DriveAlert: A solution for real-time drowsiness and yawning detection for drivers to enhance safety on highways and in transportation industries.

Yash Kumar
2021UCS1664

Faiz Ahmad
2021UCS1665

Ahmed Mohammad Yunus
2021UCS1685

PROBLEM STATEMENT

Truck drivers, responsible for transporting cargo over extended distances across varying times of day, frequently contend with sleep deprivation. Fatigue and drowsiness arising from this condition pose significant dangers, contributing to a high incidence of accidents on highways.**[4]** Recognizing the critical need to address this issue, the automotive industry has been actively researching technologies capable of detecting drowsiness and promptly alerting drivers. However, there remains a gap in implementing accessible and effective solutions tailored specifically for truck drivers. Thus, this project aims to develop a robust system that accurately detects signs of drowsiness in drivers and provides timely alerts, thereby enhancing safety and reducing the risk of accidents on highways.

1. INTRODUCTION

In transportation sector, long-distance truck drivers often grapple with the challenge of staying alert during extended journeys, leading to the perilous risk of accidents caused by drowsiness.**[1]** To mitigate this risk, DriveAlert emerges as a vital innovation, aiming to revolutionize driver safety through cutting-edge technology. DriveAlert is a Sleep Sensing and Alerting System for Drivers, meticulously crafted using Raspberry Pi, OpenCV, and a Pi camera module. This system represents a pivotal advancement in the automotive sector, leveraging facial recognition and eye-tracking capabilities to monitor the driver's condition in real-time. By building upon foundational work in facial landmark detection, DriveAlert extends its functionality to include drowsiness detection, ensuring timely intervention to prevent accidents on highways.
Some Use Cases:

- Public Transportation
- Railways / Airlines Industries
- Personal Vehicles

2. COMPONENTS REQUIRED

Hardware Components

- Raspberry Pi 3
- Pi Camera Module
- Micro USB Cable
- Buzzer

Software and Online Services

- OpenCV
- Dlib
- Python3

3. METHODLOGY

The system captures video frames from a webcam and preprocesses them by converting them to grayscale and resizing for faster processing. In this project, dlib library used to detect a face in an image and then find 68 facial landmarks on the face, , as illustrated in fig. 1.

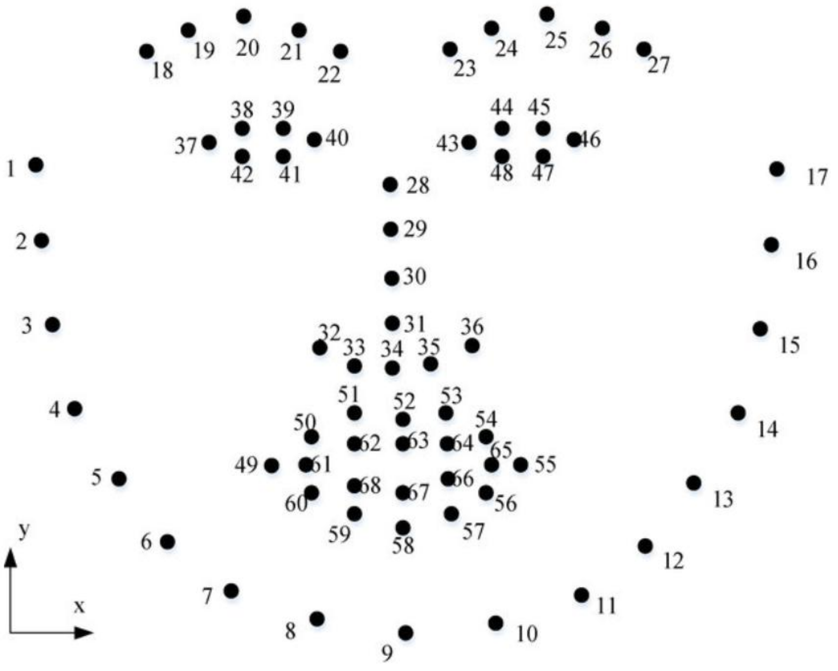


Fig1. 68 facial landmarks that dlib finds in a face

In fig. 1, 37 - 42 represents left eye, 43 - 48 represents right eye and 49 - 68 represents mouth (Upper Lip & Lower Lip).

