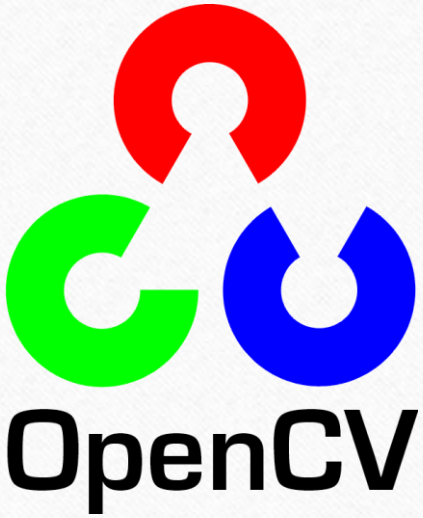
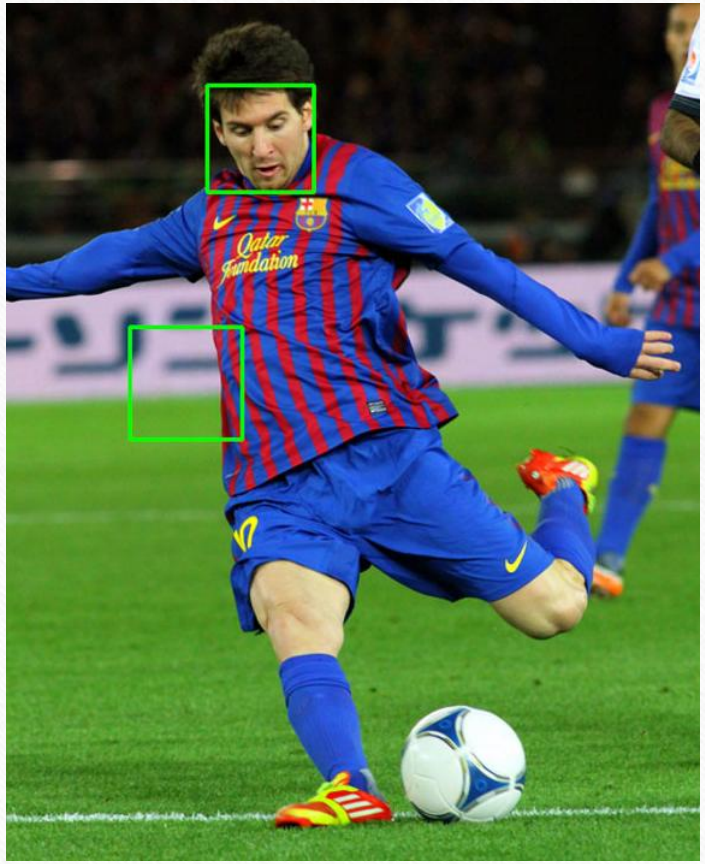
دوره آموزشی بینایی ماشین کاربردی

آکادمی ربوتک – آزمایشگاه تعامل انسان و ربات

جلسه 6 – آشنایی با کتابخانه Dlib



تشخیص چهره به کمک OpenCV



راه 1 : استفاده از HAAR Cascade

Rapid Object Detection using a Boosted Cascade of Simple Features

این الگوریتم در OpenCV پیاده سازی شده است.

