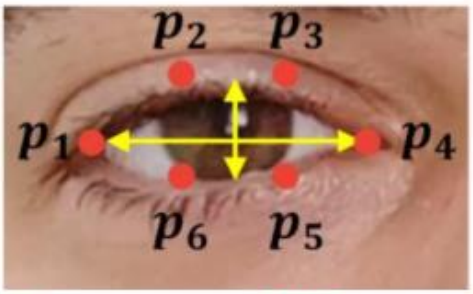
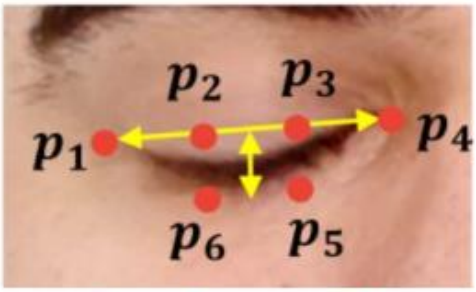
TASK 2 – Drowsiness Detection

Eye Aspect Ratio:

The Eye Aspect Ratio is an estimate of the eye opening state.



Open eye will have more EAR



Closed eye will have less EAR

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

Where ||p2-p6|| represents the distance between points p2 and p6.

The more the EAR, the more widely eye is open. We would decide a minimum EAR value and used this to decide if the eye is closed or not.

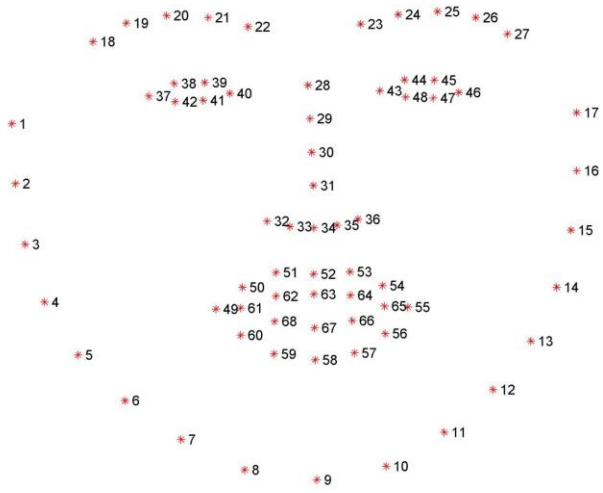
CODE to calculate EAR:

```
def eye_aspect_ratio(eye):

    p2_minus_p6 = dist.euclidean(eye[1], eye[5])
    p3_minus_p5 = dist.euclidean(eye[2], eye[4])
    p1_minus_p4 = dist.euclidean(eye[0], eye[3])
    ear = (p2_minus_p6 + p3_minus_p5) / (2.0 * p1_minus_p4)
    return ear
```

Facial Landmark Detection

We will use dlib library to detect facial landmarks. The dlib library can be used to detect a face in an image and then find 68 facial landmarks on the detected face.



Visualizing the 68 facial landmark coordinates

So according to the image, left eye will be from 37-42 and right eye will be from 43-48

The return value of the eye aspect ratio will be approximately constant when the eye is open. The value will then rapid decrease towards zero during a blink.

If the eye is closed, the eye aspect ratio will again remain approximately constant, but will be much smaller than the ratio when the eye is open.

Techniques to detect drowsiness:

We will set a minimum threshold for the eye aspect ratio (EYE_AR_THRESH) below which the eye will be consider closed. We will also set the number of frames for which if the eye is closed (EYE_AR_CONSEC_FRAMES), drowsiness will be detected. The minimum threshold for the EAR will be around 0.2-0.3. For the number frames, we should choose an optimal value so that it doesn't detect eye blink as drowsiness but also not too long, which will detect drowsiness late and thus defeat the purpose of the application.

CODE:

```
# check to see if the eye aspect ratio is below the blink

# threshold, and if so, increment the blink frame counter

if ear < EYE_AR_THRESH:

        COUNTER += 1

        # if the eyes were closed for a sufficient number of

        # then sound the alarm

        if COUNTER >= EYE_AR_CONSEC_FRAMES:

# if the alarm is not on, turn it on
```

```
        if not ALARM_ON:
                ALARM_ON = True
                # check to see if an alarm file was supplied,
                # and if so, start a thread to have the alarm
                # sound played in the background
                if args["alarm"] != "":
                        t = Thread(target=sound_alarm,
                        args=(args["alarm"],))
                        t.deamon = True
                        t.start()
                        # draw an alarm on the frame
        cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
# otherwise, the eye aspect ratio is not below the blink
# threshold, so reset the counter and alarm
else:
        COUNTER = 0
        ALARM_ON = False
```

We make a check to see if the eye aspect ratio is below the “blink/closed” eye threshold, EYE_AR_THRESH .

If it is, we increment COUNTER , the total number of consecutive frames where the person has had their eyes closed.

If COUNTER exceeds EYE_AR_CONSEC_FRAMES, then we assume the person is starting to doze off.

Another check is made, to see if the alarm is on — if it’s not, we turn it on.

We handle playing the alarm sound, provided an --alarm path was supplied when the script was executed. We take special care to create a separate thread responsible for calling sound_alarm to ensure that our main program isn’t blocked until the sound finishes playing.

Finally, we handle the case where the eye aspect ratio is larger than EYE_AR_THRESH , indicating the eyes are open. If the eyes are open, we reset COUNTER and ensure the alarm is off.

Conclusion:

We used dlib library for determining facial landmarks. The dlib library is pretty simple to use and works smoothly even without GPU. But there might also be other free-to-use libraries available on the market. Mediapipe is one such example. According to the National Highway Traffic Safety Administration, every year about 100,000 police-reported crashes involve drowsy driving. These crashes result in more than 1,550 fatalities and 71,000 injuries. The real number may be much higher, however, as it is difficult to determine whether a driver was drowsy at the time of a crash. Hence the drowsiness detection system is a very useful application. When used with raspberry pi 3 or a web cam it can detect drowsiness in real time and alert the driver, hence preventing accidents.

Reference Links:

https://medium.com/analytics-vidhya/eye-aspect-ratio-ear-and-drowsiness-detector-using-dlib-a0b2c292d706

https://towardsdatascience.com/drowsiness-detection-system-in-real-time-using-opencv-and-flask-in-python-b57f4f1fcb9e

https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/