3.1 Eye Aspect Ratio (EAR)

It measures the ratio of distances between landmarks around the eyes to detect eye openness. A lower "EAR" indicates drowsiness, as illustrated in Fig. 2. The EAR for a single eye is calculated using this formula**[2]**:

$$EAR = (||p2 - p6|| + ||p3 - p5||) / (2 * ||p1 - p4||)$$

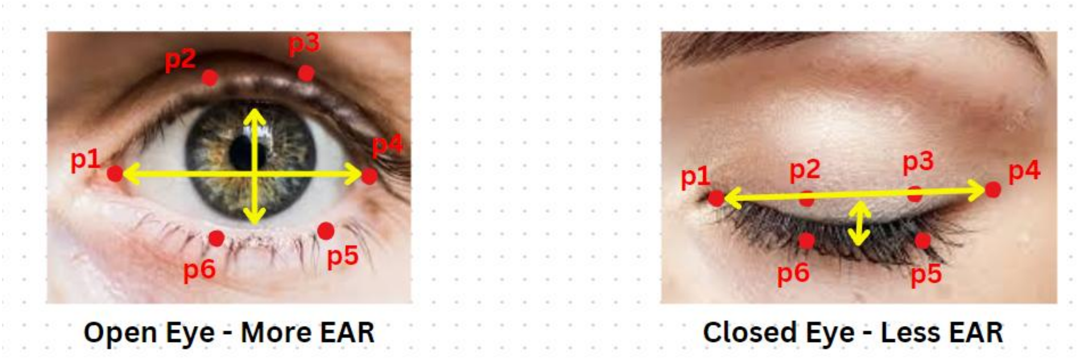


Fig2. facial landmarks for Eye

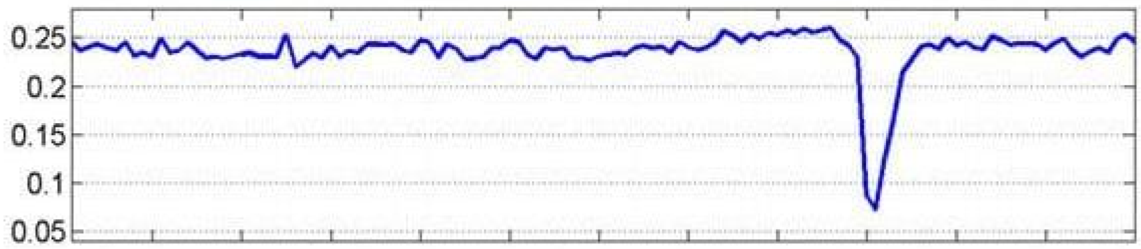


Fig3. EAR change over Time (EAR vs Time)

3.2 Yawn Detection

By calculating the distance between upper and lower lip landmarks, yawning can be detected. The mouth is characterized by 20 coordinates (from 49 to 68), as shown in Fig. 1. However, we used points from 61 to 68, as displayed to obtain the mouth openness degree. It can be calculated using this formula **[1]**:

MAR(Mouth Aspect Ratio) = (|p2 - p8| + |p3 - p7| + |p4 - p6|) / (2 * |p1 - p5|)

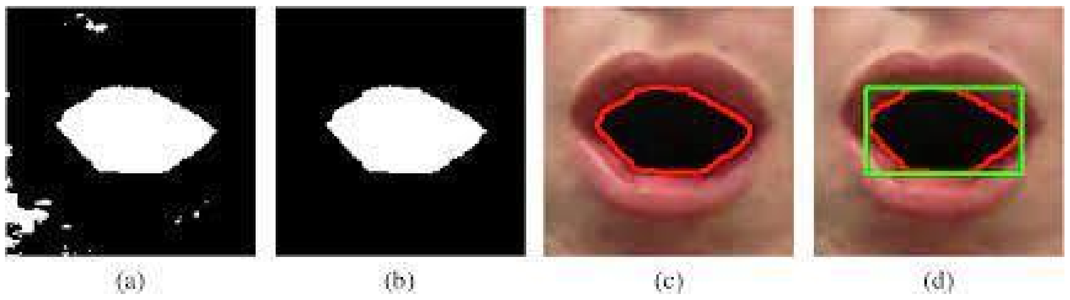


Fig4. Yawn Detection and MAR Estimation

4. SYSTEM ARCHITECTURE

As shown in Fig. 5, This system detects drowsiness! It grabs real-time video from your webcam, finds your face, and tracks your eyes and mouth. By analyzing if your eyes are closed and your mouth is wide open, it detects drowsiness or yawns. It then combines these clues to decide if you're nodding off and can trigger an alert to keep you safe.