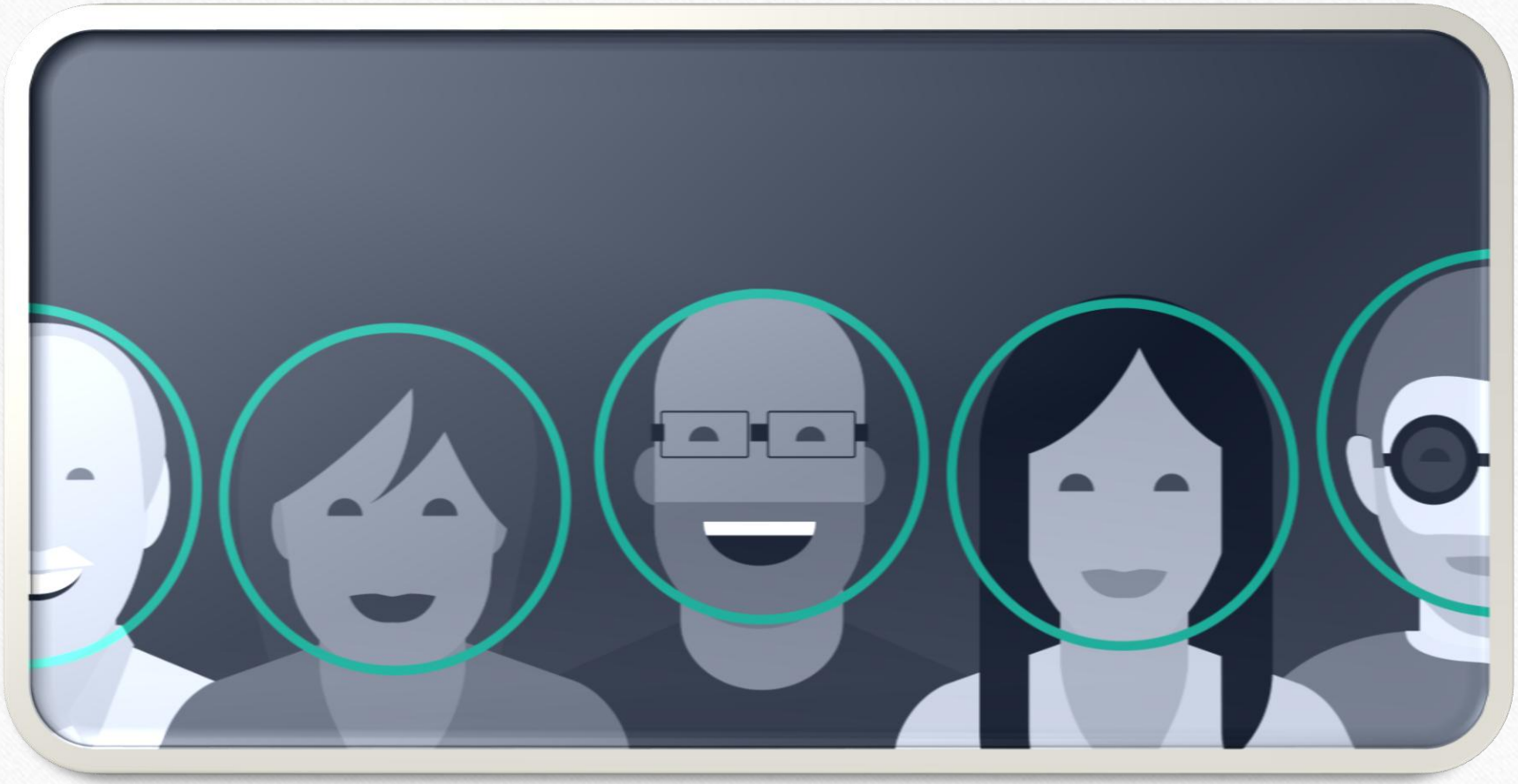
راه 2 : کتابخانه Dlib

یک کتابخانه از پیش Train شده HOG + SVM

کتابخانه Dlib

- معرفی شده توسط Davis King در سال 2002
- ارایه مقاله 2009 : Dlib-ml: A Machine Learning Toolkit
- قابل استفاده در محیط های صنعتی و آکادمیک
- دارای Documentation بسیار قوی

