MINI PROJECT REPORT
ON
“DRIVER DROWSINESS DETECTION SYSTEM”

Submitted in

Partial Fulfillment of requirements for the Award of Degree

of

Bachelor of Technology

In

Computer Science and Engineering

By

(Group Number:11)

Ashutosh Mishra (1816410079)
Dakshita Mishra (1816410103)
Avinash Kumar Pandey (1816410084)

Under the supervision of
Kumar Saurabh
(Asst. Professor CSE Dept.)

Pranveer Singh Institute of Technology.
Kanpur - Agra - Delhi National Highway - 2
Bhauti -Kanpur - 209305.
(Dr. A.P.J. Abdul Kalam Technical University)

Acknowledgements

We express our great sense of gratitude to our respected and learned guides, **Asst. Prof Mr. Kumar Saurabh** for their valuable help and guidance, we are thankful to them for the encouragement they have given us in completing this project.

We are also grateful to respected **Mr. Vishal Nagar (HOD CSE Dept.)** for permitting us to utilize all necessary facilities of the institution.

We are also thankful to all the other faculty and staff members of our department for their kind co-operation & help.

Lastly, we would like to express our apperception towards our classmates and our indebt ness to our parents for providing us the moral support and encouragement.

Ashutosh Mishra
Dakshita Mishra
Avinash Kumar Pandey

Abstract

Driver fatigue is one of the major causes of accidents in the world. Detecting the drowsiness of the driver is one of the surest ways of measuring driver fatigue. In this project we aim to develop a prototype drowsiness detection system. This system works by monitoring the eyes of the driver and sounding an alarm when he/she is drowsy. The system so designed is a non-intrusive real-time monitoring system. The priority is on improving the safety of the driver without being obtrusive. In this project the eye blink of the driver is detected. If the driver's eyes remain closed for more than a certain period of time, the driver is said to be drowsy and an alarm is sounded. The programming for this is done in OpenCV using the Haarcascade library for the detection of facial features

LIST OF FIGURES

1. Objective

To design a system that will detect drowsiness and take necessary steps to avoid accidents. The driver drowsiness detection system, being implemented in this project aims at being easily available and can be used with different types of vehicles.

2. Introduction

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident-avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time.

By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves the observation of eye movements and blink patterns in a sequence of images of a face.

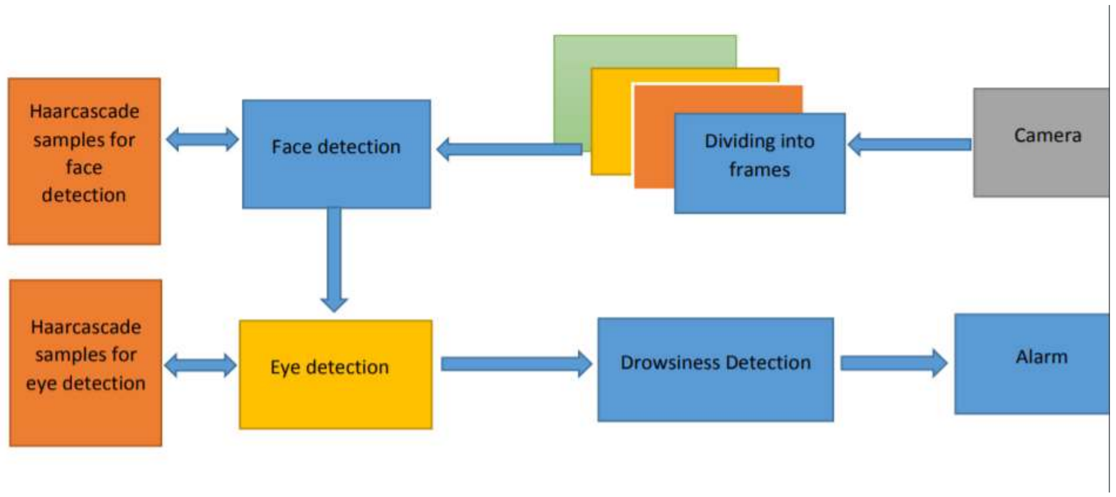
Initially, we decided to go about detecting eye blink patterns using OpenCV. OpenCV is an open-source computer vision library. It is designed for computational efficiency and with a strong focus on real time applications. It helps to build sophisticated vision applications quickly and easily. OpenCV satisfied the low processing power and high-speed requirements of our application. We have used the Haartraining applications in OpenCV to detect the face and eyes. This creates a classifier given a set of positive and negative samples. The steps were as follows: -

- Gather a data set of face and eye. These should be stored in one or more directories indexed by a text file. A lot of high-quality data is required for the classifier to work well.

