Introduction

Driver drowsiness or fatigue is an important element in vehicle accidents. Latest statistics evaluate that every year 1550 deaths and 71,000 injuries can be ascribed to accidents due to the driver fatigue. Drowsiness is a state where a person sleep or almost likely. It alludes to a failure to keep awake or a drive to rest. It alludes to a powerlessness to keep wakeful or a drive to sleep. The advancement of technology for recognizing or forestalling drowsiness in the driver's seat is a noteworthy challenge in the field of accident evasion frameworks. It would in this manner be advantageous to figure out how to recognize drowsiness before it happens and to have the capacity to caution the driver in time. A few systems already had been created, in light of recording of head developments, movement of steering wheel, heart rate variability or grip quality. The algorithm of eye detection system integrated with hardware to develop the smart vehicles, which can implement nationwide to avoid the road accidents. Microcontroller and camera are used to make and intelligent hardware and software integrated system. By checking the driver's eyes, the indications of driver drowsiness can be identified in the beginning to protect from vehicle accident.[1]

Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Another important example is an embedded Google map on a website, which can be achieved using the Static maps API, Places API or Road API. Google launched the Google Maps API in June 2005 to allow developers to integrate Google Maps into their websites. It was a free service that didn't require an API key until June 2018. By using the Google Maps API, it is possible to embed Google Maps into an external website, on to which site-specific data can be overlaid. Although initially only a JavaScript API, the Maps API was expanded to include an API for Adobe Flash applications (but this has been deprecated), a service for retrieving static map images, and web services for performing geocoding, generating driving directions, and obtaining elevation profiles. Over 1,000,000 web sites use the Google Maps API, making it the most heavily used web application development API. In September 2011, Google announced it would deprecate the Google Maps API for Flash.

The Google Maps API is free for commercial use, provided that the site on which it is being used is publicly accessible and does not charge for access and is not generating more than 25,000 map accesses a day. Sites that do not meet these requirements can purchase the Google Maps API for Business. Different APIs have different functions like- the Roads API identifies the roads a vehicle was traveling along and provides additional metadata about those roads, such as speed limits, placeid and location.[2]

Problem Statement

The objective of this project is to generate an alarm which is based on the drowsiness state of the driver and the current speed of the vehicle by using machine learning algorithm and Google API.

Components Required

Hardware required

- A Smartphone/ Laptop
- Camera

Software required

- Real time tracking system like GPS.
- Pycharm Python editor
- Google Maps API – Roads API

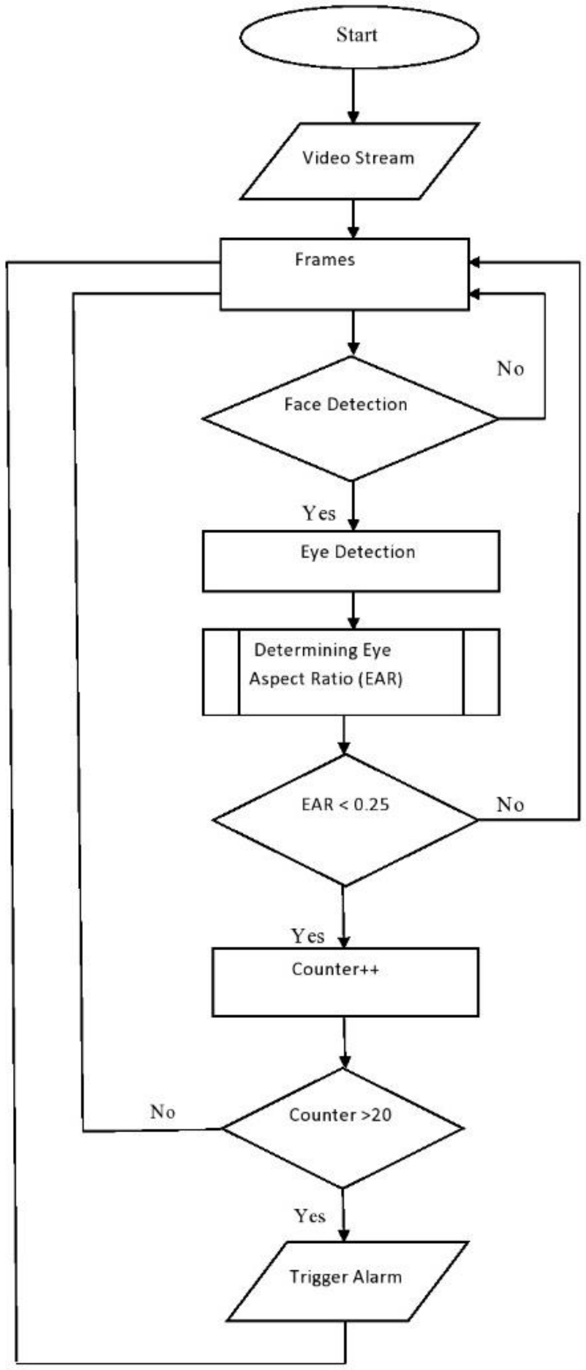
Methodology

To detect the drowsiness, we will use Haar Cascade algorithm. Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features.

