PARSHVANATH CHARITABLE TRUST

A.P. SHAH INSTITUTE OF TECHNOLOGY

Computer Engineering Department

A.P. Shah Institute of Technology
— G.B.Road,Kasarvadavali, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI
Academic Year 2020-2021

A Project Report on

Detection System for Driver's Safety using Face Detection, Drowsiness Detection and Lane Detection

Submitted in partial fulfillment of the degree of
Bachelor of Engineering(Sem-8)
in
Computer Engineering
By

Nimali Keny (17102062)
Pooja Maniyar (18202012)
Purti Lalan (17102046)
Purvi Lalan (17102047)

Under the Guidance of
Prof. Krupi Saraf

1.Project Conception and Initiation

1.1 Abstract

The development of technology allows introducing more advanced solutions in everyday life. This makes work less exhausting for employees, and also increases work safety. A new approach towards automobile safety and security with autonomous region primarily based automatic automotive systems is projected in this project idea.

Nowadays, more and more professions require long-term concentration. In recent time's fatigue connected crashes have greatly increased. Drivers must keep a close eye on the road, so they can react to sudden events immediately. Driver fatigue often becomes a direct cause of many traffic accidents. Therefore, there is a need to develop the systems that will detect and notify a driver of her/his bad psychophysical condition, which could significantly reduce the number of fatigue-related car accidents. However, the development of such systems encounters many difficulties related to fast and proper recognition of a driver’s fatigue symptoms. Some of the technical possibilities are to implement driver drowsiness detection systems and lane detection systems. The proposed system may be evaluated for the effect of drowsiness warning under various operation conditions and detect lanes while driving a car. We have tried to obtain the experimental results, which will propose the expert system, to work out effectively for increasing safety in driving. The detail of image processing technique and the characteristics have also been studied.

1.2 Objectives

- One of the many objectives involved during the training of an autonomous driving car for driver's safety is lane detection.
- To alert the driver when he is drowsy to prevent accidents.
- The system accurately monitors the open or closed state of the eye.
- To minimize development and maintenance costs.
- To increase safety while driving.

1.3 Literature Review

This paper proposes an idea of Hough lane detection technique which can detect discontinuous lanes as well. The proposed system, is supposed to have two webcams, one to detect the lane and the other to monitor the face of the driver. [1]

The aim of this paper is to develop an algorithm for drowsiness or alertness 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 [2]

Nowadays, more and more professions require long-term concentration. People, who work for transportation business (like car and truck drivers),must keep a close eye on the road, so they can react to sudden events (e.g. road accidents, animals on the road, etc.) immediately. Long hours of driving causes the driver fatigue and, consequently, reduces her/him response time. In order to reduce the number of road accidents resulting from a driver fatigue, it is of great importance to introduce to the automotive industry a system that would effectively detect the first signs of a fatigue and notify the driver. [3]

1.3 Literature Review (Contd.)

In this lane line detection project, we use OpenCV. Before detecting lane lines, we masked remaining objects and then identified the line with Hough transformation. We have done this using the concepts of computer vision using OpenCV library. To detect the lane we have to detect the white markings on both sides on the lane. Using computer vision techniques in Python, we will identify road lane lines in which autonomous cars must run. Canny edge is used to calculate the intensities [4]

This paper proposes a novel lane departure algorithm for detecting lane markers in images acquired from a forward-looking vehicle-mounted camera.It is a real-time algorithm for lane detection and tracking, which is also simple to implement.Visual field of a downward side camera is narrow, and only a few information of the lanes can be applied.It detects the left and right lane markings separately, whereas most of the previous work uses a fixed-width lane model. It combines lane detection and tracking into a single algorithm. Canny operation provides the threshold automatically. [5]

1.4 Problem Definition

- Our project is divided into 3 sub-parts on which we have been working in concern of driver’s safety. The parts are as follows:
  - Face Recognition
  - Drowsiness Detection
  - Lane Detection
- In India, on average about 1214 crashes happen on a daily basis. There are various reasons that can cause road accidents such as reckless driving, speeding, drunk driving, etc.
- Due to continuous journey, tiredness and fatigue can easily arise in drivers which may result in critical road accidents. From the survey, it is seen that almost 20% of accidents are caused by fatigue and 50% of accidents happen on the road.
- A solution to this problem is to identify when the driver is falling asleep and alert the passengers of the situation so that appropriate measures can be taken.
- In any driving scenario, lane lines are also an essential component for indicating traffic flow and where a vehicle should drive.

1.5 Scope

- The main idea behind this project is to develop a system which can detect fatigue of any human and can issue a timely warning.
- This project will be helpful in detecting driver fatigue in advance and will give warning output in the form of alarm and pop-ups.
- Moreover, the warning will be deactivated manually rather than automatically. This will directly give an indication of drowsiness/fatigue which can be further used as a record of driver performance.
- The system can be implemented in the Automobile Industry focusing mainly on Cars and Trucks.
- The Lane Detection System can be implemented in self-driving cars.
- Using the night vision camera, we can detect the lane at night as well.
- Since this system would be inbuilt in car, it will automatically start detecting the eyes as well as lane and the user can’t disable it.

1.6 Technology stack

Required for this project:

- Python 3.6 or above versions
- Integrated Camera or Webcam
- Anaconda Navigator: used for execution

Used for this project (Minimum requirement):

- Processor: - Core i3
- RAM: - 4GB

Libraries Used:

- OpenCV
- DLIB
- Subprocess
- Scipy
- Numpy
- Queue
- Matplotlib
- Tkinter

1.7 Benefits for environment & Society

- System is fully automated as it does not require any manual input from the driver.
- The system helps to Decrease road accidents.
- It can be implemented in heavy vehicles like trucks to avoid crashes.
- Detects if a driver is conscious or not. If he/she is not conscious, the system will give appropriate warning.
- Alert drivers result in safe driving which in turn results in less road accidents.
- System does not distract the driver as the driver is only notified if the system detects the driver's drowsiness, else the system runs silently in the background without distracting the driver.
- Due to the non-obstructive nature of these methods they are more practically applicable.
- Lane detection can be further used in self-driving cars.
- Car drivers, truck drivers, taxi drivers, etc. should be allowed to use this solution to increase the safety of the passengers, other road users and the goods they carry.

2. Project Design

2.1 Proposed System

Basically, the entire UI of the project is divided into three parts which are:

1. Face detection
2. Drowsiness detection
3. Lane detection

- Face Detection:

  Face Detection initially detects your face using an integrated camera or wired camera. It detects your face using a Haar Cascade file and draws a box around the face.

- Drowsiness Detection:

  Once your face is detected using an integrated camera, it detects your eyes and starts counting your eye blinks by incrementing the blink count. It continuously keeps on detecting your blink and as soon as it detects that the eyes are closed for more than the expected number of seconds (in our case, 1.5 seconds) it raises an alarm.

