Drowsiness Detection And Alerting System

SmartInternz

Shrenik Abrol 20BCE1436

Drive Link:

https://drive.google.com/drive/folders/1DvitGNkeo3Tu_b6qDxSxmSTaxeft_JwW

1. Introduction

The main cause of road accidents is a distracted driver behind the wheel. According to the recent statistics Drowsiness and Fatigue have been the top main reasons for these unfateful incidents. The concentration of the driver is tested to alert them when they are drowsy. It is implemented using an eye aspect ratio algorithm that sends the data of movements of eyelids over the cloud. The driver is alerted by an alarm sound and a message through a mobile application that wakes them up.

This project uses IBM IoT to look at the driver and find out if the driver is drowsy or awake and sends them a message when they are drowsy. It can also be used in various other forms of life activities requiring concentration.

2. Literature Survey

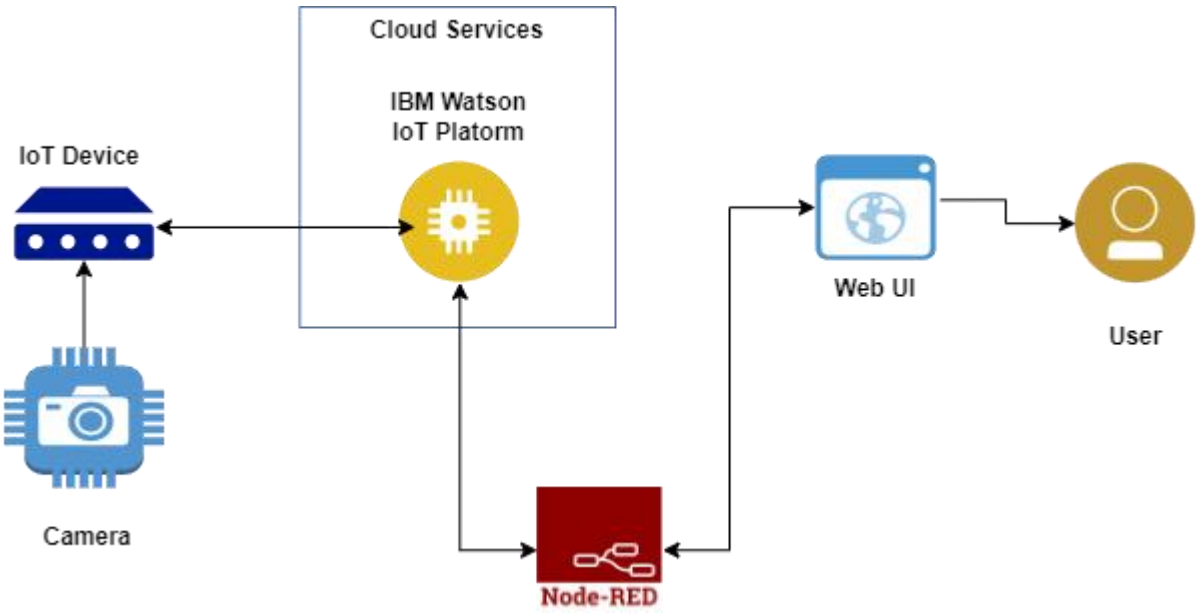
The existing approaches to solve the problem of people being drowsy is to use cameras and eye tracking systems to find if the driver is sleepy or not.

It checks yawning frequency, head movement and various other ways to see the level of fatigue of the driver.

My solution to this is to use IBM IoT and python with Node-red to check the movement of the users eyes and figure out how often their eyes are closed for more than half a second to not take blinking into effect but have a way to tell if the driver is drowsy or just blinking.

For this, a python code that uses OpenCV that checks the status of the eyes and sends flags to verify if the driver is drowsy or not. 0 being active while 1 being drowsy so whenever the flag is valued at 1 he is drowsy and the alert is sent telling the user to wake up.

