**CHAPTER 1**

**INTRODUCTION**

Vehicles have been increasing in the world rapidly in recent times, and so is the number of accidents happening. Every year the lives of more than 1.25 million people are reduced as a result of a road traffic crash. Between 20 and 50 million people suffer injuries, with many incurring a disability as a result of their injury, according to the World Health Organization(WHO). A major portion of them occurs during night driving.

We know that young generation prefers bikes and motorcycle over four wheelers. Moreover speeding and drunk driving have become common issues.Due to lack of our experience or focus and violation of traffic rules, result in several accidents. So with the help of technology problems mentioned above are avoided and their effects are minimized. The idea of developing this project t comes from our social responsibility towards society.

Almost all vehicle are captured in CCTV cameras. So it is not easy to detect and recognize license plate correctly. To overcome this problem, we propose an algorithm that automatically recognizes license plate using a CCTV camera footages. A license plate detection and recognition is one of important processes in investigating a car accident.The new license plate format is made up of ## (letter) #### where # is a number to be detected. Automatic Number Plate Recognition (ANPR) is a powerful automated system which can have immense applications in our today’s modern world where number of vehicles are increasing exponentially day by day and to control such immense traffic effectively this system can be of utmost importance. ANPR can perform vehicle surveillance by scrutinizing vehicle theft, automation at toll booths, parking management and many others. This paper will explore and elaborate the proposed algorithm for ANPR .Automated license plate recognition has many applications. It can facilitate access to secure premises and improve security by detecting unauthorized vehicles and alerting security personnel. Police cars fitted with a system for the detection and recognition of license plate can pull relevant history about a moving vehicle (e.g, repeated speed violations, expired registration, or the like). The input to the system is a digital image, taken by the high speed rotor cameras or digital cameras in our case, of a car and converted to gray scale using NTSC standard . The illumination condition which is a main bottleneck is improved first by enhancing the image by finding the variance, and performs contrast stretching or equalization of histogram. The localization of license plates using edge finding algorithms is another approach used mainly in object detection in edge maps of images. The edge based methods pertaining to plate localization are based on the principle that the plates appearing in the image of the vehicle have some distinct features that make it distinguishable easily from the rest of the image such as high contrast of license plate as compared to the rest of the image of the vehicle .

Driver Drowsiness Detection is one of the car safety features that helps prevent accidents caused by the drowsy driver. A drowsiness detection system is particularly useful for night driving, to avoid accidents due to falling asleep. It also helps in determining that the driver is not in an active state which could be a Drowsiness . It activates warning alerts to wake up the driver to prevent a possible accident and also alerts . The development of technologies for preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Sleep and drowsiness detection systems can be used to monitor the driver and alert the person to wake him up to avoid the possibility of collisions.

According to the Central Road Research Institute (CRRI) in Indian says that 40% of highway accidents occur due to drivers dozing off.

An accident is an unexpected, unusual, unintended external action which occurs in particular time and place. Carelessness of the driver is the major factor for accident. The government has made rules that rider should compulsory wear the helmet and not consume alcohol and drive. Still the riders do not obey the rules. These accidents are caused due to negligence of the rider. Not wearing the helmet causes the rider with head injuries which may lead to death of the rider. In order to overcome this an intelligent system, smart helmet is proposed, it detects the helmet and also the alcohol present in rider’s breath. This system has a pair of transmitter and receiver, the transmitter is placed in the helmet and the receiver is placed at the bike ignition. There are different sensors to ensure the helmet is on the head. These vibration sensors are placed in helmet where the probability of hitting is more. An alcohol sensor is placed near mouth of the rider. The alcohol sensor detects the presence of alcohol in rider’s breath. The data of the detection of helmet and alcohol is coded with RF encoder and then transmitted through radio frequency transmitter. The receiver at the bike receives the data and the data is decoded using RF decoder. The result of presence of helmet and the alcohol detection is analyzed on the smart phone. The proposed system will be so designed that if one of the two conditions are violated then also the bike won’t start. The bike will start only if the both conditions are followed. This smart helmet will help the rider to compulsory wear helmet and restrict drink and drive condition. MCU controls the function of relay and the ignition, it control the engine through a relay and a relay interfacing circuit.

However, the main goal of our project is to make it mandatory for the rider to wear a helmet during the ride meanwhile providing solutions to other major issues for accidents.