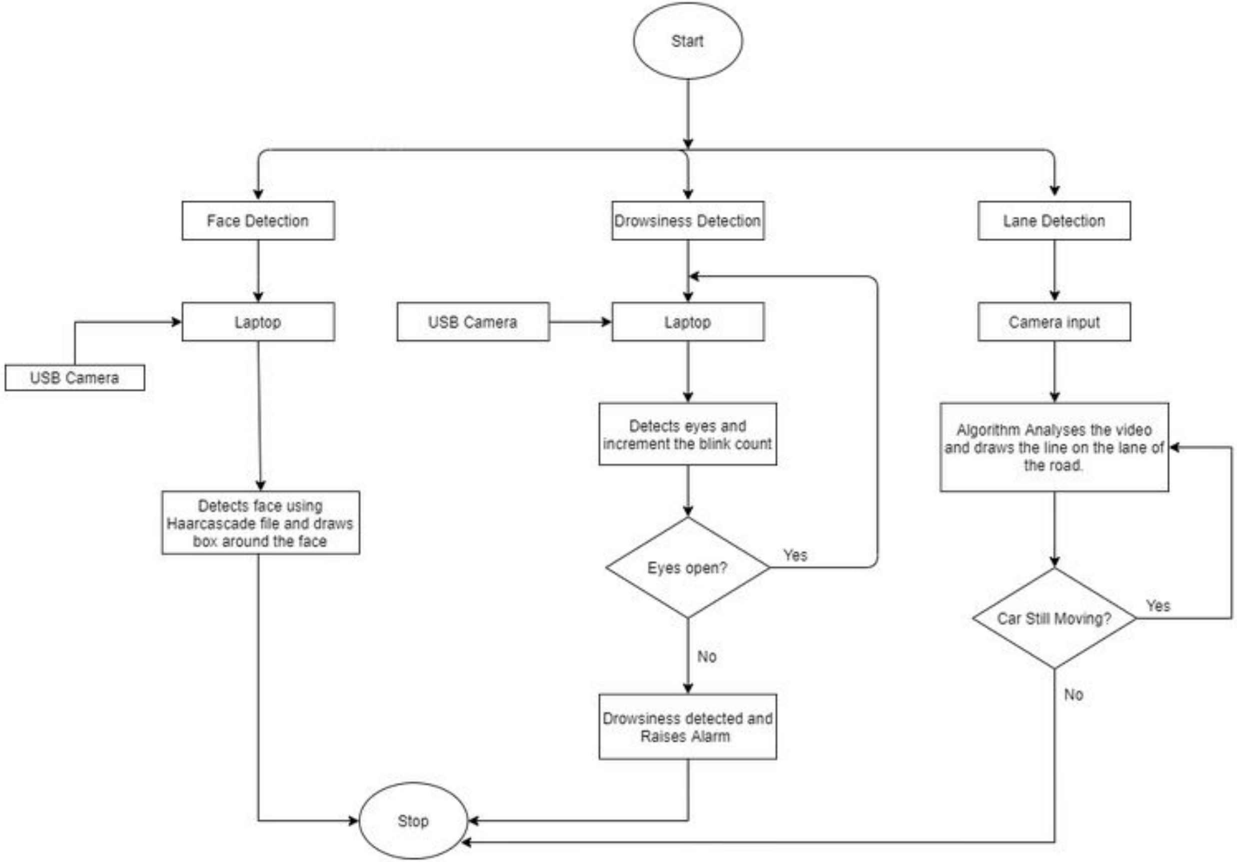
2.1 Proposed System (Contd.)

- Lane Detection:

  Tracing white markings (or similar ones) on the road, capturing and processing images using a camera which is mounted in front of the car is called Lane Detection. Lane detection and its tracking is a significant element for vision based driver assistance systems. Main purpose of these systems is to prevent crashes due to unintended lane departure movements produced by tired drivers. Virtual lines would be drawn on both sides of the road which is seen on the car's dashboard to detect the road.

  Lane detection has been a challenging task for all types of roads such as road having single marking, multiple-marking, arched roads, different lighting conditions, shadow of the trees, objects, blockage of visibility, dissimilar texture of the roads, and different weather circumstances. Lane detection can also be applied for the roads having raised pavement markers.

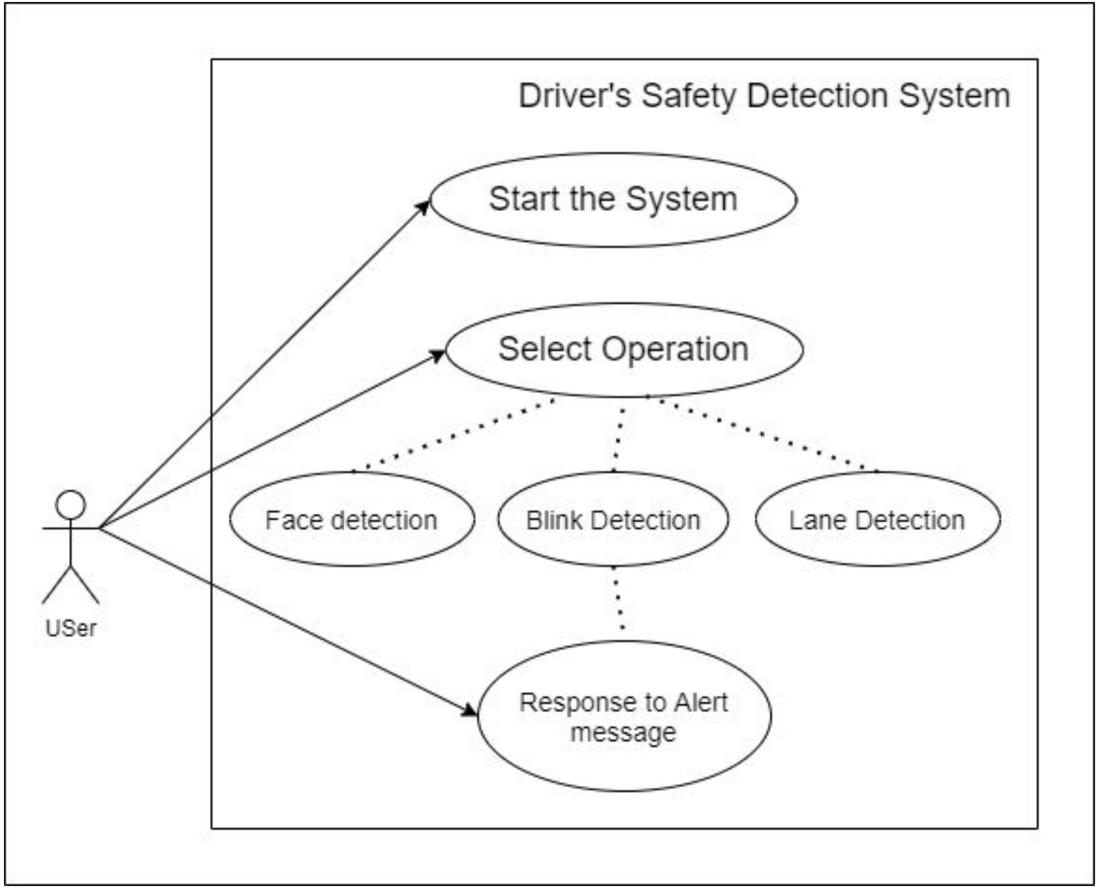
2.2 Design(Flow Of Modules)


Start
Face Detection
Drowsiness Detection
Lane Detection
Laptop
USB Camera
USB Camera
Laptop
Camera input
Detects eyes and increment the blink count
Algorithm Analyses the video and draws the line on the lane of the road.
Detects face using Haarcascade file and draws box around the face
Eyes open?
Yes
Car Still Moving?
Yes
No
No
Drowsiness detected and Raises Alarm
Stop

2.3 Description Of Use Case

A use case diagram is a dynamic or behavior diagram in UML. A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

The use case of driver’s safety detection is shown in the diagram beside.