3. Theoretical Analysis



Software Requirements:

IBM Watson IoT Platform, Node-Red, Python

Hardware Requirements:

Webcam, Laptop, Mobile

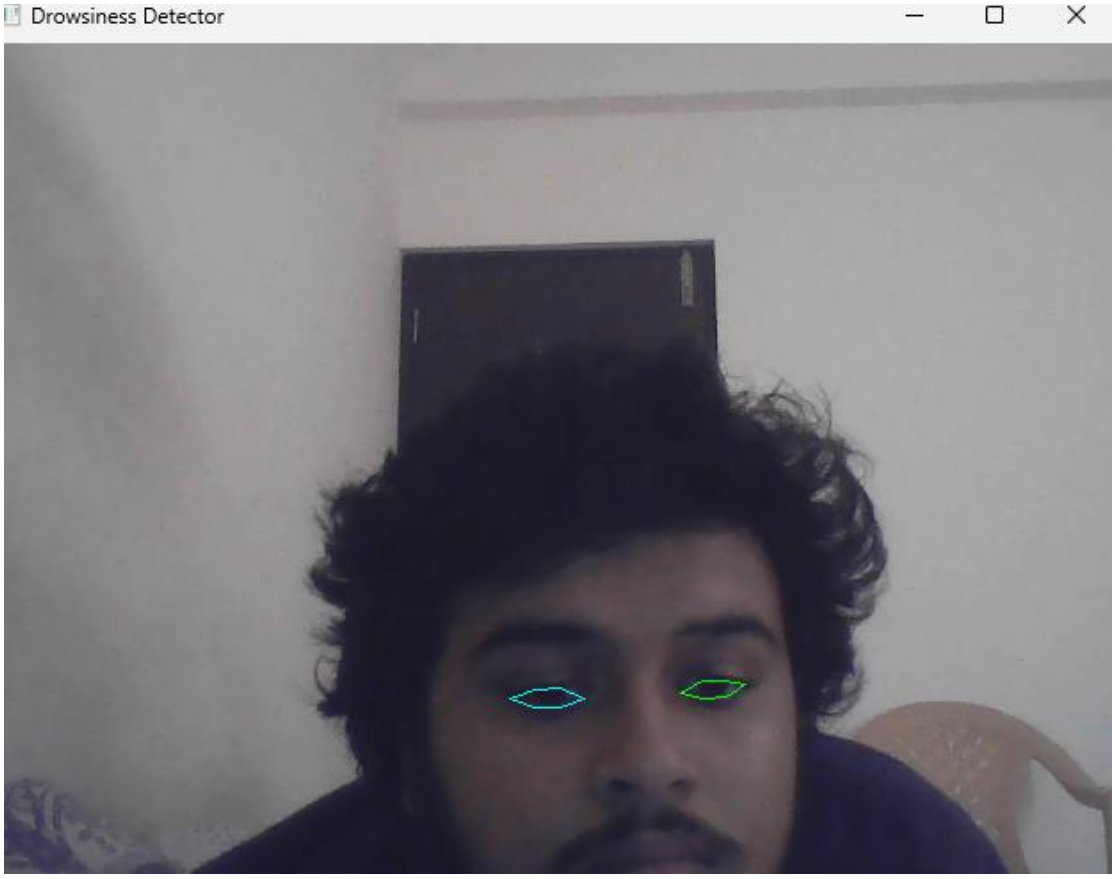
4. Experimental Investigations

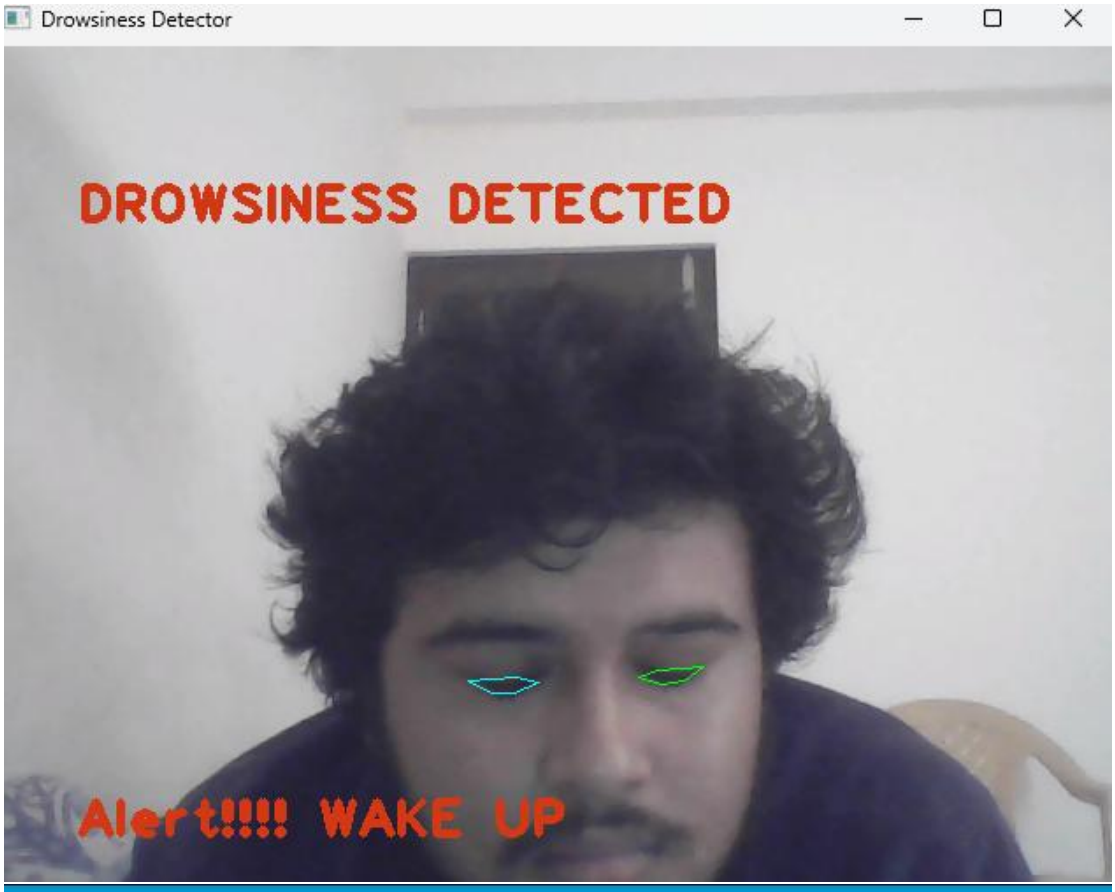
The analysis during the work is that the system works when it can focus on the eyes so the user needs to look towards the road not necessarily to the camera, so the system works.

The project is working nicely and sending alerts and showing the result in the dashboard, node-red and IBM Watson IoT.

5. Result

The final findings are that whenever the eyes of the user is not awake an alert is sent to the dashboard and the screen as well as an audio is played telling the user to wake up.





Drowsiness

Status **Good to go!**

6. Advantages & Disadvantages

The advantages of the solution are -

- It keeps the driver awake
- It informs him whenever he is feeling drowsy
- It helps keep the drivers eyes focused on the road

The disadvantages of the solution are -

- It can be distracting and annoying for the passengers
- It can take wrong data and send an alert if the driver is looking sideways and his eyes are not matching with the camera.
-

7. Applications

The applications for this project is vast as it can be used in any activity that requires the user to stay active and focused like studying, driving, monitoring vitals, etc.

8. Conclusions

The project will work on the system that a driver falling asleep on the road is dangerous therefore finding a way to keep him/her from shutting his eyes is to present ways to keep him up like automatically figuring out if he is drowsy and alerting him by telling him to wake up by showing it and also being told.

9. Future Scope

The enhancements that can be made to the project is to include ways for it to be connected directly to our cars and send messages straight to the HUD of the car.

It can also be used for kids studying who are likely to dose off, as well as office workers.