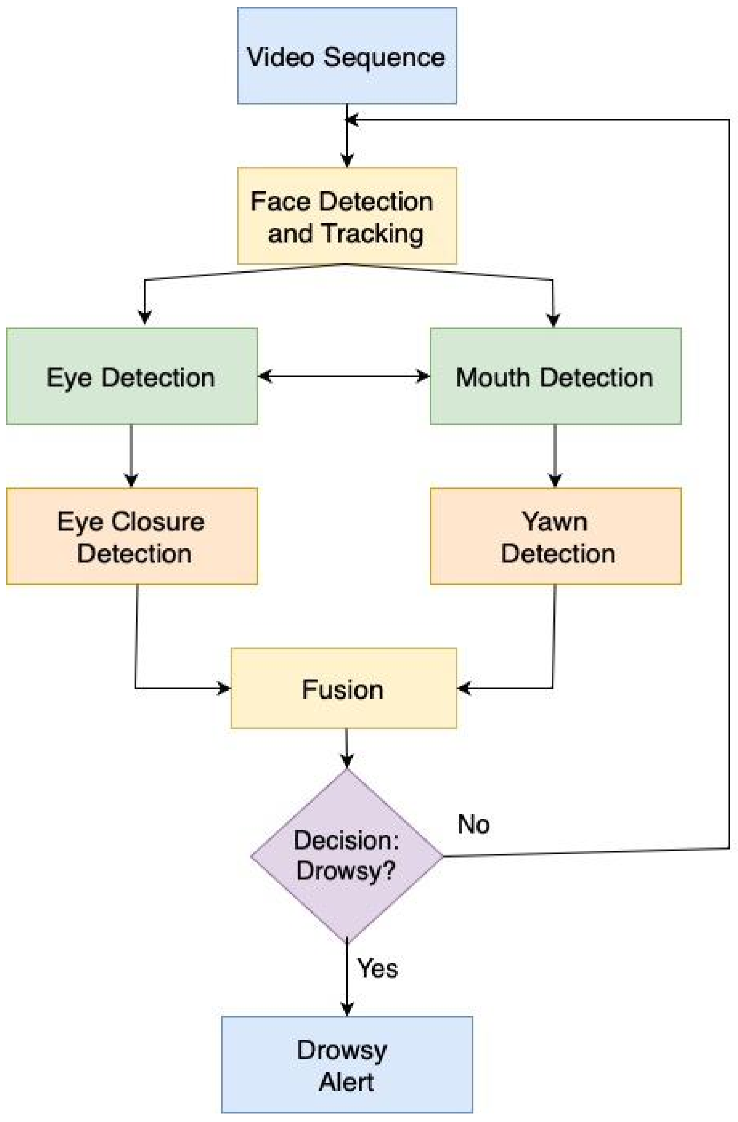
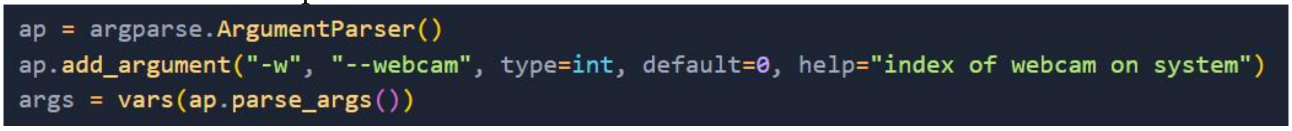


Fig. 5 System Architecture

5. CODE

5.1 Data Preparation

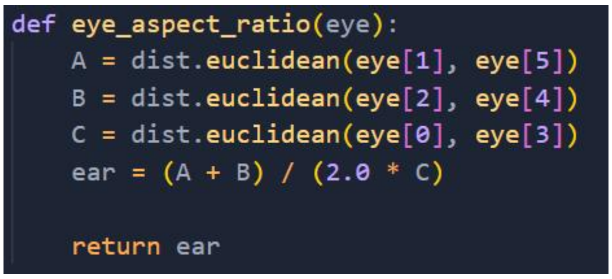
Initializes webcam to capture video frames:



```
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0, help="index of webcam on system")
args = vars(ap.parse_args())
```