**CHAPTER 2**

# **2.Related works**

## **2.1 Drowsiness Driver Detection**

By using a non-intrusive machine vision-based concepts, drowsiness of the driver detected system is developed. Many existing systems require a camera which is installed in front of driver. .[2] It points straight towards the driver’s face and monitors the driver’s eyes to identify the drowsiness. For large vehicle such as heavy trucks and buses this arrangement is not apropos. Bus has a large front glass window to have a broad view for safe driving. If we place a camera on the window of front glass, the camera blocks the frontal view of driver so it is not practical. If the camera is placed on the frame which is just about the window, then the camera is unable to detain the anterior view of the face of the driver correctly. [7]The open CV detector detects only 40% of face of driver in normal driving position in video recording of 10 minutes. In the cater-cornered view, the Open CV eye detector (CV-ED) frequently fails to trace the pair of eyes. After five successive frames if the eye found to be closed the system finalise that the driver is declining slumbering and issues a warning signal. Hence existing system is not applicable for large vehicles. In order to conquer the problem of existing system, new detection system is developed in this project work.

## 2.2 **Automatic Number plate Detection**

ANPR System utilizing OCR at the center point of the framework is the OCR (Optical Character Recognition framework) which is utilized to extricate the alphanumeric characters present on the number plate. . [1] There are just two segments in the framework, the web cameras at the front-end and the remote PCs at the back-end to process the information. The remote PCs pre-process the perform activities like OCR on the put away pictures sent by the cameras at the path level A case of a server ranch can be the London Congestion Charge venture. [9] The remote PCs can be connected with the database which stores the subtleties of the vehicle proprietors and hence the necessary data can be gotten. Utilizing this data the outlaw can be gotten.

The current framework utilizing OCR was found to have the accompanying disadvantages:

1. Misidentification:

2. Dim pictures:

3. Defects in rakish recognition

## **Smart Helmet Bike Starter**

The task has a wired correspondence and it is associated with a Microcontroller. This uses sensors to recognize a head protector or liquor discovery and the correspondence frameworks is utilized to consequently kill the start. The other existing framework has the speed cutoff wherein the biker is going in.[3]The cap has been fixed with speed sensor and as needs be educate the rider to decrease or speed up dependent on the snags before the bicycle. First we need to guarantee that climate rider is wearing protective cap or not.

This has following burdens:

1. Rider doesn't wear head protector in districts where traffic checking isn't finished.
2. It is conceivable to test liquor content present in blood in every individual rider in large nations like India is unimaginable.
3. Difficulty of usage of traffic administers by traffic police.

CHAPTER 3

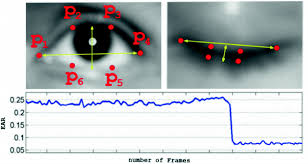
**3. PROBLEM DESCRIPTION**

**3.2 PROPOSED SYSTEM**

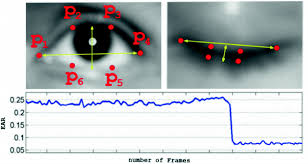
## **3.2.1Drowsiness Driver Detection-Euclidean**

Squint discovery can be assessed by estimating EAR (Eye viewpoint Ratio) utilizing OPENCV capacities and DLIB's pre prepared Neural system based expectation and finder function.In Figure-1 it shows EAR can be estimated from eye organizes came back from OPENCV utilizing EAR recipe given underneath. Unexpected plunge in EAR esteem against a set limit can be utilized for flicker recognition and miniaturized scale rest discovery appeared in Figure-2.

**Figure 1:** The eye aspect ratio equation.



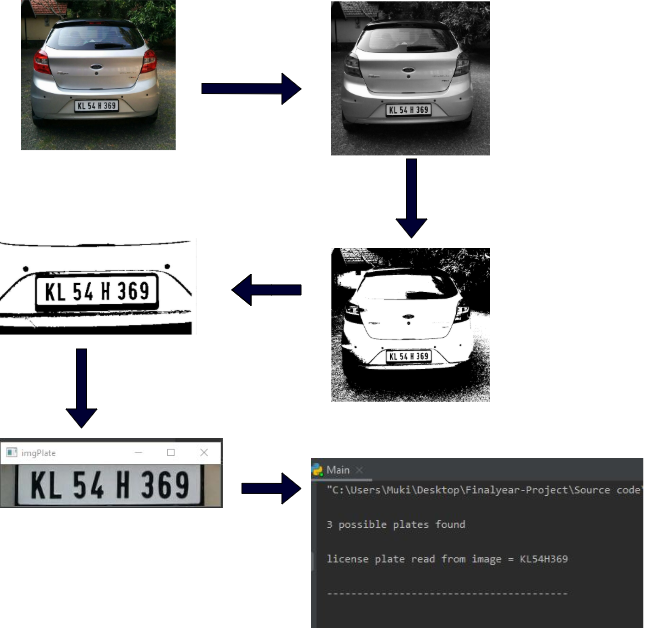
**Figure 2: Results of facial Landmark detection and identification of eye coordinates.**