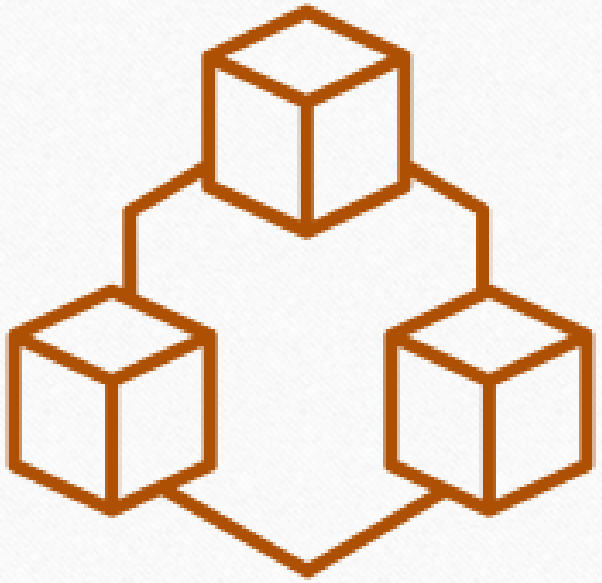
پروژه 1 : تشخیص صورت و اجزای آن به کمک Dlib



گام 1 : اضافه کردن ماژول های مورد نیاز

```
from imutils import face_utils
import imutils
import dlib
import cv2
```

از face_utils به منظور برخی تبدیلات مرتبط با Dlib استفاده میکنیم.



گام 2: تعریف Object های مربوط به detection و prediction

```
detector = dlib.get_frontal_face_detector()

predictor = dlib.shape_predictor("آدرس فایل")
```

فایل از پیش Train شده را کنار کد خود قرار دهید و نام آن را در قسمت ذکر شده بیاورید.

گام 3: خواندن تصویر و انجام پیش پردازش های اولیه

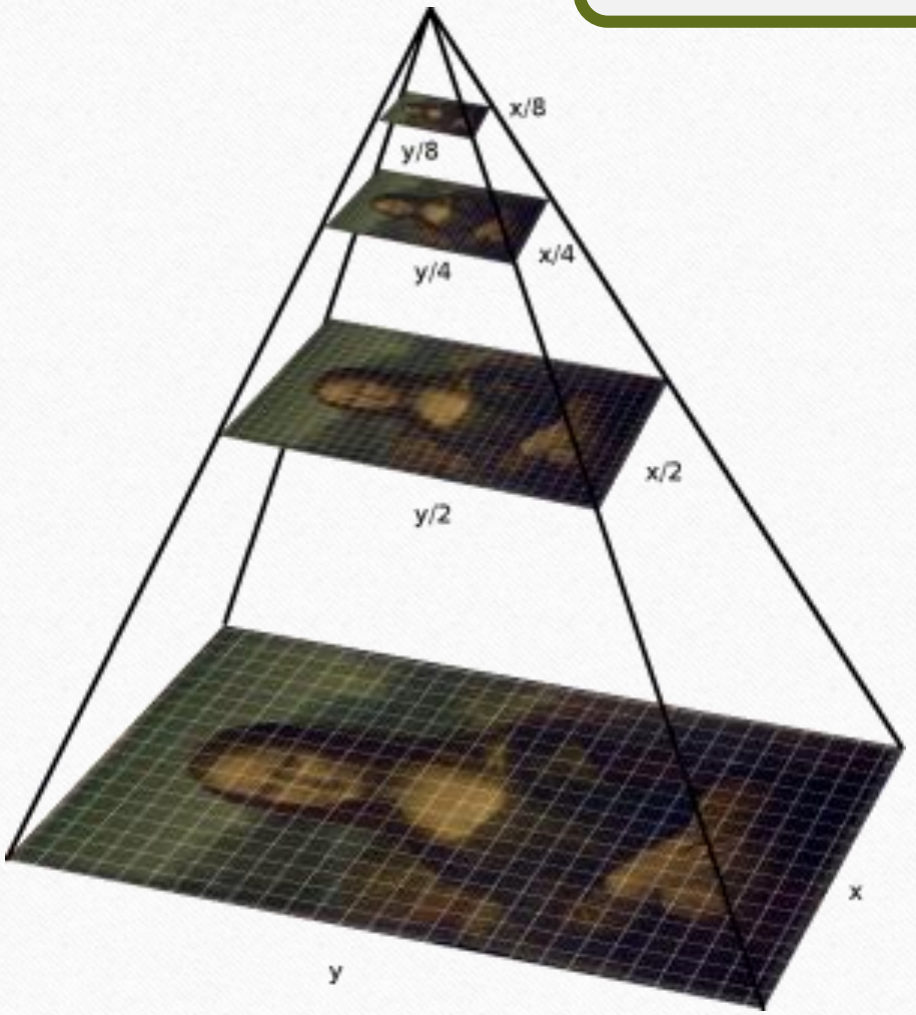
شامل :