- The utility application create samples () is used to build a vector output file. Using this file we can repeat the training procedure. It extracts the positive samples from images before normalizing and resizing to specified width and height.
- The Viola Jones cascade decides whether or not the object in an image is similar to the training set. Any image that doesn't contain the object of interest can be turned into negative sample. So, in order to learn any object, it is required to take a sample of negative background image. All these negative images are put in one file and then it's indexed.
- Training of the image is done using boosting. In training we learn the group of classifiers one at a time. Each classifier in the group is a weak classifier. These weak classifiers are typically composed of a single variable decision tree called stumps. In training the decision stump learns its classification decisions from its data and also learns a weight for its vote from its accuracy on the data. Between training each classifier one by one, the data points are reweighted so that more attention is paid to the data points where errors were made. This process continues until the total error over the dataset arising from the combined weighted vote of the decision trees falls below a certain threshold.

This algorithm is effective when a large number of training data are available. For our project face and eye classifiers are required. So we used the learning objects method to create our own haarclassifier.xml files. Around 7000 image samples are taken. Training them is a time intensive process. Finally, face.xml and haarcascade-eye.xml files are created. These xml files are directly used for object detection. It detects a sequence of objects (in our case face and eyes). Haarcascade-eye.xml is designed only for open eyes. So, when eyes are closed the system doesn't detect anything. This is a blink. When a blink lasts for more than 5 frames, the driver is judged to be drowsy and an alarm is sounded.

Flowchart of the proposed system: -



Algorithm Stages:

- Step 1- Image Capture
- Step 2- Dividing into Frames
- Step 3- Face Detection
- Step 4- Eye Detection
- Step 5- State of eye

3. Feasibility Study

- The project is easy to maintain as it uses modular programming approach.
- It is cost efficient as it uses open-source libraries and software are free to use.
- It is operationally feasible as it does not require very high-end machine to run.
- It is legally feasible as it uses free and open-source libraries and software which are free to use.

4.Technology Used

We have used different technology to make our project simple and working and easy for the community. We use

- **HTML**: - To create electronic documents that are displayed on the World Wide Web. Every web page you see on the Internet is written using one version of HTML code or another.
- **CSS:** For describing the presentation of Web pages, including colours, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens.
- **OpenCV: -** OpenCV is a library of Programming functions which are used in real-time Computer Vision. It is developed by intel corporation, willow garage, Itseez. we will be using OpenCV for gathering the images from webcam and feed them into a deep learning model.
- **TensorFlow: - It** is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. In our project, Keras uses TensorFlow as backend.
- **Keras: -Keras** is an open-source software library that provides a Python interface for artificial neural networks. **Keras** acts as an interface for the TensorFlow library. The model we used is built with Keras using Convolutional Neural Networks (CNN).
- **Pygame:- Pygame** is a set of Python modules designed for writing video games. **Pygame** adds functionality on top of the excellent SDL library. To play alarm sound we used pygame.

5.Hardware/Software Required

1. **Software Components** –
    - Python IDLE
    - Browser (Google chrome)
2. **Hardware Components –**
    - IP/Web Camera
    - Standalone Computer

