
Drowsiness Detection :
970122680025
970122680010
970122680007

Library های مورد استفاده :

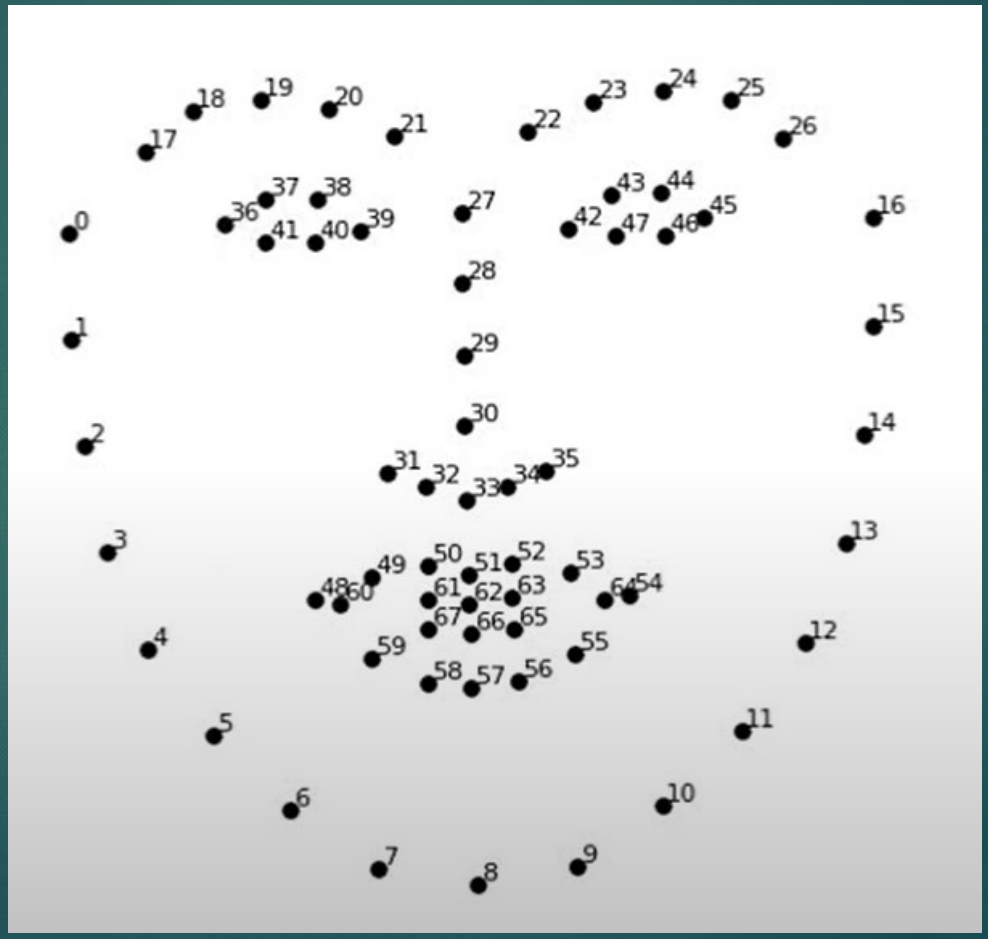
```
import cv2
import dlib
from scipy.spatial import distance as d
```

- کتابخانه ی opencv
- کتابخانه ی dlib برای تشخیص facial land mark ها
- پکیج scipy به منظور محاسبه ی فاصله ی اقلیدسی بین facial landmark های چشم

```
cap = cv2.VideoCapture(0)
detector = dlib.get_frontal_face_detector()
face_landmarks = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

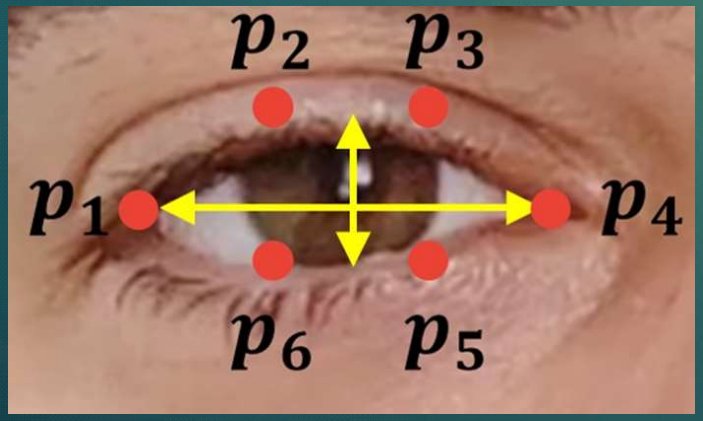
- از کتابخانه ی opencv برای دریافت تصویر از وب کم استفاده شده.(۰ شماره ی وب کم و به معنای وب کم اصلی است)
- از کتابخانه ی dlib با عنوان detector برای تشخیص چهره های موجود در تصویر استفاده میشود.
- از فایل دیتا با عنوان shape_predictor_68_face_landmarks برای یافتن landmark های چهره استفاده میشود.

Facial landmarks:



```python
# calculate aspect ratio
def aspect_ratio_cal(eye):
    AR = (d.euclidean(eye[1], eye[5])+  d.euclidean(eye[2], eye[4])) / (2.0 * d.euclidean(eye[0], eye[3]))
    return AR
```

- از این تابع برای محاسبه ی eye aspect ratio استفاده شده است.
- محاسبه ی این رابطه به صورت زیر میباشد :



$$\frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

چرا از این رابطه استفاده میشود؟

Well, as we'll find out, the eye aspect ratio is approximately constant while the eye is open, but will rapidly fall to zero when a blink is taking place.

Using this simple equation, we can *avoid image processing techniques* and simply rely on the *ratio of eye landmark distances* to determine if a person is blinking.