1- تغییر اندازه تصویر

2- histogram equalization

3- تبدیل تصویر به تصویر gray

گام 4: پیدا کردن صورت ها در تصویر



ورودی ها :

تصویر gray

تعداد لایه های image pyramid

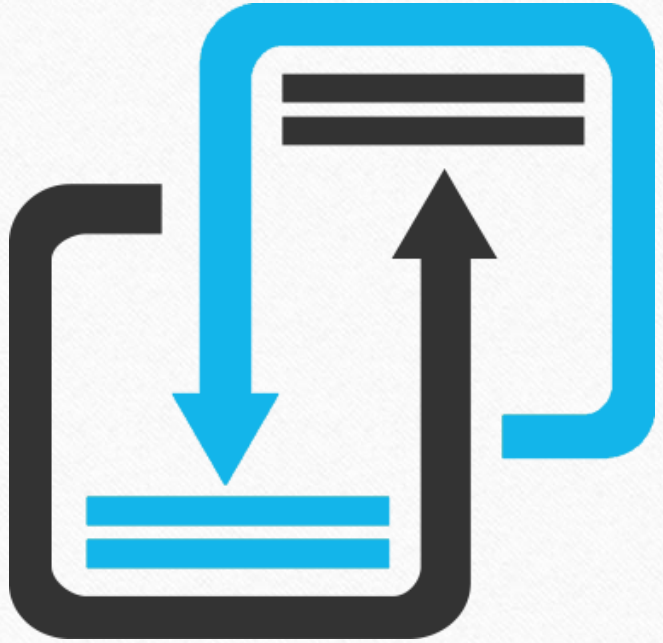
خروجی ها :