10. Bibliography

Source Code:

```
import cv2
import dlib
import requests
import pyttsx3
from scipy.spatial import distance

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "3j24p3"
deviceType = "abcd"
deviceId = "1234"
authMethod = "token"
authToken = "12345678"

def ibmstart(x):

    def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data['command'])
        print(cmd)

    try:
      deviceOptions = {"org": organization, "type": deviceType,
"id":   deviceId,   "auth-method":   authMethod,   "auth-token":
authToken}
      deviceCli = ibmiotf.device.Client(deviceOptions)
      #.........................................

    except Exception as e:
      print("Caught exception connecting device: %s" % str(e))
      sys.exit()

    deviceCli.connect()
    data = { 'Status' : x}
    #print data
    def myOnPublishCallback():
        print ("Published Status = %s" % x, "to IBM Watson")

    success = deviceCli.publishEvent("DD", "json", data, qos=0,
on_publish=myOnPublishCallback)
```

```
    if not success:
        print("Not connected to IoTF")


    deviceCli.commandCallback = myCommandCallback
    deviceCli.disconnect()



# INITIALIZING THE pyttsx3 SO THAT
# ALERT AUDIO MESSAGE CAN BE DELIVERED
engine = pyttsx3.init()

# SETTING UP OF CAMERA TO 1 YOU CAN
# EVEN CHOOSE 0 IN PLACE OF 1
cap = cv2.VideoCapture(0)

# FACE DETECTION OR MAPPING THE FACE TO
# GET THE Eye AND EYES DETECTED
face_detector = dlib.get_frontal_face_detector()

# PUT THE LOCATION OF .DAT FILE (FILE FOR
# PREDECTING THE LANDMARKS ON FACE )
dlib_facelandmark                                                =
dlib.shape_predictor("C:/Users/shren/Desktop/project/shape_pre
dictor_68_face_landmarks.dat")

# FUNCTION CALCULATING THE ASPECT RATIO FOR
# THE Eye BY USING EUCLIDEAN DISTANCE FUNCTION
def Detect_Eye(eye):
  poi_A = distance.euclidean(eye[1], eye[5])
  poi_B = distance.euclidean(eye[2], eye[4])
  poi_C = distance.euclidean(eye[0], eye[3])
  aspect_ratio_Eye = (poi_A+poi_B)/(2*poi_C)
  return aspect_ratio_Eye


# MAIN LOOP IT WILL RUN ALL THE UNLESS AND
# UNTIL THE PROGRAM IS BEING KILLED BY THE USER
while True:

  null, frame = cap.read()
  flag=0
  gray_scale = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

  faces = face_detector(gray_scale)

  for face in faces:
    face_landmarks = dlib_facelandmark(gray_scale, face)
```

```
leftEye = []
rightEye = []

# THESE ARE THE POINTS ALLOCATION FOR THE
# LEFT EYES IN .DAT FILE THAT ARE FROM 42 TO 47
for n in range(42, 48):
  x = face_landmarks.part(n).x
  y = face_landmarks.part(n).y
  rightEye.append((x, y))
  next_point = n+1
  if n == 47:
    next_point = 42
  x2 = face_landmarks.part(next_point).x
  y2 = face_landmarks.part(next_point).y
  cv2.line(frame, (x, y), (x2, y2), (0, 255, 0), 1)

# THESE ARE THE POINTS ALLOCATION FOR THE
# RIGHT EYES IN .DAT FILE THAT ARE FROM 36 TO 41
for n in range(36, 42):
  x = face_landmarks.part(n).x
  y = face_landmarks.part(n).y
  leftEye.append((x, y))
  next_point = n+1
  if n == 41:
    next_point = 36
  x2 = face_landmarks.part(next_point).x
  y2 = face_landmarks.part(next_point).y
  cv2.line(frame, (x, y), (x2, y2), (255, 255, 0), 1)

# CALCULATING THE ASPECT RATIO FOR LEFT
# AND RIGHT EYE
right_Eye = Detect_Eye(rightEye)
left_Eye = Detect_Eye(leftEye)
Eye_Rat = (left_Eye+right_Eye)/2

# NOW ROUND OF THE VALUE OF AVERAGE MEAN
# OF RIGHT AND LEFT EYES
Eye_Rat = round(Eye_Rat, 2)

# THIS VALUE OF 0.25 (YOU CAN EVEN CHANGE IT)
# WILL DECIDE WHETHER THE PERSONS'S EYES ARE CLOSE OR NOT
if Eye_Rat < 0.25:
  cv2.putText(frame, "DROWSINESS DETECTED", (50, 100),
        cv2.FONT_HERSHEY_PLAIN, 2, (21, 56, 210), 3)
  cv2.putText(frame, "Alert!!!! WAKE UP", (50, 450),
        cv2.FONT_HERSHEY_PLAIN, 2, (21, 56, 212), 3)

  # CALLING THE AUDIO FUNCTION OF TEXT TO
  # AUDIO FOR ALERTING THE PERSON
```

```
        engine.say("Alert!!!! WAKE UP")
        flag=1
        engine.runAndWait()



    cv2.imshow("Drowsiness Detector", frame)
    print(flag)
    ibmstart(flag)

    key = cv2.waitKey(9)
    if key == 27:
      break

# Disconnect the device and application from the cloud
#deviceCli.disconnect()
cap.release()
cv2.destroyAllWindows()
```