VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Digital Assignment-III, APRIL 2021

B.Tech., Winter-2020-2021

COURSE CODE	ITE3999
COURSE NAME	Technical Answers to Real World Problems (TARP)
SLOT	TE1
FACULTY	Prof. VIJAYAN E.

TEAM 7 MEMBERS:

REG. NO.	NAME
18BIT0027	SAKINA HUSSAIN BANDOOKWALA
18BIT0231	KUSHAGRA AGARWAL
18BIT0272	PRIYAL BHARDWAJ

Implementation of the proposed approach

- First of all, we have captured video from the camera and detected face from the video frames.
- After that we have calculated Eyes Aspect Ratio(EAR) and compared that to threshold ratio. If the EAR is less than the threshold then an alert message will be displayed upon the screen.
- Also apart from the message upon the screen, we will be sending a message on the phone of the emergency contact of driver along with the location of the driver.

Eye Detection

- In the system we have used facial landmark prediction for eye detection.
- The facial landmark detector included in the dlib library is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).
- This method starts by using:
    1. A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
    2. Priors, of more specifically, the probability on distance between pairs of input pixels.
- The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.
- We can detect and access both the eye region by the following facial landmark index show below
    1. The right eye using [36, 42].
    2. The left eye with [42, 48].
- These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on.
- Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data.

Eye Aspect Ratio (EAR)

- For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.
    - $EAR = \frac{||p2 - p6|| + ||p3 - p5||}{2||p1 - p4||}$

where p1, . . ., p6 are the 2D landmark locations.

- The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye.
- Aspect ratio of the open eye has a small variance among individuals, and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face.
- Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.

Eye State Detection

- Finally, the decision for the eye state is made based on EAR calculated in the previous step. If the distance is zero or is close to zero, the eye state is classified as “closed” otherwise the eye state is identified as “open”.
- The last step of the algorithm is to determine the person’s condition based on a pre-set condition for drowsiness. The average blink duration of a person is 100-400 milliseconds (i.e. 0.1-0.4 of a second).
- Hence if a person is drowsy his eye closure must be beyond this interval. We have set an interval of 48 frames which is nearly 3 seconds. If the eyes remain closed for three or more seconds, drowsiness is detected and alert regarding this is triggered.

Code

```
# USAGE
# python detect_drowsiness.py --shape-
predictor shape_predictor_68_face_landmarks.dat
# python detect_drowsiness.py --shape-
predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav

# import the necessary packages
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import playsound
import argparse
import imutils
import time
import dlib
import cv2
import serial
from twilio.rest import Client

import geocoder
```

```
g = geocoder.ip('me')
print(str(g.latlng))

account_sid = 'ACc5166e7f1310f5d7fba4d6a96bfdb9a2'
auth_token = 'd02288d9d634561e51b2de0086b74a95'
client = Client(account_sid, auth_token)


###ser = serial.Serial('COM6', 9600, timeout=1)
#ser.open()
###ser.setDTR(False)
###ser.close()

def sound_alarm(path):
    # play an alarm sound
    playsound.playsound(path)

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

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-
predictor", required=True, help="path to facial landmark predictor")
ap.add_argument("-a", "--
alarm", type=str, default="", help="path alarm .WAV file")
ap.add_argument("-w", "--
webcam", type=int, default=0, help="index of webcam on system")
args = vars(ap.parse_args())

# define two constants, one for the eye aspect ratio to indicate
# blink and then a second constant for the number of consecutive
# frames the eye must be below the threshold for to set off the
# alarm
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48  #48*30fps
```