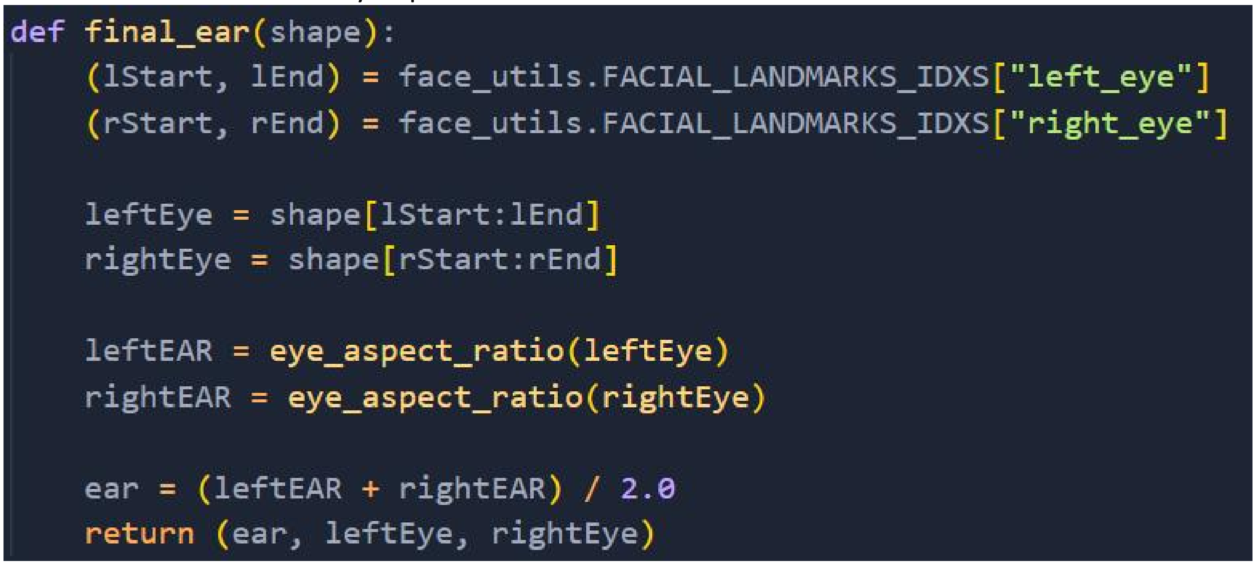
5.2 Model

Eye Aspect Ratio (EAR):



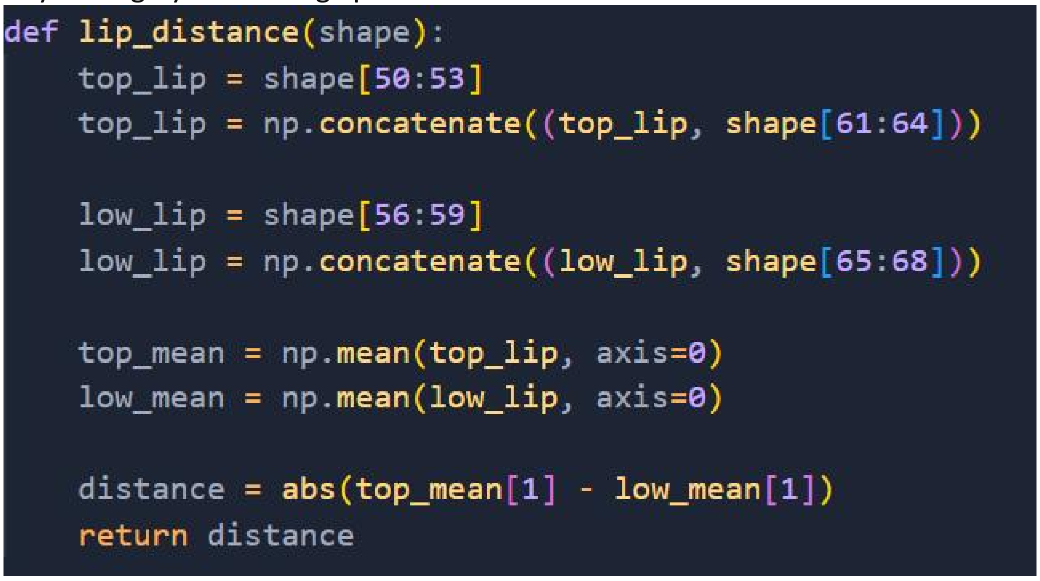
```
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)

    return ear
```

Calculates EAR to measure eye openness :



```
def final_ear(shape):
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]

    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)

    ear = (leftEAR + rightEAR) / 2.0
    return (ear, leftEye, rightEye)
```

