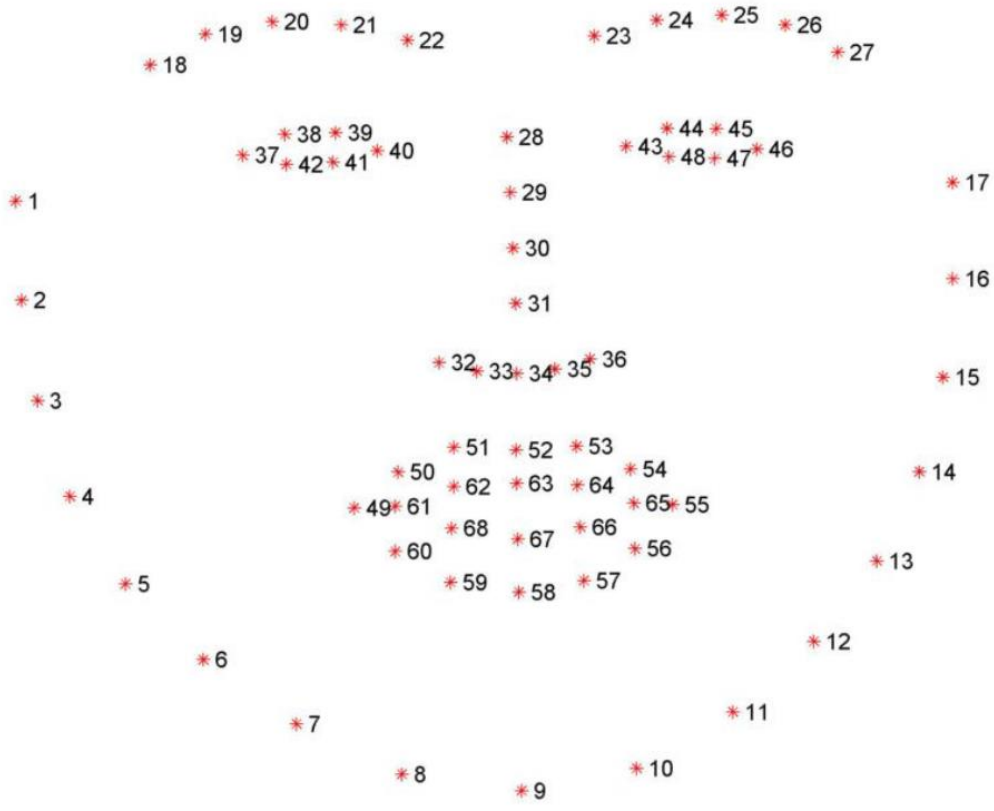
COMPUTER VISION

Pre-processing:

The image taken from the dashboard of the cars have to be converted into Gray Scale image from BGR(Blue-Green-Red) image. This can be done using cvtColor() function from OpenCV library (cv2) which uses COLOR_BGR2GRAY function from same library.

Detection of Eye:

1. The first technique and the one that I have used is a C++ library dlib for facial detection and facial landmark detection. Dlib HoG is the fastest method on CPU for facial detection which are not small (< 70x70) and as the faces we get on dashboard are not that small so I used this.
   First I used the get_frontal_face_detector() function from dlib which detects the front view of the grey scaled face.
   Then with the help of shape_predictor and 68 face landmarks, we added the face landmarks on the front face using grey image.

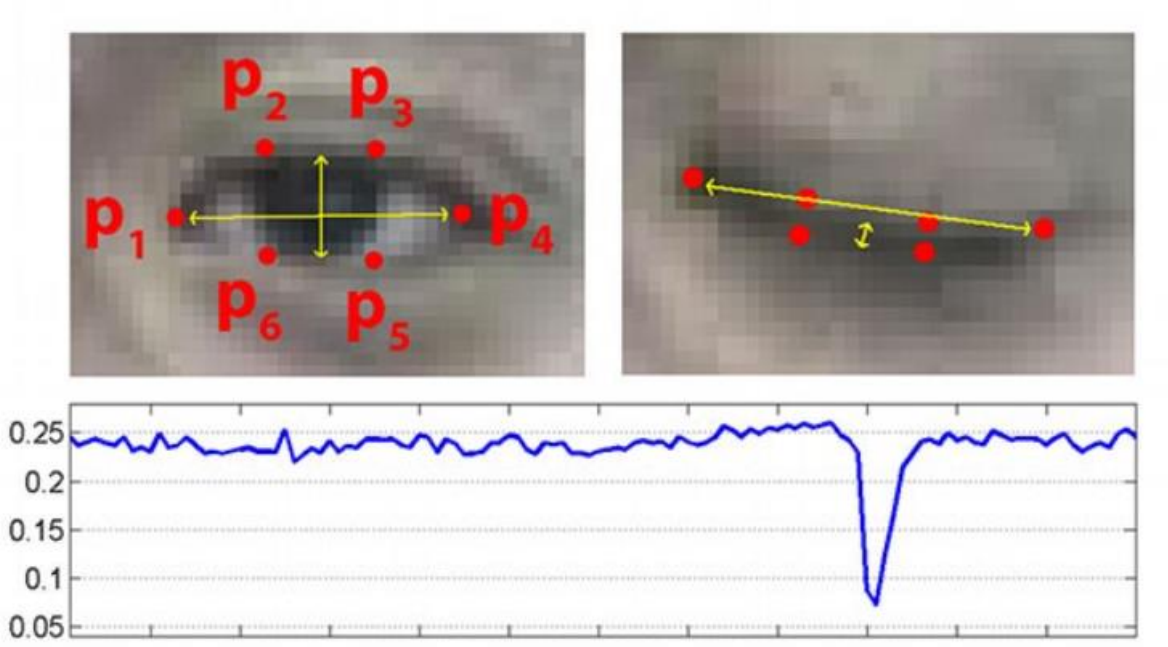


The above image shows all the face landmarks. The landmarks of the eyes are from 37-42 for left eye and 43-48 for the left eye. Now using these points co-ordinates we can draw an outline around the eye using the line() function of the OpenCV library (cv2).

2. The second technique I thought of was the Haar Cascade method to detect the face and eyes but then I found it difficult to check whether the eyes are closed or open as compared to the First technique so I didn't used this.

Drowsiness Detection:

So, after detecting the eyes, the drowsiness can be detected be checking whether the eyes are closing slowly. To check whether the eyes are open or closed I used the co-ordinates of the eye landmarks on the face and checked the vertical and horizontal length of the eyes. For this I used the distance function from scipy library. In this I used the Euclidean() function to determine the distance between 2 co-ordinates and found the Eye Aspect Ratio (EAR).



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

Eye Aspect Ratio(EAR) Equation.

Now after getting the EAR, we have to set a limit after which it will detect drowsiness of the person. I have take 0.20 as the limit below which it will detect drowsiness and alert the driver.

Conclusion:

I have selected this algorithm cause its fast and easy to understand. Being a C++ library, it is faster than other library and we get fast and accurate response and as the faces are not too small captured by the dashboard so its limitations is also cancelled out.