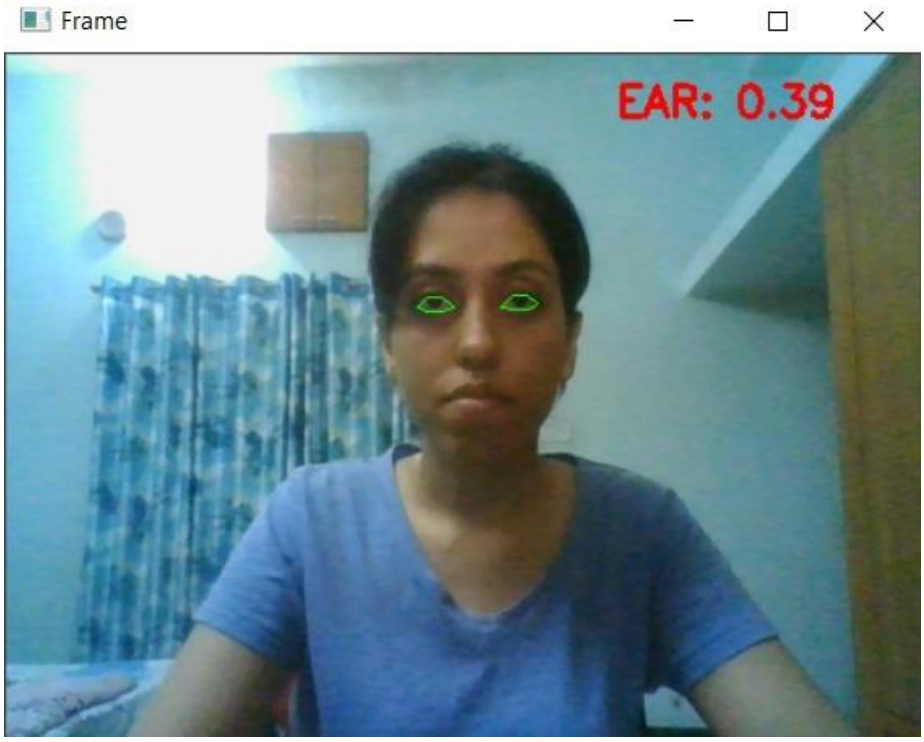
```
sms = 1

# initialize the frame counter as well as a boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

# grab the indexes of the facial landmarks for the left and
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# start the video stream thread
print("[INFO] starting video stream thread...")
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)

# loop over frames from the video stream
while True:
    #time.sleep(0.1)
    # grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces in the grayscale frame
    rects = detector(gray, 0)

    # loop over the face detections
    for rect in rects:
        # 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 = face_utils.shape_to_np(shape)

        # 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

        # compute the convex hull for the left and right eye, then
        # visualize each of the eyes
        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)

        # 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

                    ###ser.open()
                    ###ser.write(b'1')
                    ###ser.close()

                    if(sms == 1):
                        message = client.messages \
                        .create(
                            body="Drowsy driver detected" + " at location " +
str(g.latlng),

                            from_='+18306269708',
                            to='+919424157184'
                            )
                        print(message.sid)
                        sms = 0
                        print('SMS Sent')

                    # 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"],))
```