2.5 Module Explanation

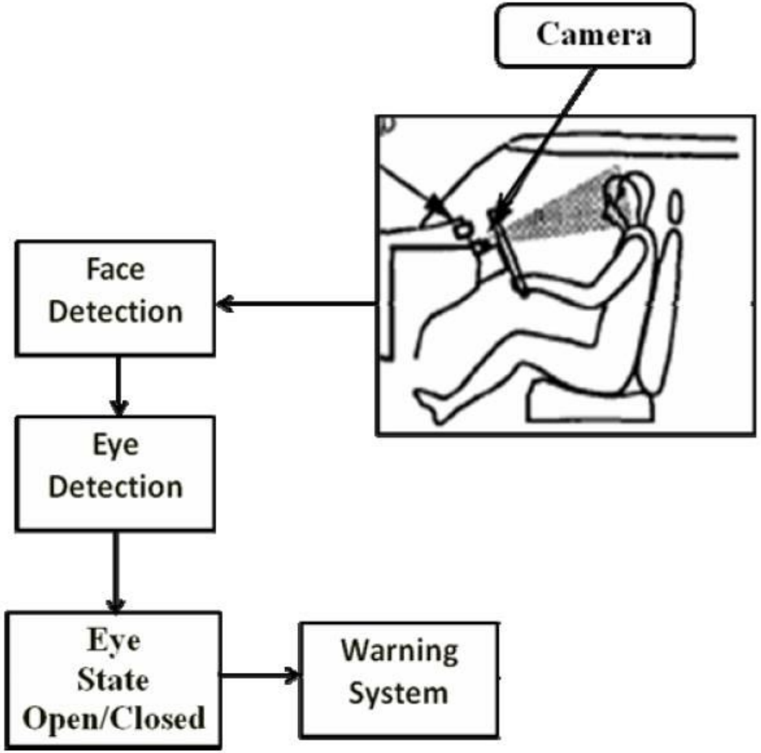
Face Detection

- The System takes input from the integrated camera or the wired camera.
- Face Detection uses Haar Cascade feature-based cascade classifiers as an effective object detection methodology planned by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.
- The haar cascade algorithm will convert the image to grayscale to analyze the white area which will be obtained once the image is converted to grayscale.
- The algorithm will then match that grayscale image with the pretrained face images xml file.
- If the grayscale and co-ordinates among the xml file matches, then the algorithm draws a box around the face.

2.5 Module Explanation (contd.)

Blink Detection

- It is a hard test of endurance for drivers to take long distance driving. It is very difficult for them to pay attention to driving on the entire trip unless they have very strong willpower, patience, and persistence. Thus, the driver fatigue problem has become an important factor of causing traffic accidents. Driver drowsiness is a significant factor in a large number of vehicle accidents.
- Thus when the drowsiness module is selected through the UI of the system which would be provided at the start, the system will detect the drowsiness or fatigue by continuously monitoring the eyes of the driver.
- Once the recalibration process is finished using the trained dataset i.e., DLIB library, the module will produce virtual points around your eyes which will help the system to increment the blink counts.
- We can detect and access both the eye region by the following facial landmark index:
    - The right eye using [37, 38, 39, 40, 41, 42].
    - The left eye with [43, 44, 45, 46, 47, 48].
- The processed image is continuously monitored. If the driver’s eyes are found to be closed for more than 1.5 seconds, then it is considered to be sleeping and thus, the system will raise a warning alarm which will be integrated with the system.


Camera
Face Detection
Eye Detection
Eye State Open/Closed
Warning System

2.5 Module Explanation (Contd.)

Lane Detection

- Lane detection basically involves detection of the lane markings made on the road.
- An Integrated camera would be mounted on the front of the car which will divide the video into frames.
- When the Lane detection module would be selected through the UI of the system which will be provided at the beginning, the system will take some seconds to recalibrate the road marking.
- The system uses Canny edge detection and Hough Transform to detect lanes on the road. The Hough transform used is simple and faster for adequate detection of lanes of the road.
- The lane which will be detected would be seen on the car’s dashboard. Virtual lines would be drawn on both sides of the road which is seen on the car’s dashboard to detect the road and its curves.

3.Implementation

3.1 Proposed System

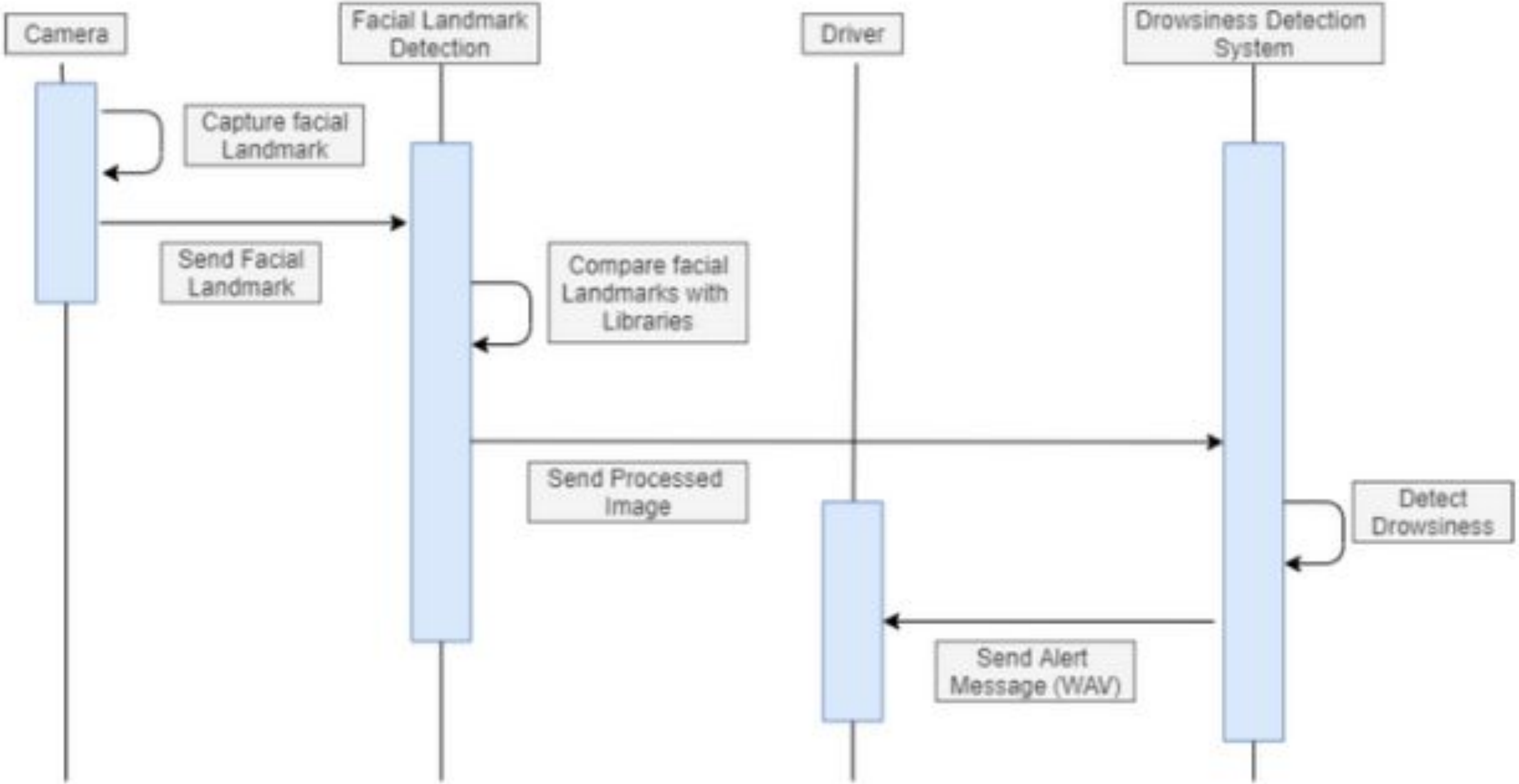
Face detection.