مختصات صورت های پیدا شده

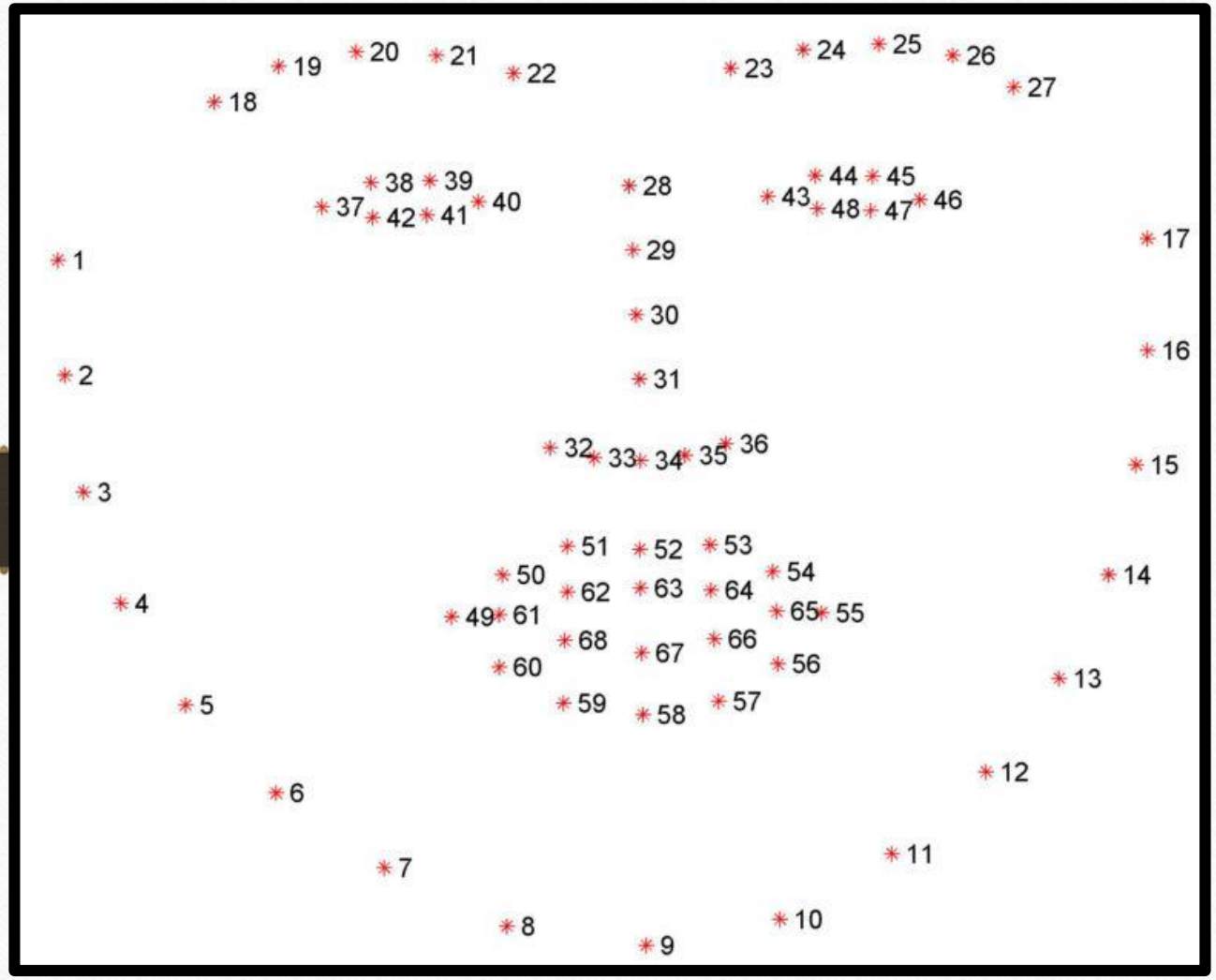
```
rects = detector(gray, 1)
```

گام 5 : تبدیل نقاط و رسم مستطیل

```
(x1, y1) = (item.left(), item.top())
(x2, y2) = (item.right(), item.bottom())

cv2.rectangle(img, (x1, y1), (x2, y2), (0,
255, 0), 3)
```



گام 6 : تشخیص اجزای صورت



Dlib صورت را به صورت 68 نقطه می بیند.

ورژن 5 نقطه ای نیز وجود دارد که فایل و عملیات آن سبک تر است.

```
shape = predictor(gray, rect)
```

گام 7 : تبدیل Shape به آرایه numpy و رسم نقاط صورت

```
shape = face_utils.shape_to_np(shape)

for (x, y) in shape:
        cv2.circle(img, (x,y), 1, (0, 0, 255), -1)
```