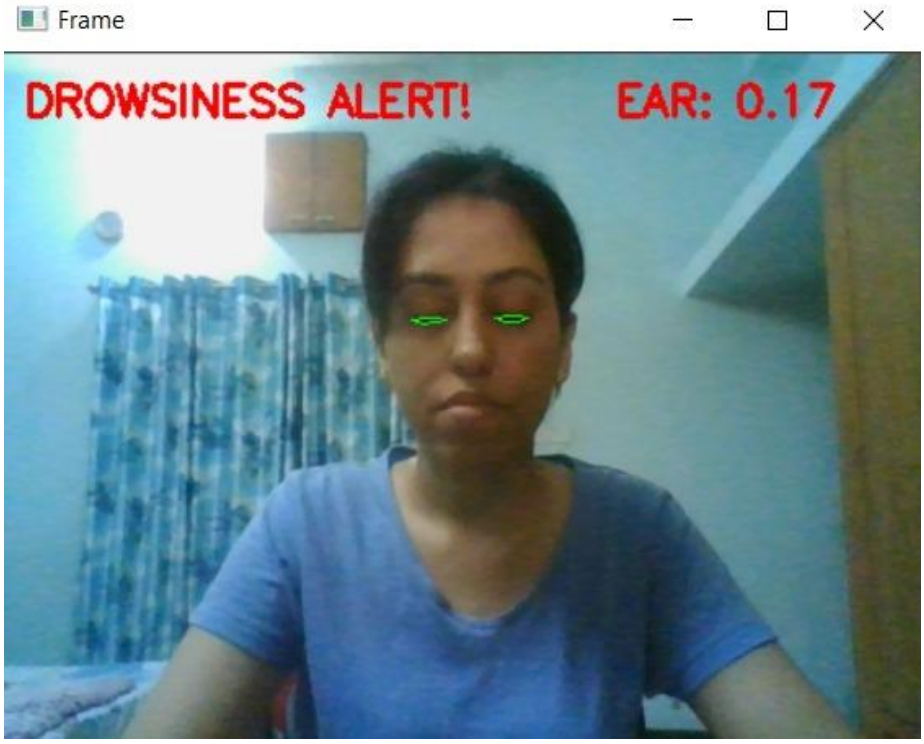
```
                        t.deamon = True
                        t.start()

                # 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
            sms = 1
            ALARM_ON = False

            ###ser.open()
            ###ser.write(b'0')
            ###ser.close()

        # draw the computed eye aspect ratio on the frame to help
        # with debugging and setting the correct eye aspect ratio
        # thresholds and frame counters
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # show the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Results

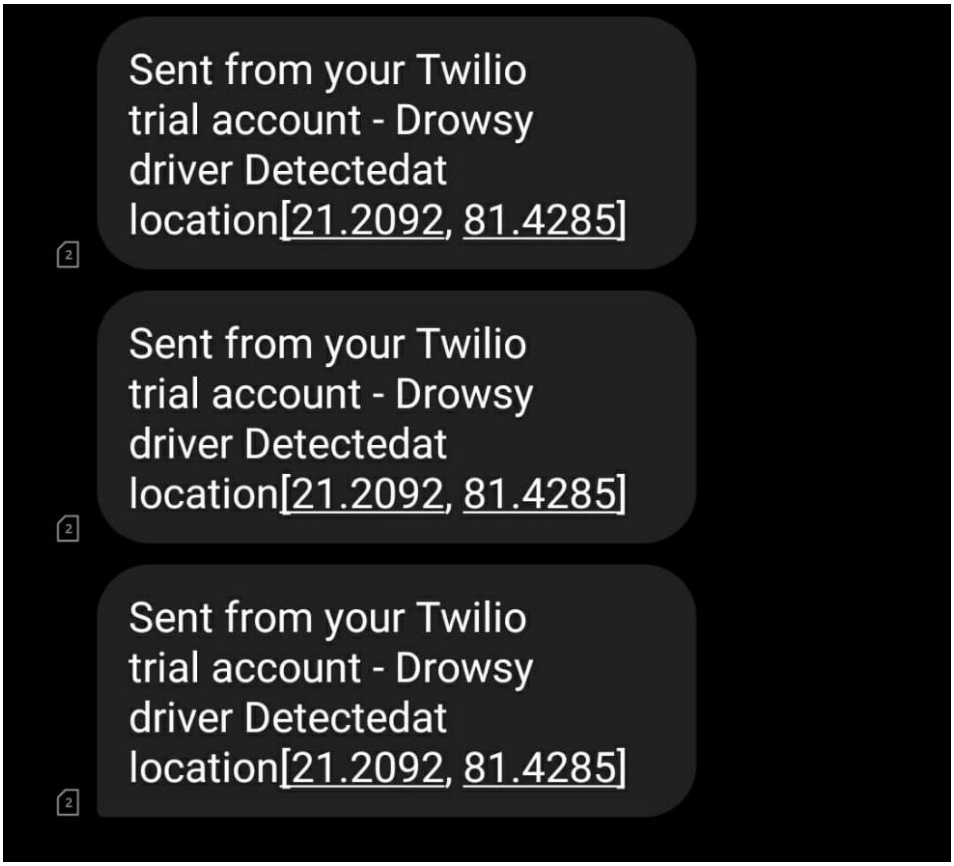
Eyes open



Eyes closed



Below is a sample of the message received on mobile of emergency contact given by the driver. Message comprises of the location of the drowsy driver and can be further customised as per the driver choice.


Sent from your Twilio trial account - Drowsy driver Detectedat location[21.2092, 81.4285]
Sent from your Twilio trial account - Drowsy driver Detectedat location[21.2092, 81.4285]
Sent from your Twilio trial account - Drowsy driver Detectedat location[21.2092, 81.4285]