Voorbereiding

Voor het detecteren van gezichten, maken we gebruik van de module face_recognition. Deze module heeft een goed getraind model op basis van Histogram of Gradient (HoG), hier zo meer over.

Installatie DLIB

Voor de installatie van face_recognition moeten we een uitgebreide library voor machine learning, DLIB, installeren. Deze library is geschreven voor linux en MacOS, maar is ook in windows te gebruiken. Het installeren kan een uitdaging zijn, tenzij je de instructies goed en *in de juiste volgorde* opvolgt:

Voor het detecteren van objecten en gezichten installeren we de volgende libraries:

Voor Windows: Installeer eerst de Visual Studio C++ Build Tools: https://visualstudio.microsoft.com/visual-cpp-build-tools/ Zorg dat je de volgende opties in elk geval aan hebt staan:

- MSVC v142 – VS 2019 C++ x64/86 build tools
- Windows 10 SDK
- C++ Cmake tools for windows

Alle OS:

pip install cmake

pip install dlib

pip install face_recognition

Laten we Gezichts Features Detecteren

1. Download practicum3.zip op de DLO -> Opdrachten -> week 3.
2. Dit is een variatie op het gezichtsdetectie programma wat we in week 1 hadden gemaakt. Hierin gebeurt het basiswerk, webcam uitlezen & wegschrijven van de data. Elke keer als je op de spatiebalk drukt (even ingedrukt houden), wordt een gezicht gedetecteerd en inclusief features weggeschreven.

3. Allereerst gaan we de variabelen aanpassen voor ons script. Ga naar regel 19 en pas de code aan.

```
#PAS DEZE AAN
base_folder =  'C:/temp'
naam = "alize"
show_frame = False
```

base_folder: de output folder waar je naartoe wilt schrijven.
Naam : geef hierbij jouw naam op. Volgende week lezen we hieruit welke gezicht bij wie hoort, dit is dus belangrijk!
show_frame: Dit is voor jou ter controle. Als je deze waarde op true zet, wordt het webcam frame met daarop alle ingetekende features aan jouw getoond.

4. Nu gaan we de face_recognition module gebruiken om meer features te detecteren:

```
import face_recognition
```

5. Ga in nu naar de functie gebruikface_detect en daaronder gaan we de benodigde code plaatsen om gezichten te detecteren.

```
face_bounding_boxes = face_recognition.face_locations(frame)
nr_gezichten = len(face_bounding_boxes)
print(f"er zijn {nr_gezichten} gezichten gevonden")
```

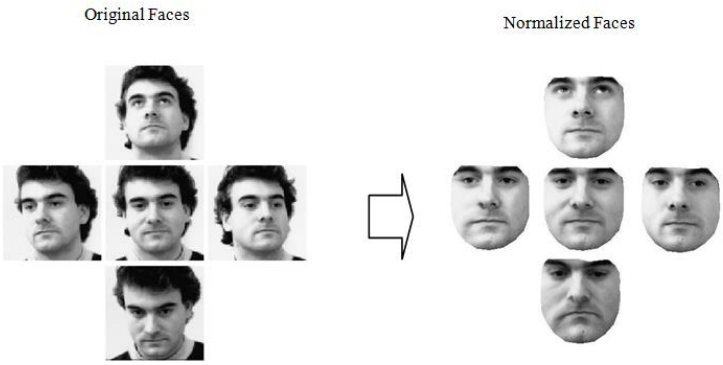
6. De variabele **face_bounding_boxes** die wordt teruggegeven bevat een lijst met de rechthoeken met de ROI voor gezichten. We kunnen dit controleren door de ROI van het gezicht te bekijken met de volgende stap:

```
x1 = left
y1 = top
x2 = right
y2 = bottom

#haal de ROI van het gezicht op
roi_gezicht = frame[y1:y2, x1:x2]
cv.imwrite(base + "_gezicht.png", roi_gezicht)
```