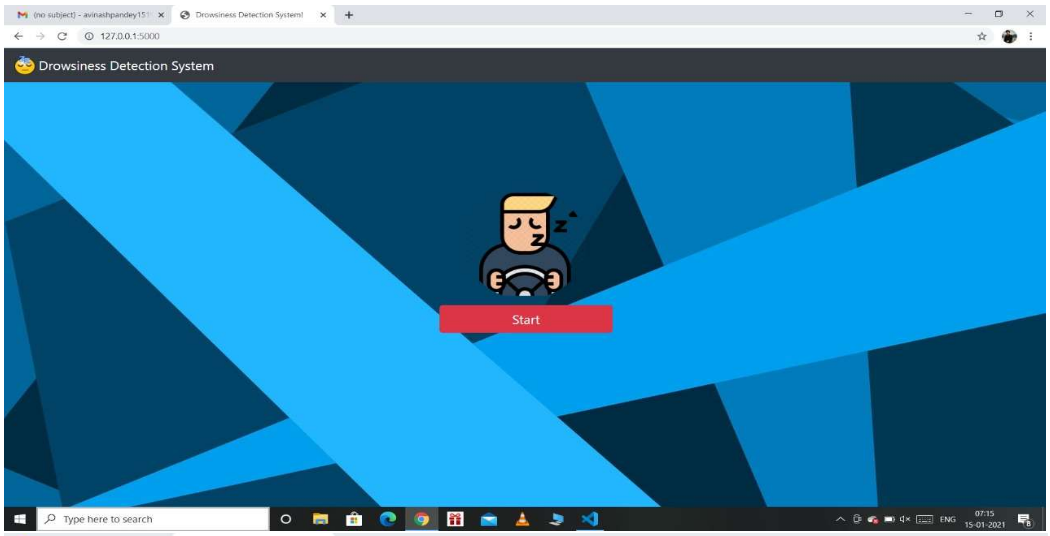
6.Coding

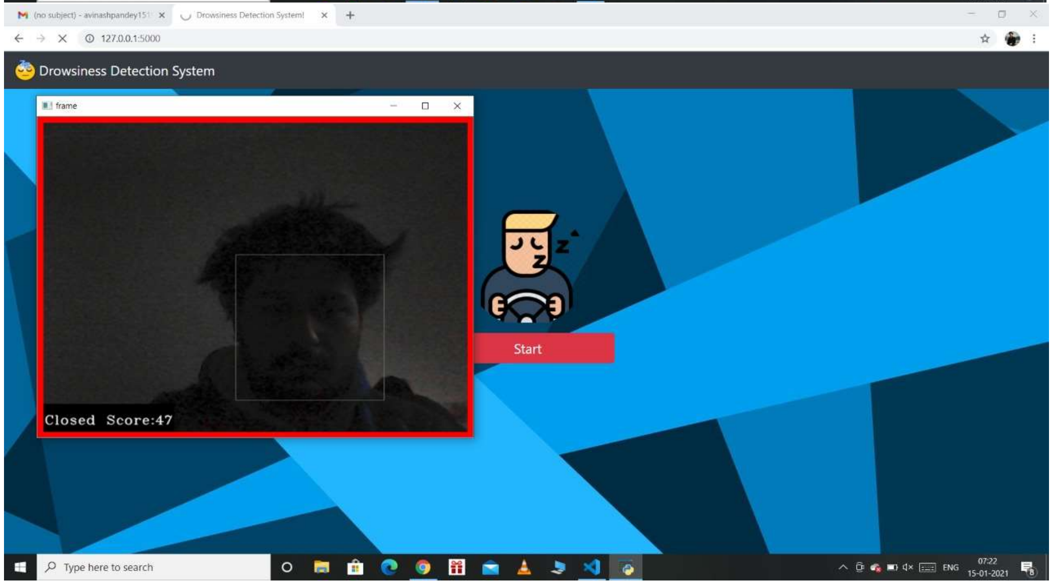
Frontend-

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootst
rap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" cross
origin="anonymous">
    <link rel="stylesheet" href="/static/css/style.css"/>
    <title>Drowsiness Detection System!</title>
  </head>
  <body>
    <!-- Image and text -->
<nav class="navbar navbar-dark bg-dark">
  <a class="navbar-brand" href="#">
    <img src="https://webstockreview.net/images/clipart-sleeping-
18.png" width="30" height="30" class="d-inline-block align-top" alt="">
    Drowsiness Detection System
  </a>
</nav>