The Haar cascade algorithm for eye detection contains four stages:

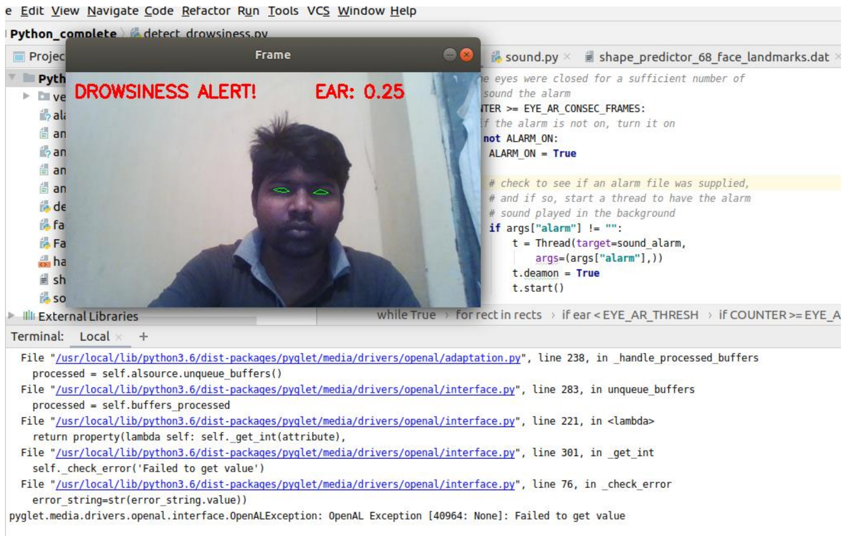
- Haar Feature Selection
- Creating Intergal Images
- Adaboost Training
- Cascading Classifiers[3]

- Steps to determine the drowsiness-



A color image is consist of three basic color RGB (Red, Green, Brown) and computer read the image either binary or gray so, for detecting features from the color image such as eye, nose, head gray image is required. First the webcam is used for acquired the image for processing. Then to search and detect the faces we use the Haarcascade file face.xml in each individual frame.

The main purpose of this paper is to utilize the reflection of retina as a way to discovering the eyes on the human face, and as a method utilizing the nonappearance of this reflection distinguishing whether the eyes are closed. It was observe that this technique may not be the good for system for checking the eyes due to the two reasons. First, in situation when lighting
is lower, the retinal reflection amount declines and next, the retinal reflection may not appear, if the individual has little eyes, as seen in Figure.[3]