```
while cap.isOpened():
    # capture frames
    ret, frame = cap.read()
    gry = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gry)
```

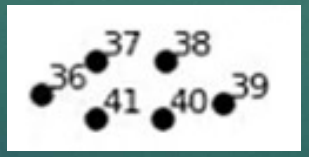
- تا هنگامی که وب کم فعال است حلقه تکرار میشود.
- تصویر دریافت میشود و با استفاده از detector که قبلا تعریف شده بود؛ چهره های موجود در تصویر دریافتی از وب کم به صورت real time شناسایی میشوند.

```
for face in faces:
    lmarks = face_landmarks(gry, face)
    # array for the right eye
    R = []
    # array for the left eye
    L = []
    # number of next node
    next = 0
```

- به ازای هر چهره ی موجود در تصویر مراحل زیر انجام میشوند :
- لند مارک های چهره در lmarks ذخیره میشوند.
- دو آرایه به صورت جداگانه برای چشم چپ و راست تعریف میشوند که در آینده با مختصات هر کدام از لندمارک های چشم پر میشوند.
- متغیر next به عنوان نقطه ی بعدی است و در اسلاید های بعد توضیح داده میشود.

```python
for n in range (36, 42):
  # filling the left eye array
    L.append((lmarks.part(n).x ,lmarks.part(n).y))
  # find the number of the next node
    next = n + 1
  # for the last node
    if n == 41:
      next = 36
  # draw a line between node and next node
    cv2.line(frame, (lmarks.part(n).x, lmarks.part(n).y), (lmarks.part(next).x ,lmarks.part(next).y), (255, 0, 0), 1)
```

- در این حلقه n از ۳۶ تا ۴۱ تغییر میکند که شماره های لندمارک های مربوط به چشم چپ است(طبق شکل)



- این حلقه ۲ وظیفه ی مهم و اصلی دارد :
- ۱- پر کردن آرایه ی مربوط به مختصات چشم چپ (L)

```
# filling the left eye array
  L.append((lmarks.part(n).x ,lmarks.part(n).y))
```

- ۲- کشیدن خط دور چشم :

با داشتن مختصات نقطه ی فعلی و نقطه ی بعدی(next) و وصل کردن آن ها به یکدیگر خطی دور چشم رسم میشود.

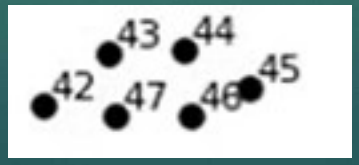
[به استثنای نقطه ی آخر که باید به نقطه ی اول متصل شود]

```
# find the number of the next node
  next = n + 1
# for the last node
  if n == 41:
    next = 36
# draw a line between node and next node
  cv2.line(frame, (lmarks.part(n).x, lmarks.part(n).y), (lmarks.part(next).x ,lmarks.part(next).y), (255, 0, 0), 1)
```

```python
for n in range (42, 48):
    # filling the right eye array
    R.append((lmarks.part(n).x ,lmarks.part(n).y))
    # find the number of the next node
    next = n + 1
    # for the last node
    if n == 47:
        next = 42
    # draw a line between node and next node
    cv2.line(frame, (lmarks.part(n).x, lmarks.part(n).y), (lmarks.part(next).x ,lmarks.part(next).y), (255, 0, 0), 1)
```

- در این حلقه n از ۴۲ تا ۴۷ تغییر میکند که شماره های لندمارک های مربوط به چشم راست است(طبق شکل)

و بقیه ی مراحل مشابه بلوک قبلی کد مربوط به چشم چپ است.

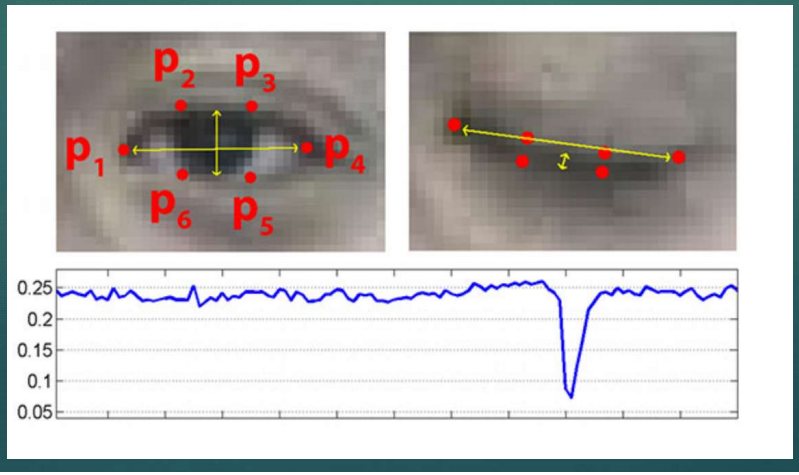


تکه کد مربوط به تشخیص خواب آلودگی :

```
# if the eyes are closed (the average of aspect ratios are less than 0.20)
if round((aspect_ratio_cal(L) + aspect_ratio_cal(R))/2, 2) < 0.20:
    # detect the drawsiness and print on the screen
    cv2.putText(frame,"Drowsiness Detected", (175, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,0), 3)
```

- در این قسمت از کد در شرط مربوط به تشخیص بسته بودن چشم و خستگی؛ eye aspect ratio مربوط به هر چشم به صورت جدا گانه محاسبه شده و سپس میانگین گرفته میشود.

در صورتی که این مقدار کمتر از ۰,۲۰ باشد؛ عبارت Drowsiness Detected روی تصویر نمایش داده میشود.



خروجی کد :

در صورت باز بودن چشم :

خروجی کد :

در صورت بسته بودن چشم :