- In OpenCV we can also capture live streams with a camera.
- To create a video, you need to create a VideoCapture object. We used the read function to read the frames of the camera.
- Taking the coordinates of the faces in the faces array. Multiscale detects objects of different sizes in the input image and returns rectangles positioned on the face.
- The first argument is image, second is scale factor and third is MinNeighbours.
- The Haar cascade approach works with a sliding window approach. You can slide a window through the picture. So when this window is slided in picture resized and slided again, it detects many false positives.
- This increases face detection. So to eliminate false positives and get proper rectangles out of detection, neighbourhood approach is applied.
- If it is in the neighbourhood of other rectangles, then you can pass it further.
- The code will keep getting snapshots from the camera and apply the detect function to detect the face and draw rectangles.

Drowsiness Detection

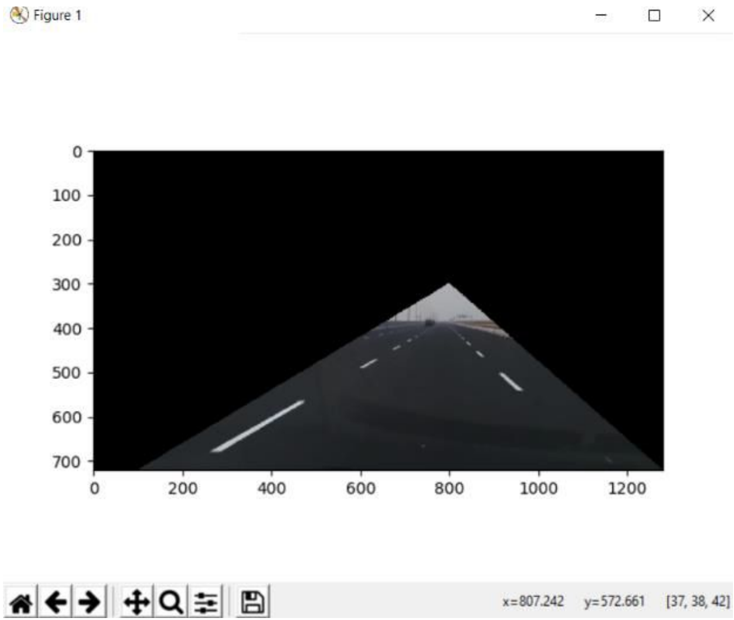
- Our blink detection system is divided into three parts and the basic follow for which is also shown in the diagram below.
  - In the first part we'll discuss the *eye aspect ratio* and how it can be used to determine if a person is blinking or not in a given video frame.
  - From there, we'll write Python, OpenCV, and dlib code to perform facial landmark detection and detect blinks in video streams.
  - Based on this implementation we'll apply our method to detecting blinks in example webcam streams along with video files.
- The basic process used motion analysis technique for eye detection, which involves the analysis of the involuntary blinks of a user of the system. After the system had been initialized, the eye was tracked using the EAR (Eye Aspect Ratio) and calculating the Euclidean distance. The major parameter subsequently used to detect drowsiness was the frequency of blinks, such that an alarm is triggered when it gets to a critical level.
- The system will locate the position of the eye by a motion analysis technique. This is used to detect the motion of the eye by analyzing the first involuntary blink of the user. In this case, each pixel assumes two discrete values of 0(eyes are closed) and (eyes are open); representing black and white respectively.

- We declare a variable Threshold value which records the count of consecutive frames in which the EAR was less than the *MINIMUM_EAR*.
- Detect all the faces in the image using dlib's *detector*
- Loop over each face and find the 68 landmarks using the getLandmarks of dlib
- Get the landmarks for the left and right eye and then send them to *eye_aspect_ratio()* to get EAR values for the left and right eye
- Find the cumulative EAR for both eye
- ear = (leftEAR + rightEAR) / 2.0
- If for the current frame the cumulative EAR is less than the threshold, increase the counter else reset the counter as we are interested in consecutive frames only.


Camera
Facial Landmark Detection
Driver
Drowsiness Detection System
Capture facial Landmark
Send Facial Landmark
Compare facial Landmarks with Libraries
Send Processed Image
Detect Drowsiness
Send Alert Message (WAV)