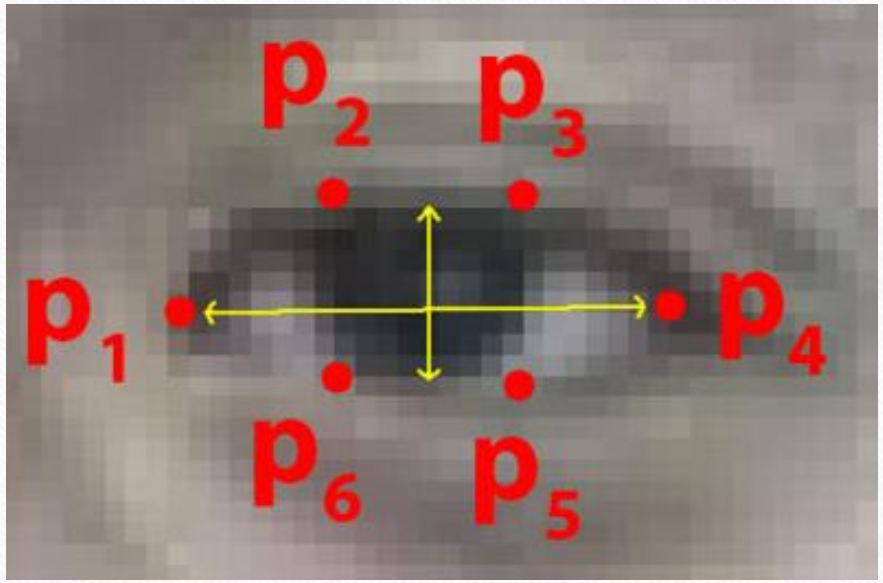
تمرین : از وبکم تصویر صورت خود را بگیرید و نتیجه را تست کنید !

تشخیص پلک زدن با Dlib

روش قدیم :

- ✓ جداسازی ناحیه مربوط به چشم
- ✓ استفاده از Thresholding و پیدا کردن مردمک چشم
- ✓ تعداد دفعات ناپدید شدن = تعداد چشمک

تشخیص پلک زدن با Dlib



چشم در Dlib به صورت 6 نقطه توصیف می شود.

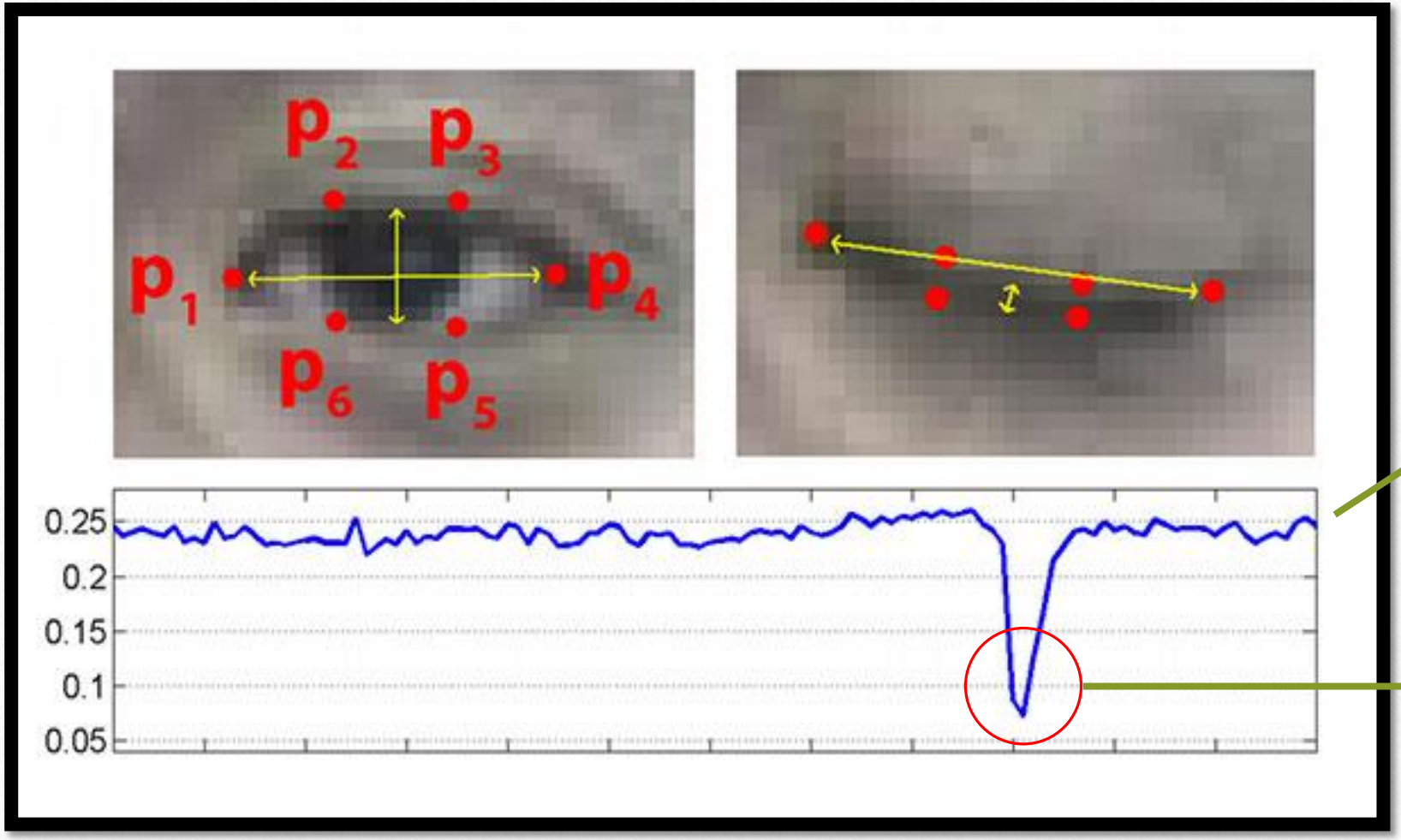
به کمک پارامتر EAR می توان پلک زدن را تشخیص داد:

