**Liveness Detection:**

**Prerequisite:**

1. Install below libraries.

!pip install opencv-python  
!pip install cmake dlib  
!pip install --upgrade imutils

!pip install flask

If any issues installing dlib please run as below it will install.

pip install <https://pypi.python.org/packages/da/06/bd3e241c4eb0a662914b3b4875fc52dd176a9db0d4a2c915ac2ad8800e9e/dlib-19.7.0-cp36-cp36m-win_amd64.whl#md5=b7330a5b2d46420343fbed5df69e6a3f>

note: Any library related error run the requirement.txt file as below.



1. Check the below files are there in liveness\_detection/face\_landmark\_dat

* haarcascade\_frontalface\_default.xml (used for face detection)
* shape\_predictor\_68\_face\_landmarks.dat (used for face landmarks)

note: both files will be in zip folder which I shared separately

1. Webcam

**Steps to follow for eye blink detection:**

1. Please check the api “/eyeblink” is responsibe micro service for eye blink detection
2. EAR\_THRESHOLD & EAR\_CONSEC\_FRAMES variables are responsible for the eye blinks

**EAR – Eye Aspect Ratio**

EAR\_THRESHOLD = 0.3 --> eye aspect ratio to indicate blink

EAR\_CONSEC\_FRAMES = 3 --> number of consecutive frames the eye must be below the threshold

Note: you can change the EAR threshold value to adapt for it.

def eye\_aspect\_ratio(eye):  
 # compute the euclidean distances between the two sets of  
 # vertical eye landmarks (x, y)-coordinates  
 A = dist.euclidean(eye[1], eye[5])  
 B = dist.euclidean(eye[2], eye[4])  
 # compute the euclidean distance between the horizontal  
 # eye landmark (x, y)-coordinates  
 C = dist.euclidean(eye[0], eye[3])  
 # compute the eye aspect ratio  
 ear = (A + B) / (2.0 \* C)  
 # return the eye aspect ratio  
 return ear

To know more about EAR -

<http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf>

1. Loaded the pre-trained model shape\_predictor\_68\_face\_landmarks.dat using dlib which will predict face landmarks in frame.

dlib\_detector = dlib.get\_frontal\_face\_detector()  
dlib\_predictor = dlib.shape\_predictor('liveness\_detection/face\_landmark\_dat/shape\_predictor\_68\_face\_landmarks.dat')

1. Eye landmark details
   * Left eye Start=42
   * Left eye End=48
   * Right eye Start=36
   * Right eye End=42
2. **Frame\_max\_counter** is variable responsible for latency. This variable is used for latency removal. In our local experiment, we received good results in 8 – 10 frames.

Note: I had configured it in function itself, you can pass as args.

1. Vc = Cv2.Videocapture() – will enable the connection to connect with camera and vc.read() will read the frames.
2. We need to detect the face and predict the landmarks using predictor function then we need calculate the EAR for predicted landmarks.

# determine the facial landmarks for the face region, then  
# convert the facial landmark (x, y)-coordinates to a NumPy  
# array  
shape = predictor(gray, rect)  
shape = np.array([[p.x, p.y] for p in shape.parts()])  
# extract the left and right eye coordinates, then use the  
# coordinates to compute the eye aspect ratio for both eyes  
leftEye = shape[lStart:lEnd]  
rightEye = shape[rStart:rEnd]  
leftEAR = eye\_aspect\_ratio(leftEye)  
rightEAR = eye\_aspect\_ratio(rightEye)  
# average the eye aspect ratio together for both eyes  
ear = (leftEAR + rightEAR) / 2.0

1. Computing to the convex hull for both eyes and draw the contours respectively for highlighting the eye.

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)

1. We will count the blinks by comparing previous frame eye closeness/eye open state with current frame eye closeness/eye open state which is calculated based on EAR.
2. Function will return True or False based on the blinks count.
3. Clean up the video capture window

Note: I have added the necessary comments in entire script.

**Steps to follow for head pose tilt detection:**

1. Please check the api “/righttilt” is responsibe micro service for eye blink detection
2. Bounding box coordinates for right tilt follows as below

Fixed\_x1, Fixed\_y1, Fixed\_x2, Fixed\_y2 = 100, 150, 250, 300

1. Vc = Cv2.Videocapture() – will enable the connection to connect with camera and vc.read() will read the frames.
2. **Frame\_max\_counter** is variable responsible for latency. This variable is used for latency removal. In our local experiment, we received good results in 8 – 10 frames.

Note: I had configured it in function itself, you can pass as args.

1. Load the haarscade frontal face classifier for face detection

haar\_detector = cv2.CascadeClassifier('liveness\_detection/face\_landmark\_dat/haarcascade\_frontalface\_default.xml')

1. Once user try to move the face towards bounding box will calculate the movement between face detected bounding box vs fixed bounding box.

img = cv2.rectangle(frame, (fixed\_x1, fixed\_y1), (fixed\_x2, fixed\_y2), (255, 255, 255), 2)  
  
# calculating the pixel difference b/w face detection bounding box & fixed bounding box  
var\_pixel = abs(fixed\_x1 - x1) + abs(fixed\_y1 - y1) + abs(fixed\_x2 - x2) + abs(fixed\_y2 - y2)  
if var\_pixel <= 80:  
 result = True

1. Based on movement calculation will decided the result.
2. Cleanup the video capture window.

Note: Left tilt, Up Tilt and Down tilt same as right tilt only fixed bounding box coordinates will change.

---------------------------------------------------END--------------------------------------------------------