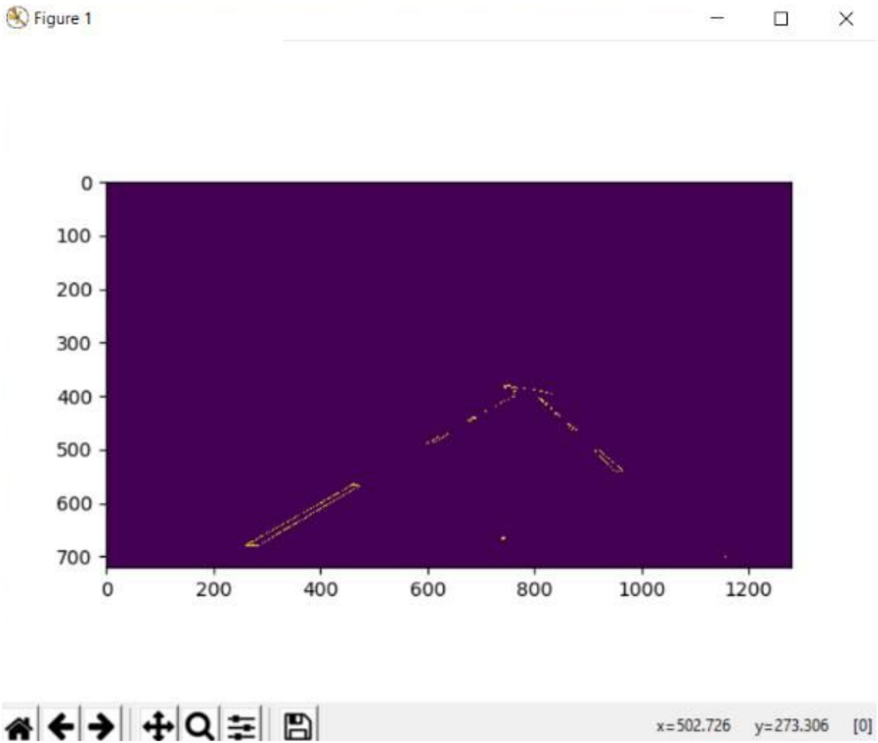
Lane Detection

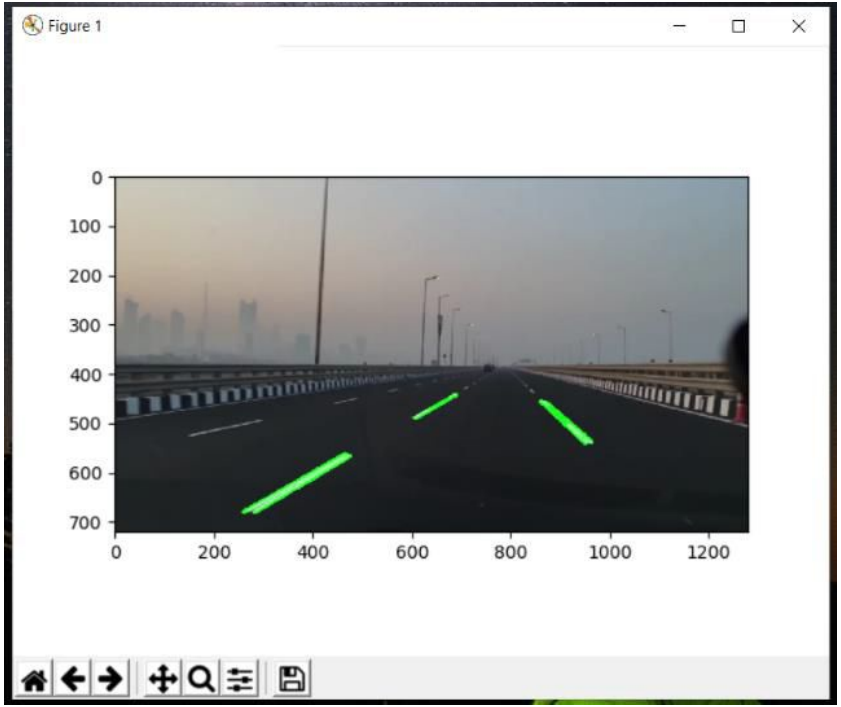
- The implementation of the lane detection system started with working on a single frame from the video and then keeping the values constant, we passed the video in place of an image.



- We have started with detecting the height and width of the image and deciding our *region_of_interest_vertices.*
- After we got our desired *region_of_interest_vertices* by trial and error we proceeded toward the *region_of_interest* function which will mask the image as shown below.

- In practice the road surface can be made up of many different colors due to shadows, different pavement style or age, which causes the color of the road surface and lane markings to change from one image region to another. Therefore, color images are converted into grayscale. However, the processing of grayscale images becomes minimal as compared to a color image. This function transforms a 24-bit, three-channel, color image to an 8-bit, single-channel grayscale image.
- Once we got our desired region of interest, we applied a canny edge detection algorithm to detect the lane in the unmasked part of the image.



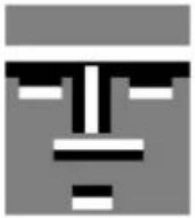


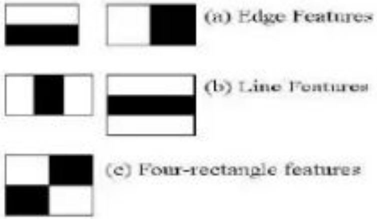
- After the edges are detected, we pass hough transform which is an inbuilt function in opencv, python. The Hough transform is simple and faster for adequate detection of lanes of the road. Here we have used the houghlinesp() function since the lines would not be discontinuous.
- Each frame obtained from the video is taken and Hough Transform for line detection is applied to it. Then according to the values present in the accumulator array, the lines are plotted in the original image to show the lanes.
- When the lines are detected, we have drawn the line on the image using the user defined function *draw_the_lines* which takes the image and detected lines as the input.
- After getting the desired output from the image, we simply pass the video in place of the image by using the inbuilt function *VideoCapture()* to capture the frames of the video.

3.1.2 Algorithms

Face Detection

In face detection, we have used the OpenCV library. It is a cross-platform library used to develop real time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. OpenCV's algorithm uses Haar-like features as input to basic classifiers. Haar cascade is a machine learning approach where a cascade function is trained from a lot of positive and negative images. For face detection, positive images means images of faces and negative images means images without faces. In an image, most of the image region is a non-face region. So it's better to check if a window is not a face region. If it is not, discard it in a single shot and don't process it again. This way, we can find more time to check a possible face region. For this they introduced a cascade of classifiers. Instead of applying all the 600 features on a window, group the features into different stages of classifiers and apply one by one. If a window fails at the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all the stages is the face region. OpenCV comes with a trainer as well as a detector. It already contains many pre-trained classifiers for face, smile, eyes etc.





Eye Detection

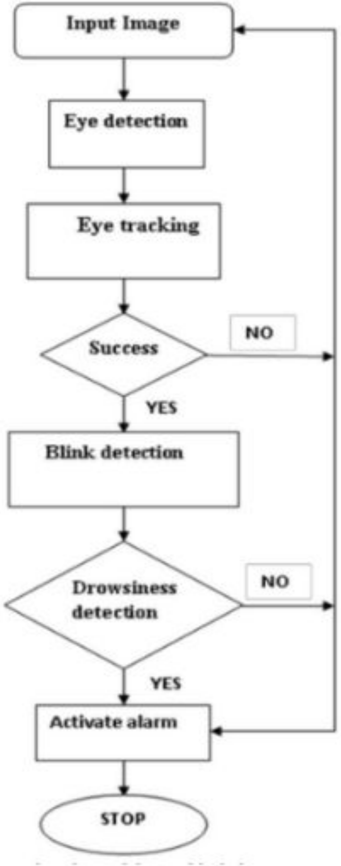
The facial landmark detector included in the dlib library is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014). This method starts by using:

a. 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.

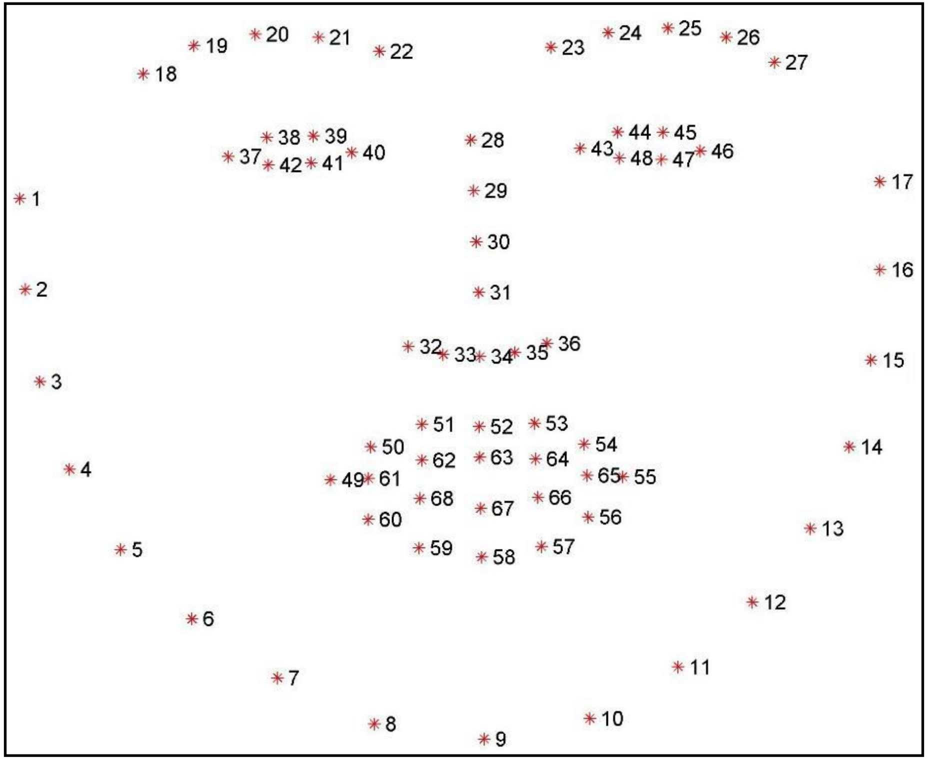
b. Priors, or more specifically, the probability of distance between pairs of input pixels. The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face. The indexes of the 68 coordinates.