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

این پارامتر در مقاله زیر معرفی شد:

Eye-Blink Detection Using Facial Landmarks 2016

میزان Threshold باید چقدر باشد؟



در حالت پیش فرض مقدار EAR تقریبا 0.3 می باشد.

جایی که پلک زدن اتفاق می افتد

گام 1 : اضافه کردن ماژول های مورد نیاز

```
from scipy.spatial import distance as dist
from imutils import face_utils
import numpy as np
import imutils
import dlib
import cv2
```

از Scipy به منظور محاسبه فاصله اقلیدسی استفاده خواهیم کرد.

گام 2 : تعریف چند ثابت مورد نیاز

```
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 3

COUNTER = 0
TOTAL = 0
```

1- مقدار آستانه پارامتر EAR

2- حداقل تعداد فریم هایی که باید مقدار EAR از Threshold کمتر باشد تا پلک زدن تشخیص داده شود.

3- شمارنده مربوط به فریم ها

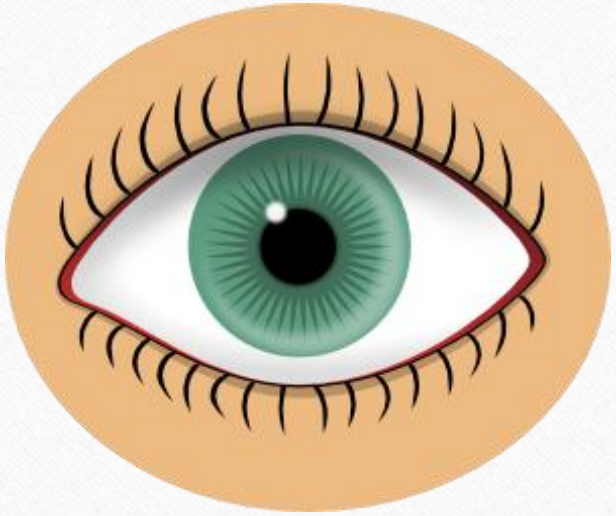
4- مجموع تعداد پلک زدن ها

گام 3 : پیدا کردن صورت و اجزای آن

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(»Address«)

frame = imutils.resize(frame, width=450)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

rects = detector(gray, 0)
for rect in rects:
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
```

گام 4 : جداسازی نقاط مربوط به چشم