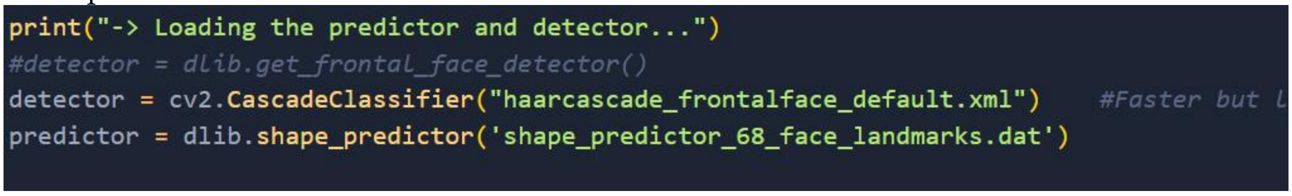
Determines yawning by calculating lip distance :



```
def lip_distance(shape):
    top_lip = shape[50:53]
    top_lip = np.concatenate((top_lip, shape[61:64]))

    low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))

    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)

    distance = abs(top_mean[1] - low_mean[1])
    return distance
```

5.3 Training the Model

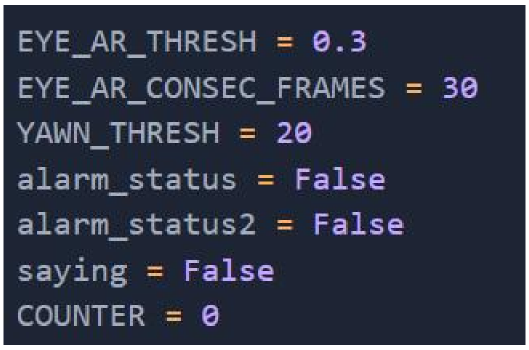
Utilizes pre-trained models for face and facial landmark detection:



```
print("-> Loading the predictor and detector...")
#detector = dlib.get_frontal_face_detector()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")    #Faster but L