We can detect and access both the eye region by the following facial landmark index:

The right eye using [37, 38, 39, 40, 41, 42].
The left eye with [43, 44, 45, 46, 47, 48].



The indexes of the 68 coordinates can be visualized on the image below:

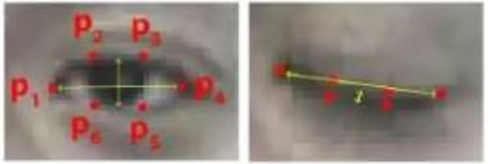


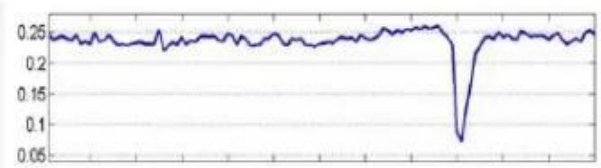
Eye Aspected Ratio Calculation:

For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

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

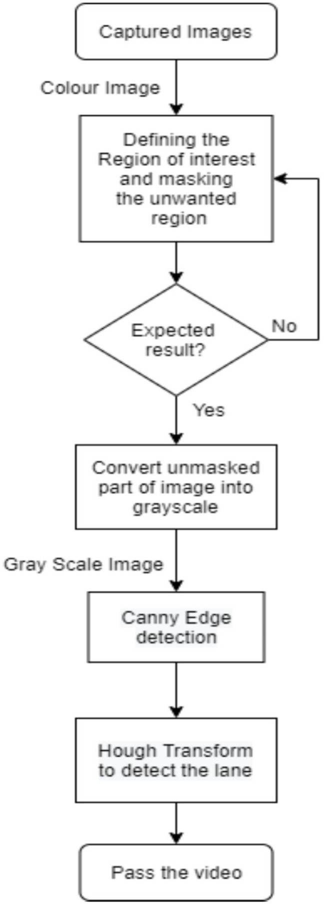
Where p1....p6 are the 2D landmark locations, depicted in the figure above. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals, and it is 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





Lane Detection:

1. Region of interest is decided over the image, which is captured from the video.
2. Unwanted area from the image is then masked by passing the *region_of_interest* function and if the result is as expected we proceed.
3. The input data is a color image sequence taken from a moving vehicle. A color camera is mounted inside the vehicle. It takes images of the environment in front of the vehicle, including the road, vehicles on the road, roadside, and sometimes incident objects on the road.
4. The on-board computer will capture the images in real time, and save them in the computer memory.
5. The lane detection system reads the image sequence from the memory and starts processing.
6. In this paper, the input to the algorithm is a color image. Therefore the first thing the algorithm does is to convert the image to a grayscale image in order to minimize the processing time.
7. Then the edge detector is used to produce an edge image by using a canny filter with automatic thresholding to obtain the edges; it will reduce the amount of learning data required by simplifying the image edges considerably.
8. Then edged image sent to the line detector after detecting the edges which will produce a right and left lane boundary segment.
9. The projected intersection of these two line segments is determined and is referred to as the horizon. The lane boundary scan uses the information in the edge image detected by the Hough transform to perform the scan. The scan returns a series of points on the right and left side.


Captured Images
Colour Image
Defining the Region of interest and masking the unwanted region
Expected result?
No
Yes
Convert unmasked part of image into grayscale
Gray Scale Image
Canny Edge detection
Hough Transform to detect the lane
Pass the video

Edge Detection:

Lane boundaries are defined by sharp contrast between the road surface and painted lines or some type of non-pavement surface. These sharp contrasts are edges in the image. Therefore edge detectors are very important in determining the location of lane boundaries. It also reduces the amount of learning data required by simplifying the image considerably, if the outline of a road can be extracted from the image. The edge detector implemented for this algorithm and the one that produced the best edge images from all the edge detectors evaluated was the 'canny' edge detector. The canny edge detector also has a very desirable characteristic in that it does not produce noise like the other approaches.

Line Detection

The line detector used is a standard Hough transform with a restricted search space. The Hough Line Transform is a transform used to detect straight lines. To apply the Transform, first an edge detection preprocessing is desirable.

3.1.3 Pseudo Code

Face Detection

First we Import OpenCV library to develop real time computer applications like face detection and object detection.

Then we pass the xml file to the face cascade object declared.

OpenCV algorithm uses haar-like features as input to basic classifiers. It comes with a trainer as well as detector and contains many pre-trained classifiers for face, smile etc.

Then we create a VideoCapture object to capture a live stream with a camera. Sometimes the capture is not initialised. So we use cap.isOpened function to check whether it's initialised or not.

It returns a boolean value for the same.

Later we convert the image into grayscale conversion.

We also pass the detectMultiscale function to the faces array. This will return the face coordinates and draw the rectangle around it.

The thickness and the color of the rectangle can also be adjusted.

The imshow function displays an image in the window.

Finally to exit the program the user needs to press q key on the keyboard.

cap.release function will turn off the camera and destroy function will close the window.

Blink Detection

Import all the modules like dlib, sys, openCV, numpy, scipy and Queue.

Set the threshold to 0.7. Then we retrieve face information and predict the landmarks on input images and video streams.

We set the left eye and right eye index values and compute the gamma value. It is used to optimise the usage of bits and display images correctly on screen.

Then we use histogram equalizer to adjust the contrast of the image by effectively spreading out frequency intensity values. We create alert alarms by implementing multithreading.

Next we calculate the EAR ratio by euclidean distance. A mask is created of a specific shape to check eye status. Then we draw a convex polygon on the exterior of both the eyes.

Eye status is calculated to check if the user is opening or closing his eyes. Thereby calculating the no. of blinks. Interpolation method is used to resize the image and frames.

Then we capture frame by frame video. This takes place in seconds and returns a string representing local time. Then we draw a string on the image displaying the required message and an image is displayed in a window. We use MPJG frames to load avi files.

If the eye satus of the user is closed for more than 1.5 seconds then an alarm sound file is supplied to start the thread and the alarm is played in background. Otherwise the eye aspect ratio is not below the blink threshold. So we reset the blink counter and alarm.

To close the program the user has to press Esc key of the keyboard.