```
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)

ear = (leftEAR + rightEAR) / 2.0
```



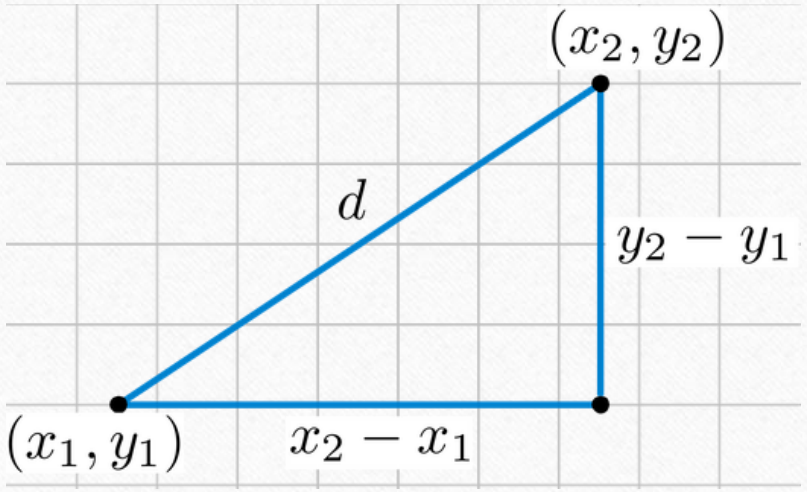
EAR نهایی را به صورت میانگین EAR چپ و راست در نظر میگیریم.

گام 5 : تعریف تابع مربوط به EAR

```
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])

    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)

    return ear
```

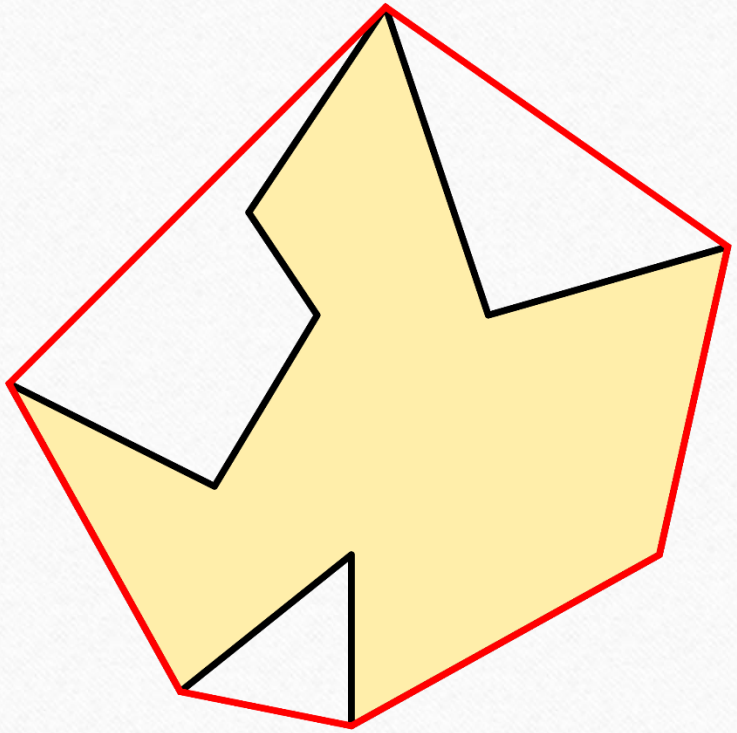
طبق رابطه ی زیر EAR را محاسبه می کنیم:

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



گام 6 : نمایش قسمت مربوط به چشم برای مشاهده بهتر

```
leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1,
(0, 255, 0), 1)
cv2.drawContours(frame, [rightEyeHull], -1,
(0, 255, 0), 1)
```



گام 7 : محاسبه تعداد پلک زدن ها

```
if ear < EYE_AR_THRESH:
        COUNTER += 1

else:
    if COUNTER >= EYE_AR_CONSEC_FRAMES:
        TOTAL += 1

    COUNTER = 0
```

گام 8 : نمایش در خروجی

```
cv2.putText(frame, "Blinks: {}".format(TOTAL), (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
        break

cv2.destroyAllWindows()
```