ord('q'):
107                 break
108         video_capture.release()
109         cv2.destroyAllWindows()
```

IV. Rezultate

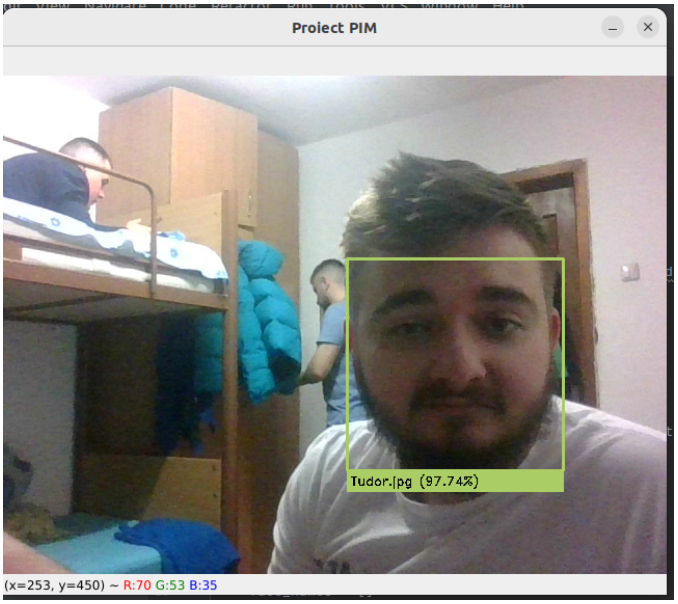
Pentru testarea programului cate o poza cu fiecare dintre noi pe care le-am incarcat in folder-ul ”imagini” si am testat mai multe situatii:

A. *O persoana din baza de date*

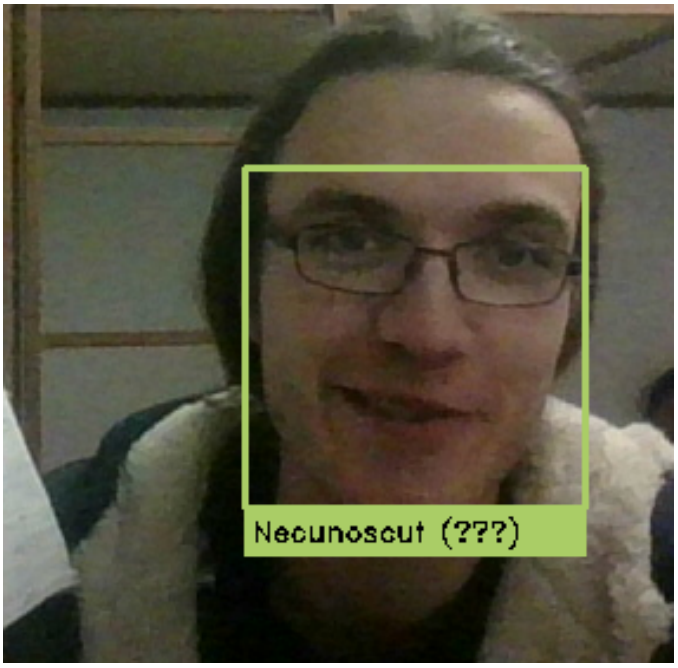
- Nelu



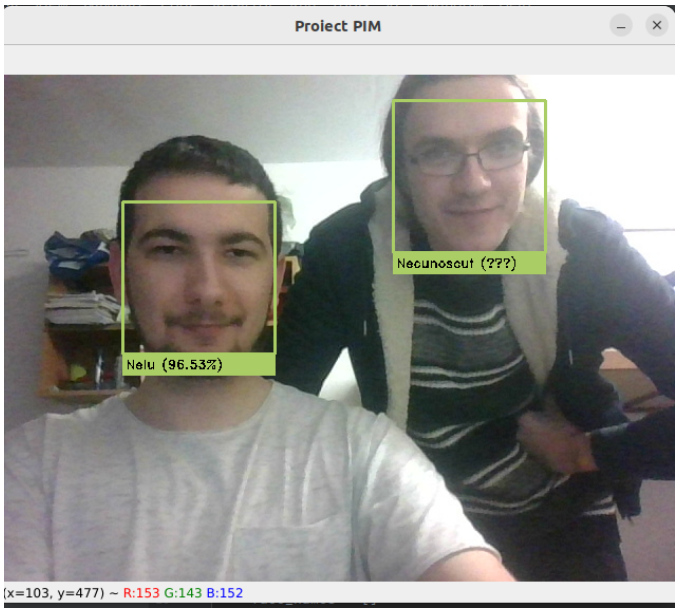
- Tudor



B. O persoana necunoscuta



C. O persoana din baza de date si o persoana necunoscuta



https://pypi.org/project/face-recognition/

REFERENCES

[1] https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
[2] http://dlib.net/python/index.html
[3] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.