Lane Detection

```
import matplotlib, openCV and numpy libraries

Define a function region_of_interest and take image and vertices as two parameters
    Write a method to mask the image
    Match the mask color by setting it to 225
    Fill the polygon by passing masked image and match mask color as the two parameters
    return the macked image

Define a function to draw line on image and input image and lines as two parameters
    copy the passed image
    Create a blank image which matches the original dimensions

    for loop to draw continuous lines:
        for loop for points x1,y1 to x2, y2:
        draw a line on blank image from 1st point to 2nd point of blue color and having thickness of 10
    return image
```

Define a function which will return the actual height and width of the image

decide the region of interest vertices by referring to the height and width shown in the window created by matplotlib.

Convert the image into grayscale

Pass the canny edge detection algorithm on the grayscale image and set 100, 120 as the threshold values

Crop the image by passing the canny edge detection and region of interest vertices in the region of interest function.

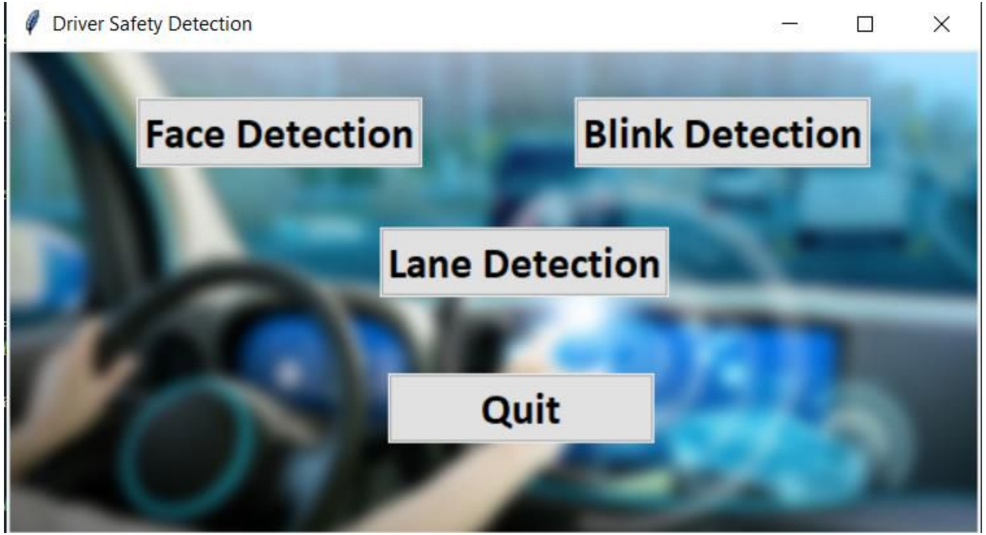
To mark the lines we will use the hough lines probability function, which is a predefined function of openCV. We will pass a cropped image as input array, rho value as 2, theta value as pi/180, threshold value as 50 and get an output array. with minimum line length of 40 and maximum line gap of 100.

we will get image with lines by using draw the line function which we have already declared and passing image and lines as the input

read the video
read the frame
press Q to quit the video.