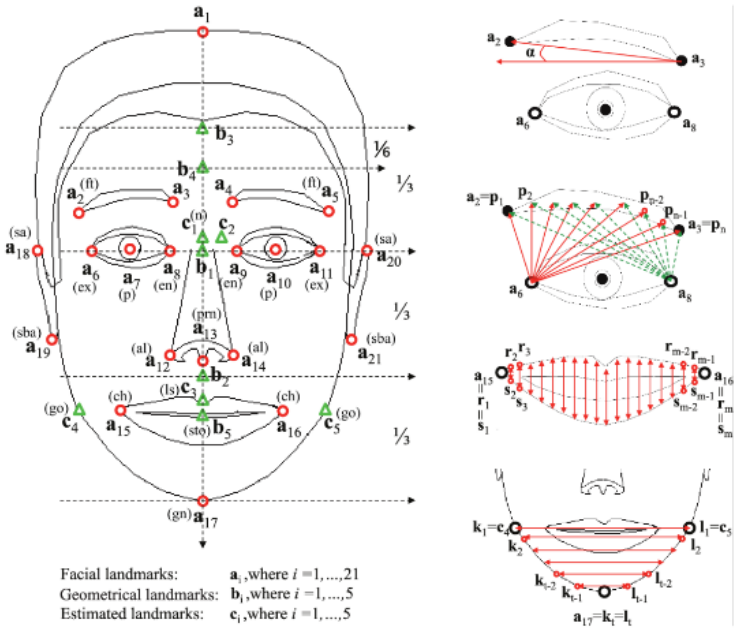
7. Tot nu toe hebben we precies hetzelfde als we bij de haarclassifier ook al hadden, daar hebben we al die moeite om dlib te installeren niet voor gedaan, we willen meer features! Gelukkig kan dat. We gaan nu alle gezichtspunten detecteren en deze in het frame tekenen:

```
#haal de landmarks van dit gezicht op
face_landmarks = face_recognition.face_landmarks(frame, [(top, right, bottom, left)])

#loop over deze landmarks
for structure, punten in face_landmarks.items():
    # teken deze punten ook alvast in het frame
    for punt in punten:
        cv.circle(frame, punt, 2, kleur_wit, 1)
```

8. Maar hoe kunnen we deze nu vergelijken met andere gezichten? Nou dat doen we door locatie en transformatie invariante afstanden uit de gezichtspunten te halen.
    a. We normaliseren de gezichten (we zetten ze alle gezichten dezelfde kant op met de neuzen in het midden):



b. We bereken de afstanden tussen de verschillende gezichtspunten en dan de verhoudingen daartussen:

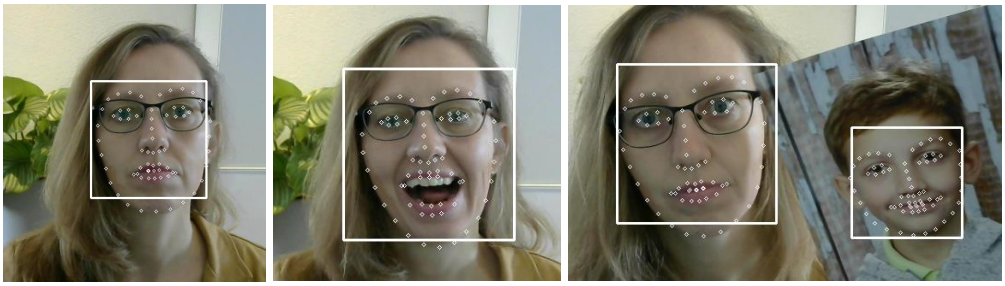


c. We normaliseren die verhoudingen in getallen tussen de 0 en 1.

In code komt dat neer op 1 regel python + 1 regel om de features naar een file te schrijven (die gaan we volgende week gebruiken voor classificatie:

```
feature_list = face_recognition.face_encodings(frame, [(top, right, bottom, left)])[0]
np.save(base + '_feature.npy', feature_list)
```

9. Run de code en zie of het werkt.
   Als je je het goed gedaan hebt, worden de punten goed op je gezicht gelegd, zie hieronder hoe het eruit moet zien



Ook heb je in jouw base_folder een nieuwe folder met jouw naam waarin voor elke detectie een file genaamd:

- frame_{1:40}.png
- frame_{1:40}_{0:?}_feature.npy
- frame_{1:40}_{0:?}_ gezicht.png

De {1:40} telt hierbij het aantal frames dat je hebt opgeslagen.
de {0:?} telt het aantal gezichten dat in elk frame te vinden is (zie het voorbeeld hierboven), daar worden in 1 frame 2 gezichten gedetecteerd.

10. Als alles klopt, zorg dat je elk geval **<u>10 frames</u>** van *jouw eigen hoofd* (en dus niet van jouw huisgenoot erbij 😉) maakt, met bijbehorende features en frames.

11. Doe vervolgens hetzelfde voor minimaal 1 huisgenoot. Hoe meer hoe beter uiteraard.
    LET OP: pas dan wel de naam aan in regel 20 in je script. Zo maak je , net als facebook, een database met gezichten aan. Op jouw computer heb je dan de volgende structuur:

```
<temp_folder>/
    <jouw_naam>/
        frame_1.png
        frame_2.png
        .
        .
        frame_n..png
   <naam__huisgenoot>/
       frame_1.png
       frame_2..png
        .
        .
       frame_n.png
    .
    .
```

Disclaimer: Je gaat de gezichten van jouw huisgenoten niet met de klas delen, die blijven op jouw computer staan.