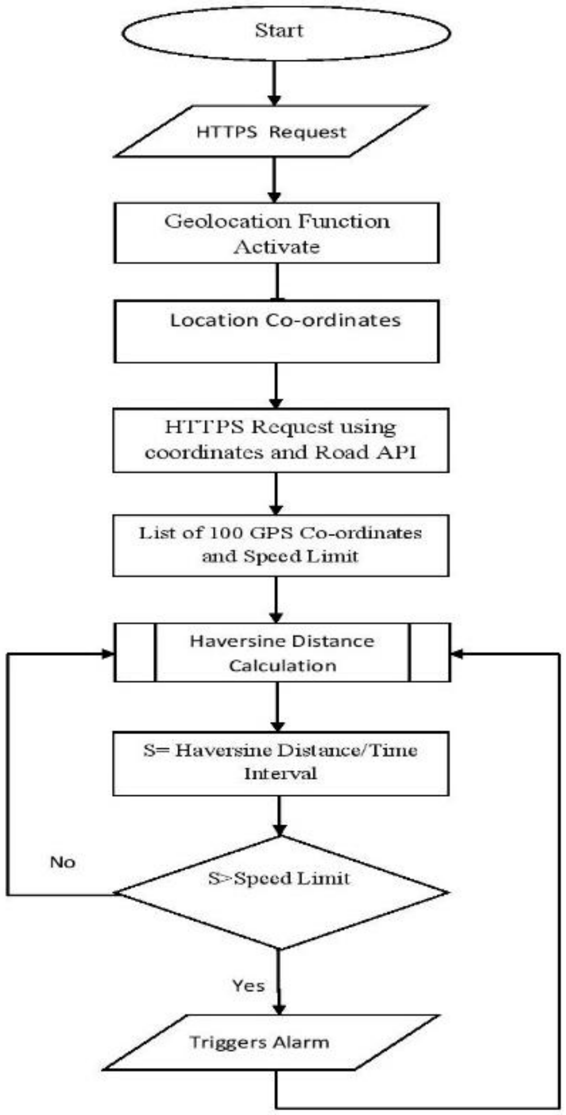
```
e eyes were closed for a sufficient number of
sound the alarm
TER >= EYE_AR_CONSEC_FRAMES:
f the alarm is not on, turn it on
not ALARM_ON:
 ALARM_ON = True

 # 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()
```

```
  File "/usr/local/lib/python3.6/dist-packages/pyglet/media/drivers/openal/adaptation.py", line 238, in _handle_processed_buffers
    processed = self.alsource.unqueue_buffers()
  File "/usr/local/lib/python3.6/dist-packages/pyglet/media/drivers/openal/interface.py", line 283, in unqueue_buffers
    processed = self.buffers_processed
  File "/usr/local/lib/python3.6/dist-packages/pyglet/media/drivers/openal/interface.py", line 221, in <lambda>
    return property(lambda self: self._get_int(attribute),
  File "/usr/local/lib/python3.6/dist-packages/pyglet/media/drivers/openal/interface.py", line 301, in _get_int
    self._check_error('Failed to get value')
  File "/usr/local/lib/python3.6/dist-packages/pyglet/media/drivers/openal/interface.py", line 76, in _check_error
    error_string=str(error_string.value))
pyglet.media.drivers.openal.interface.OpenALException: OpenAL Exception [40964: None]: Failed to get value
```

Figure1- Eye Tracing image

To detect the speed of vehicle Geolocation API of HTML is used. The HTML Geolocation API is used to get the geographical position of a user and the Roads API is used for obtaining a list of about 100 coordinates and speed limits on that road. Using these coordinates, we will be able to calculate the real time current speed of the vehicle in every t seconds of time interval. For calculating speed we will be using Haversine distance between the two locations on the earth as the earth has a curved surface.

Steps to detect and compare the speed are –


Start
HTTPS Request
Geolocation Function Activate
Location Co-ordinates
HTTPS Request using coordinates and Road API
List of 100 GPS Co-ordinates and Speed Limit
Haversine Distance Calculation
S= Haversine Distance/Time Interval
No
S>Speed Limit
Yes
Triggers Alarm

Geolocation refers to the identification of the geographic location of a user or computing device via a variety of data collection mechanisms. Typically, most geolocation services use network routing addresses or internal GPS devices to determine this location. Geolocation is a device-specific API. This means that browsers or devices must support geolocation in order to use it through web applications. A request via secure contexts such as HTTPS is sent to the geolocation method to get the user's position. The getCurrentPosition() method is used then to get the coordinates of location. The main steps included are -

- Check if Geolocation is supported

- If supported, run the getCurrentPosition() method. If not, display a message to the user
- If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)
- The showPosition() function outputs the Latitude and Longitude

```
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>[2]
```

The Roads API allows you to map GPS coordinates to the geometry of the road, and to determine the speed limit along those road segments. The API is available via a simple HTTPS interface, and exposes the following services:

- **Snap to roads** This service returns the best-fit road geometry for a given set of GPS coordinates. This service takes up to 100 GPS points collected along a route, and returns a similar set of data with the points snapped to the most likely roads the vehicle was traveling along. Optionally, you can request that the points be interpolated, resulting in a path that smoothly follows the geometry of the road.
- **Nearest roads** This service returns individual road segments for a given set of GPS coordinates. This services takes up to 100 GPS points and returns the closest road segment for each point. The points passed do not need to be part of a continuous path.

- **Speed limits** This service returns the posted speed limit for a road segment. The Speed Limit service is available to all customers with an Asset Tracking license. For Google Maps Platform Premium Plan customers who transitioned to pay-as-you-go pricing, the feature remains active.[4]

A request for speed limits must be sent via HTTPS, and takes the following form:

https://roads.googleapis.com/v1/speedLimits?parameters&key=YOUR_API_KEY?

Required parameters for this request:

- Enter a path parameter.
- Your application's API key.
- Optional parameter can be units of speed.