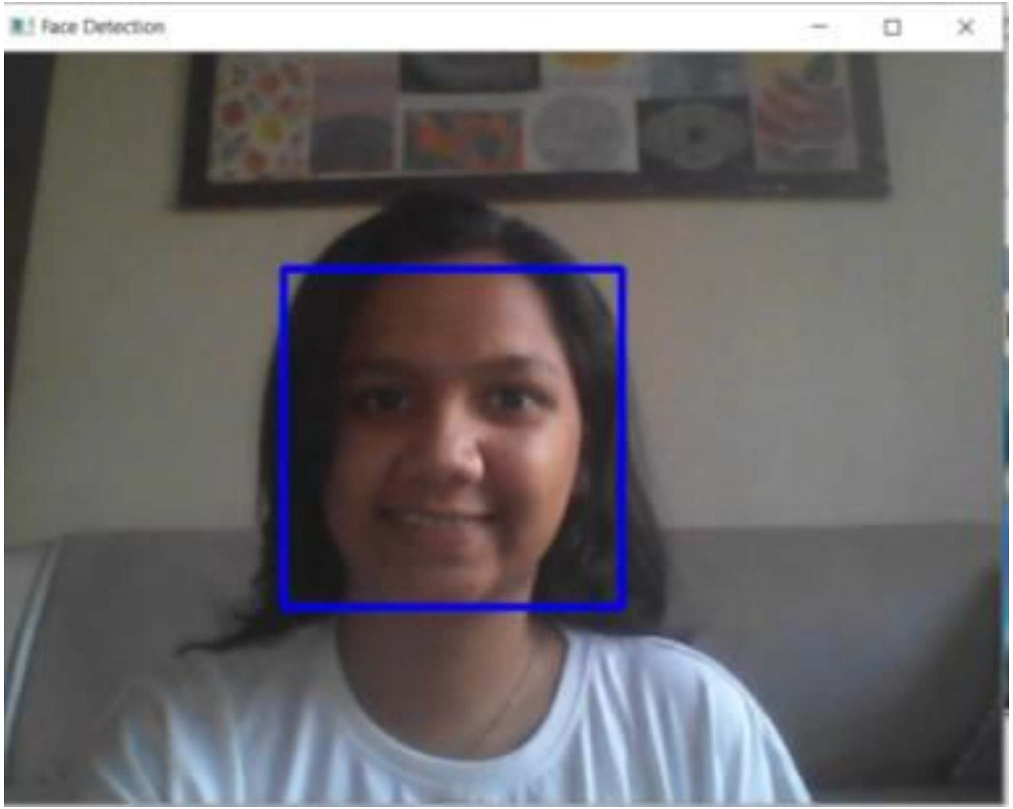
4. Results

4.1 UI of the project



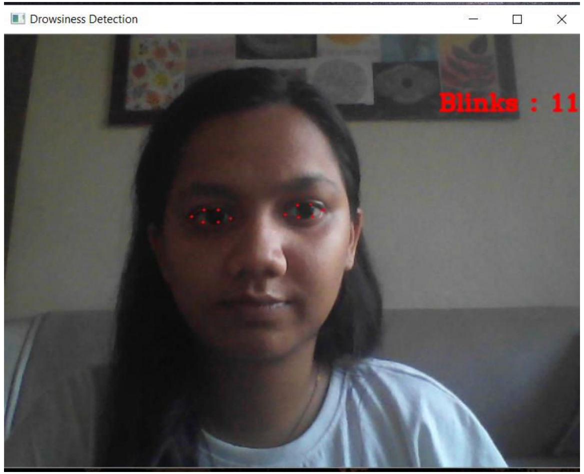
4. Results

4.2 Face Detection System



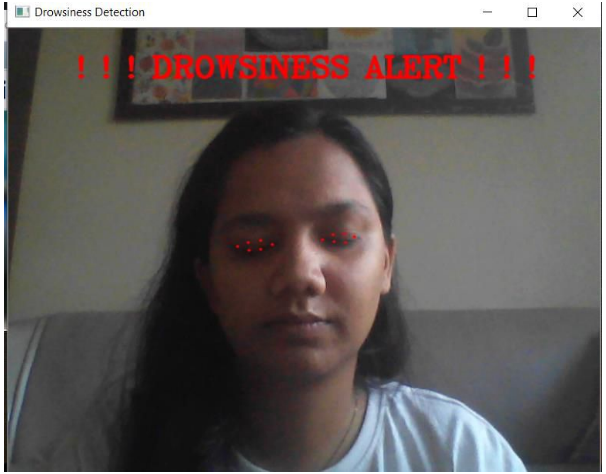
4. Results

4.3 Drowsiness Detection System



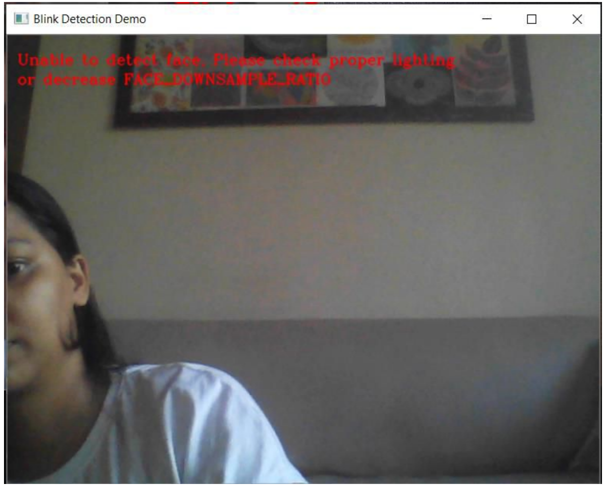
```
Blinks : 11
```

Fig 1. Blink Count



```
! ! ! DROWSINESS ALERT ! ! !
```

Fig 2. Drowsiness Detection

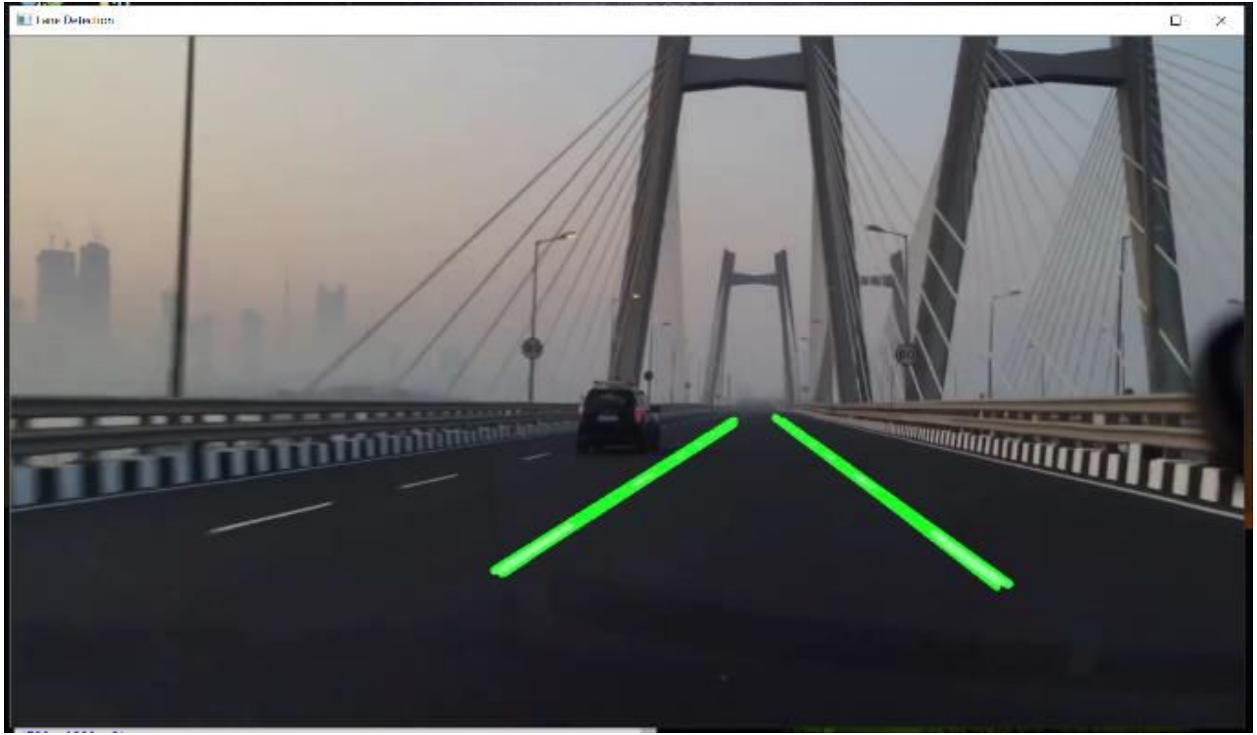


```
Unable to detect face. Please check proper lighting
or decrease FACE_DOWNSAMPLE_RATIO
```

Fig 3. Unable to detect Face

4. Results

4.3 Lane Detection System



5. Future Scope

- The model can be improved incrementally by using other parameters like blink rate, state of the car, etc.
- We can also stop the car engine if the driver doesn't turn off the alarm rings for too long.
- 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.
- Same drowsiness system can also be used in aircraft. The yawing and nodding of head can be implemented for detecting drowsiness for future work.
- Roads can be categorized into two groups on the basis of their marking:
  - Structured road,
  - Non structured road

  In this paper we have used lane detection for structured roads. Lane detection methods can further be extended to cover unstructured roads.
- Lane detection can be further improved for detecting sharp curves.
- We can add an automatic sensor which can detect that the car is taking a left or a right turn and give the indicator automatically or whenever the car crosses the lane marking without signalling, either a alarm signal will sound, or else, a brake will be applied to the wheels, to slow down the speed, just in order to avoid a possible accident.

6. References

[1] Yashika Katyal ,Suhas Alur,Shipra Dwivedi, “Safe Driving By Detecting Lane Discipline and Driver Drowsiness”, in 2014 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)

[2] V.V. Dixit, A.V. Deshpande, and D. Ganage, “Face Detection for Drivers’ Drowsiness Using Computer Vision” N.A. Abu Osman et al. (Eds.): BIOMED 2011, IFMBE Proceedings 35, pp. 308–311, 2011. www.springerlink.com

[3] Damian Sałapatek, Jacek Dybała, Paweł Czapski, Paweł Skalski, “Driver Drowsiness Detection Systems”, publication at: https://www.researchgate.net/publication/319464008

[4] Sunil Kumar Vishwakarma, Akash, Divakar Singh Yadav, “Analysis of Lane Detection Techniques using openCV”, IEEE INDICON 2015 1570168493

[5] Yue Dong, Jintao Xiong, Liangchao Li, Jianyu Yang, “Robust lane detection and tracking for lane departure warning”, Published in: 2012 International Conference on Computational Problem-Solving (ICCP)

[6] Yong Chen, Mingyi He, “Sharp Curve Lane Boundaries Projective Model and Detection”, 978-1-4673-0311-8/12/$31.00 ©2012 IEEE

[7] Belal ALSHAQAQI, Abdullah Salem BAQUHAIZEL, Mohamed El Amine OUIS, Meriem BOUMEHED, Abdelaziz OUAMRI, Mokhtar KECHE, “DRIVER DROWSINESS DETECTION SYSTEM”, 978-1-4673-5540-7/13/$31.00 ©2013 IEEE

[8] Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, Sheroz Khan, “Real Time Lane Detection for Autonomous Vehicles”, 978-1-4244-1692-9/08/$25.00 ©2008 IEEE

Thank You