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

5.4 Parameter Tuning:

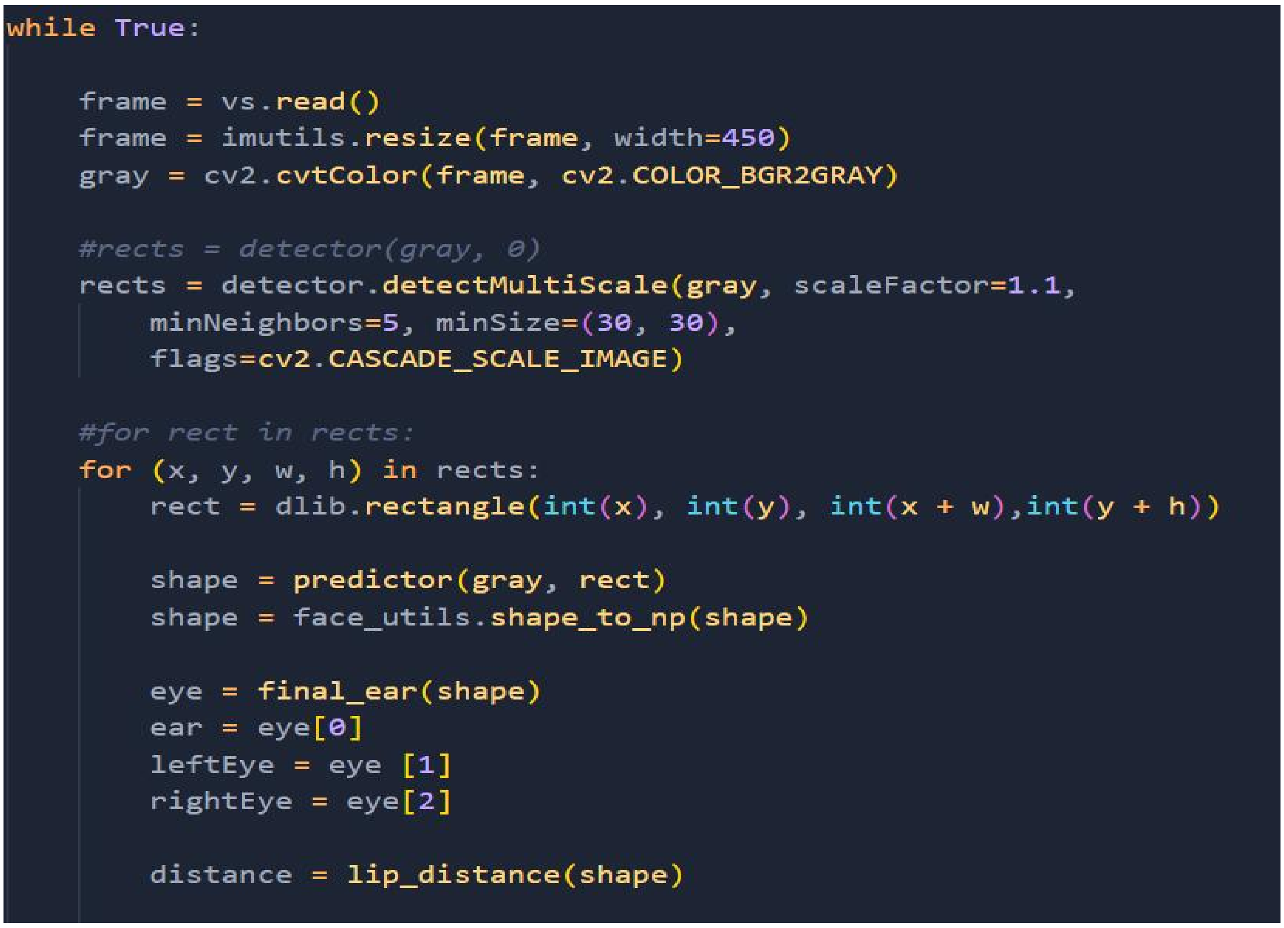
Defines thresholds (**EYE_AR_THRESH**, **YAWN_THRESH**) for drowsiness and yawning detection:



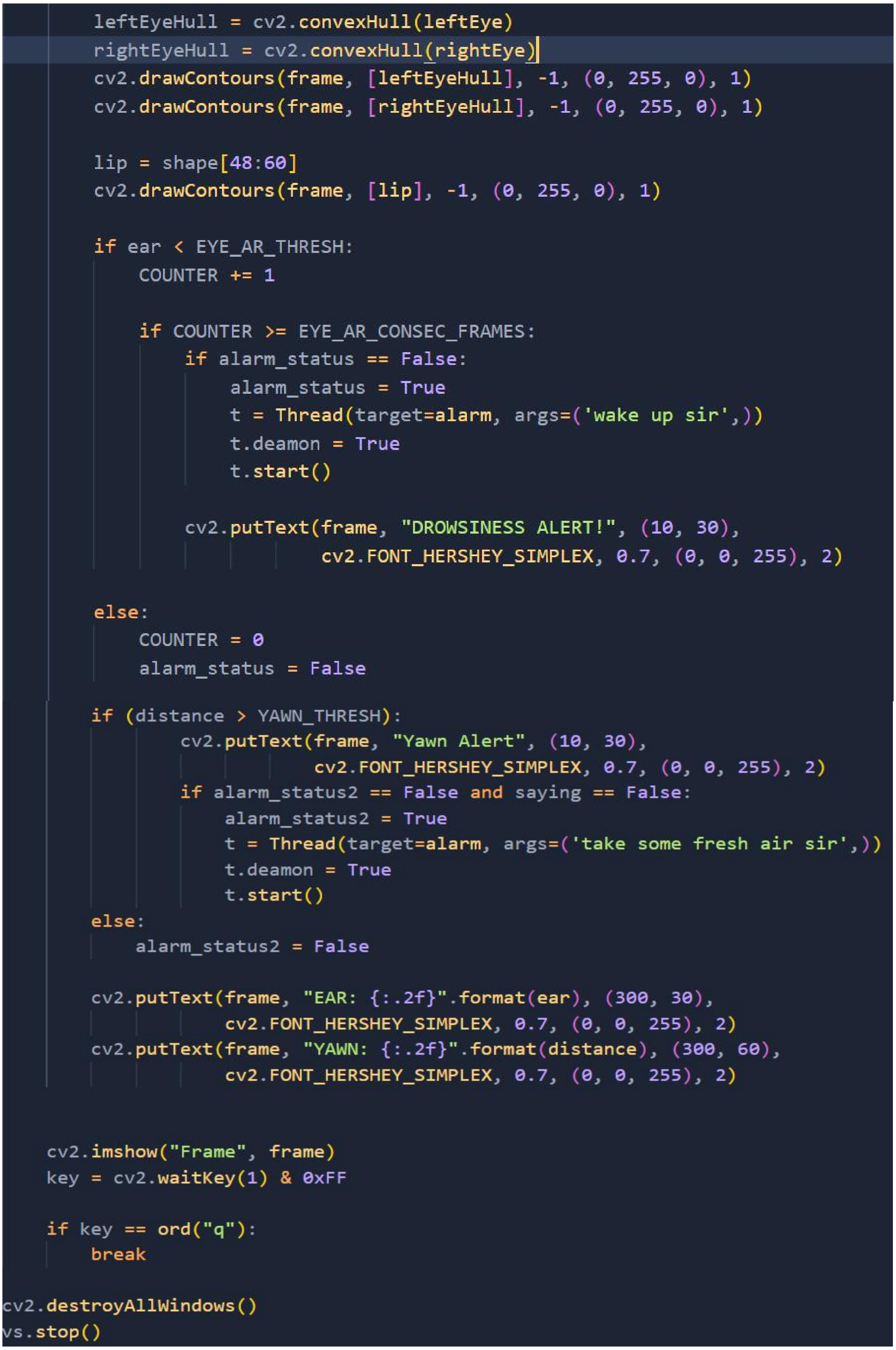
```
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 30
YAWN_THRESH = 20
alarm_status = False
alarm_status2 = False
saying = False
COUNTER = 0
```

5.5 Prediction:

- Continuously reads frames from webcam.
- Detects faces using Haar cascade classifier.
- Extracts facial landmarks with dlib.
- Determines drowsiness and yawning based on calculated values and thresholds.
- Displays processed frame with annotations in real-time.
- Continues until user presses 'q' to exit.