Responses:

- speedLimit — An array of road metadata. Each element consists of the following fields:
  - placeId
  - speedLimit
  - units

- snappedpoint — an array of snapped points. This array is present only if the request contained a path parameter. Each point consists of the following fields:
  - location
  - originalIndex
  - placeId

- warning_message — A string containing a user-visible warning.[3]

The following elements may be present in a speedLimit response-

{

speedLimits:

[
  {
    placeId: "ChIJX12duJAwGQ0Ra0d4Oi4jOGE",
    speedLimit: 105,
    units: "KPH"
  }
],
snappedPoints:
[
  {
    location:
    {
      latitude: 38.75807927603043,
      longitude: -9.037417546438084
    },
    originalIndex: 0,
    placeId: "ChIJX12duJAwGQ0Ra0d4Oi4jOGE"
  },
],
warningMessage: "Input path is too sparse. You should provide a path where consecutive points are closer to each other. Refer to the 'path' parameter in Google Roads API documentation."
}[4]

Packages and Libraries used

- from scipy.spatial import distance as dist
- from imutils.video import VideoStream
- from haversine import haversine, Unit
- 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[5]

Functions used in python script

Some functions that are being used in the python script of this system are-

- **sound_alarm()-** This function accepts a path to an audio file that will be the alarm tone which is stored or available on the disk. This function includes methods of pyglet package for implementation of sound.

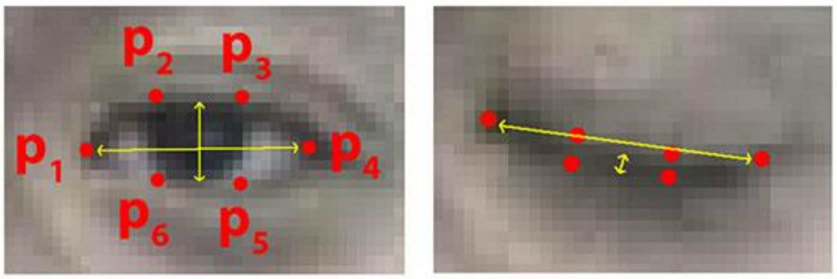
  def sound_alarm(path):

  music = pyglet.resource.media(&#39;alarm.wav&#39;)

  music.play()

  pyglet.app.run()

- **eye_aspect_ratio()-** This function is used to compute the ratio of distances between the vertical and horizontal eye landmarks. To calculate euclidean distance between different eye coordinates we used euclidean() method of scipy package.



  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])
    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)
    return ear
```

We defined two constants, first constant, *EYE_AR_THRESH*, for the eye aspect ratio to indicate eye blink and then a second constant, *EYE_AR_CONSEC_FRAMES,* is defined for the number of consecutive frames which has a certain threshold value. If the eye aspect ratio falls below the first constant value then the counter will be increased to count the number of frames when the eye was closed. If the counter value exceeds the second constant the alarm would be set up and we will start checking the symptoms of drowsiness in the person.

```
EYE_AR_THRESH = 0.25
EYE_AR_CONSEC_FRAMES = 20
COUNTER = 0
ALARM_ON = False
```

We created a separate thread which will be responsible for calling sound alarm to ensure the efficient working of our main program until the drowsiness alert sound finishes playing. If the eyes are open, we reset the counter value and ensure that the alarm is off.

```
if args["alarm"] != "":
t = Thread(target=sound_alarm,
args=(args["alarm"],))
t.deamon = True
t.start()
```

Scope of the Project

The scope of this project is to determine the eye behaviour and facial expressions of the driver to detect if the driver is drowsy or not. Also, this system in future will be designed to detect the speed of the vehicle and compare this speed with the maximum speed limit based on the position of the vehicle. This system is designed to generate an alarm when either of the above cases are fulfilled. This can result in the decrement of the number of accidents happening in our day to day life and thus improve in road safety.

This alarm system will result in the decrement in the number of incidents and crashes on the roads which will result in the improved road safety. This system is highly economic and easy to install saving space, money and most importantly human lives. Also, the system is very user friendly and easy to operate.

This system is very useful for the drivers who have to drive for long durations and without taking much breaks. In the future, this system may be expanded to trains and buses all over the world as these are the transports used for long journeys and for longer durations.

References

- http://ieeexplore.ieee.org.sci-hub.tw/xpl/artic%C2%ADleDetails.jsp?arnumber=7493990\ by Javed Ahmed, Jian-Ping Li, Saeed Ahmed Khan, Riaz Ahmed Shaikh
- https://en.wikipedia.org/wiki/Google_Maps#Google_Maps_API
- http://www.willberger.org/cascade-haar-explained
- https://developers.google.com/maps/documentation/roads/speed-limits
- https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/