    <div class="container h-80">
      <div class="row align-items-center h-100">
          <div class="col-3 mx-auto">
              <div class="text-center">
                  <img id="profile-img" class="rounded-circle profile-
img-card" src="https://cdn.iconscout.com/icon/premium/png-512-
thumb/sleepy-while-driving-832116.png" />
                  <p id="profile-name" class="profile-name-card"></p>
                  <form  class="form-
signin" method="POST" action="/start">
                  <button class="btn btn-danger btn-lg btn3d btn-
block btn-signin" type="submit"> Start</button>
                  </form><!-- /form -->
              </div>
          </div>
      </div>
      </div>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
  </body>
</html>
```


Drowsiness Detection System
127.0.0.1:5000
Start
Type here to search


Drowsiness Detection System
127.0.0.1:5000
frame
Closed Score:47
Start
Type here to search

Backend-

- **Main program file-**

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time
mixer.init()
sound = mixer.Sound('alarm.wav')


face=cv2.CascadeClassifier('haar
cascadefiles\haarcascade_frontalface_alt.xml')
leye=cv2.CascadeClassifier('haar
cascade files\haarcascade_lefteye_2splits.xml')
reye=cv2.CascadeClassifier('haar
cascade files\haarcascade_righteye_2splits.xml')


lbl=['Close','Open']
model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred= [99]
lpred= [99]

while(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces=face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=
(25,25))
    left_eye = leye.detectMultiScale(gray)
    right_eye = reye.detectMultiScale(gray)

cv2.rectangle(frame,(0,height50),(200,height),(0,0,0),thickness=cv2.FILLED)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

    for (x,y,w,h) in right_eye:
        r_eye=frame[y:y+h,x:x+w]
        count=count+1
        r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye,(24,24))
        r_eye= r_eye/255
        r_eye=  r_eye.reshape(24,24,-1)
        r_eye = np.expand_dims(r_eye,axis=0)
        rpred = model.predict_classes(r_eye)
        if(rpred[0]==1):
            lbl='Open'
        if(rpred[0]==0):
            lbl='Closed'
        break

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = model.predict_classes(l_eye)
        if(lpred[0]==1):
           lbl='Open'
        if(lpred[0]==0):
           lbl='Closed'
        break

    if(rpred[0]==0 and lpred[0]==0):
        score=score+1
cv2.putText(frame,"Closed",(10,height20),font,1,(255,255,255),1,cv2.LINE_
AA)
    else:
        score=score-1

cv2.putText(frame,"Open",(10,height20),font,1,(255,255,255),1,cv2.LINE_A
A)
     if(score<0):
        score=0

cv2.putText(frame,'Score:'+str(score),(100,height20),font,1,(255,255,255),1,c
v2.LINE_AA)
    if(score>15)
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
          sound.play()
        except:
           pass
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0), (width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

7.Conclusion

The driver abnormality monitoring system developed is capable of detecting drowsiness, drunken and reckless behaviors of driver in a short time. The Drowsiness Detection System developed based on eye closure of the driver can differentiate normal eye blink and drowsiness and detect the drowsiness while driving. The proposed system can prevent the accidents due to the sleepiness while driving. The system works well even in case of drivers wearing spectacles and even under low light conditions if the camera delivers better output. Information about the head and eyes position is obtained through various self-developed image processing algorithms. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for too long, a warning signal is issued. processing judges the driver's alertness level on the basis of continuous eye closures.

8.Future Scope

1-The model can be improved incrementally by using other parameters like blink rate, yawning, state of the car, etc. If all these parameters are used it can improve the accuracy by a lot.

2-We plan to further work on the project by adding a sensor to track the heart rate in order to prevent accidents caused due to sudden heart attacks to drivers.

3- Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when the user is asleep and stop the video accordingly. It can also be used in application that prevents user from sleeping.

9. References

- https://en.wikipedia.org/wiki/Driver_drowsiness_detection
- Beyeler, M. (2017). Machine Learning for OpenCV: Intelligent image processing with Python.
- https://www.sciencedirect.com/science/article/pii/S18770509183

GitHub Link- https://github.com/avinashpandey981/Drowsiness-detection-system.git