```
while True:

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

    #rects = detector(gray, 0)
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE)

    #for rect in rects:
    for (x, y, w, h) in rects:
        rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))

        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

        eye = final_ear(shape)
        ear = eye[0]
        leftEye = eye [1]
        rightEye = eye[2]

        distance = lip_distance(shape)
```



```
        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)

        lip = shape[48:60]
        cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

        if ear < EYE_AR_THRESH:
            COUNTER += 1

            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                if alarm_status == False:
                    alarm_status = True
                    t = Thread(target=alarm, args=('wake up sir',))
                    t.deamon = True
                    t.start()

                cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        else:
            COUNTER = 0
            alarm_status = False

        if (distance > YAWN_THRESH):
                cv2.putText(frame, "Yawn Alert", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                if alarm_status2 == False and saying == False:
                    alarm_status2 = True
                    t = Thread(target=alarm, args=('take some fresh air sir',))
                    t.deamon = True
                    t.start()
        else:
            alarm_status2 = False

        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                    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()
vs.stop()
```

6. RESULT

The DriveAlert system exhibited accurate detection of drowsiness and yawning in drivers through real-time analysis of webcam feed, as shown in fig. 6. Leveraging the Eye Aspect Ratio (EAR) algorithm and lip distance calculations, it provided timely alerts based on predefined thresholds. The system offered reliable performance, continuously monitoring drivers and delivering immediate feedback through annotated video frames**[4]**. Moreover, its lightweight design allows deployment on Raspberry Pi, ensuring portability and accessibility for various applications. DriveAlert's efficacy in enhancing driver safety underscores its potential to mitigate accidents caused by fatigue, with scalability for deployment in diverse settings, including personal vehicles and commercial fleets.

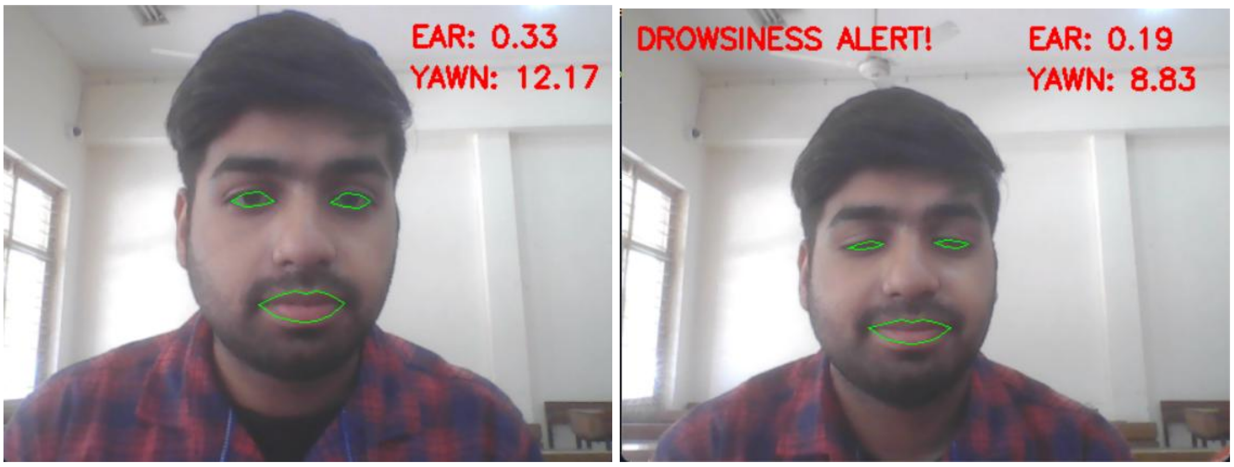


Fig 6. Result Produced from model

7. CONCLUSION

The DriveAlert project presents a robust solution for real-time drowsiness and yawning detection using computer vision techniques. By harnessing facial landmarks and image processing algorithms, the system effectively identifies signs of fatigue, offering significant potential for enhancing safety in critical contexts like driving and operating machinery**[1]**. Through accurate measurement of eye openness and lip distance, DriveAlert provides timely alerts to drivers, mitigating the risks associated with drowsy driving and preventing accidents on highways.

8. FUTURE WORK

In future iterations, the integration of MediaPipe technology holds promise for enhancing DriveAlert's capabilities. MediaPipe is a cutting-edge framework developed by Google that offers advanced solutions for real-time perception tasks, including facial landmark detection, pose estimation, and gesture recognition**[1]**. By harnessing MediaPipe's sophisticated algorithms and deep learning models, DriveAlert could benefit from improved accuracy and robustness in detecting drowsiness and yawning. MediaPipe's ability to accurately track facial landmarks and analyze facial expressions in real-time could provide more precise measurements of eye openness and lip distance as shown in Fig 6, essential for identifying signs of fatigue. Integrating MediaPipe technology represents a significant opportunity to advance DriveAlert's capabilities and reinforce its role in enhancing driver safety on highways and other transportation routes.

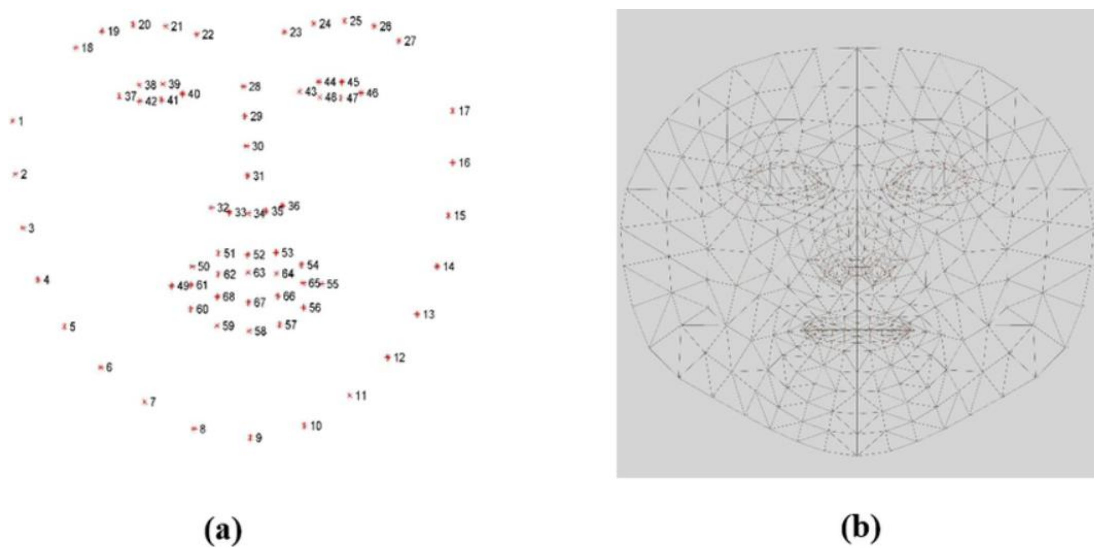


Fig 7. The map of the two landmarks solutions that were used. (a) Dlib facial landmarks solution map. (b) MediaPipe face mesh solution map.

9. References

[1] Yaman Albadawi, Maen Takruri (2023). “Real-Time Machine Learning-Based Driver Drowsiness Detection Using Visual Features”
https://www.researchgate.net/publication/370456907_Real-Time_Machine_Learning-Based_Driver_Drowsiness_Detection_Using_Visual_Features

[2] Dhruv Pandey (2021) “Eye Aspect Ratio(EAR) and Drowsiness detector using dlib”
https://medium.com/analytics-vidhya/eye-aspect-ratio-ear-and-drowsiness-detector-using-dlib-a0b2c292d706

[3] PRASAD V PATIL “Drowsiness Detection Dataset”
https://www.kaggle.com/datasets/prasadvpatil/mrl-dataset

[4] Ashish Choudhary(2020) “ Driver Drowsiness Detector System using Raspberry Pi and OpenCV”
https://circuitdigest.com/microcontroller-projects/driver-drowsiness-detector-using-raspberry-pi-and-opencv