**Figure 3: Results of eye blinking detection**

## **3.2.2Automatic Number plate Detection**

To recognize the number plate in the vehicles we use K-closest neighbor (KNN) classifiers to distinguish these isolated characters.[5] The calculation K-closest neighbor (KNN) measures KNN is more comparative than PNN (Probabilistic Neural Network) calculation and its acknowledgment rate is up to 96.51 % by and large. [11] The acknowledgment rate on normal is about 95.87 % for the PNN classifier and about 96.51 % for the KNN classifier. The most noteworthy acknowledgment rate for all contentions and square sorts for PNN are 97.14 %, the most noteworthy acknowledgment rate for all case and square sorts for KNN are 100 %.[8] The most noteworthy acknowledgment rate for square kind is square 5x5, and the second is 10x5, regardless of which classifier. Their acknowledgment rates are 96.97 % (PNN) and 99.77(KNN), separately.



**Figure 4: Steps for detecting number plate**

**CHAPTER 4**

**4.1 SYSTEM SPECIFICATION:**

**4.4.1 Software Requirements**

* The web cam of the laptops
* HP laptop (Elite book 840 G1)
* CPU-Core-I5, 2.4 GH
* RAM-8.0 GB
* Graphic card: GeForce GT 230M
* 64-bit windows OS.

**4.4.2 Hard ware Requirments**

* ESP8266
* Alcohol detector
* Pressure sensor
* Circuit board
* Node MCU
* LED Bubls
* Resistors
* Capacitors
* 2.5V battery
* Buzzer
* Helmet
* LED Display
* Jumper wires

**CHAPTER 5**

**MODULES**

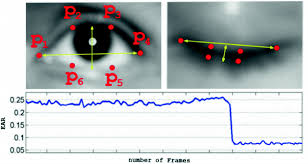
**5.1 DROWSINESS DETECTION:**

The development of technologies for preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Sleep and drowsiness detection systems can be used to monitor the driver and alert the person to wake him up to avoid the possibility of collisions. There are different methods for this purpose. the triggering of sleep is detectable , since it is preceded by a drop down in temperature of the brain, which can be measured within the auditory canal. Temperature of the auditory canal is hence measured and its value is used to detect drowsiness and inturn activate a visible or audible alarm to prevent the driver from falling asleep. Other methods like monitoring rate of blinking the eyes, pattern of driver activity and heart rate can be used for the purpose. Monitoring the rate of blinking of the eye, and head positions are reliable and are achieved by image processing.

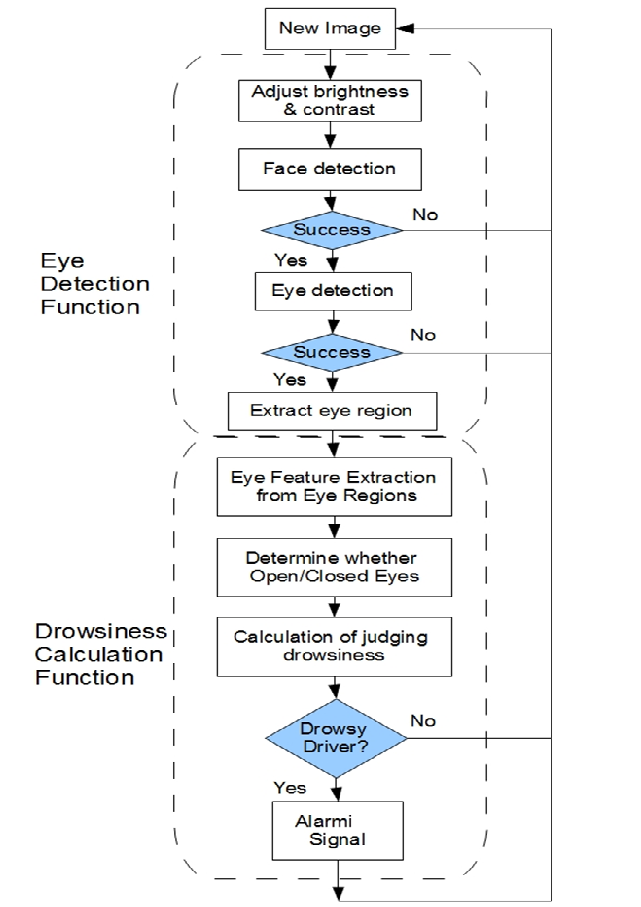
On positive detection of drowsiness ,the system activated noise or vibrating alert to wake the driver .It increases in intensity as it try to wake up the driver .This can be switched off like an alarm.

The Euclidean algorthim can be used to find the position of the eye by using

the six points of the eye.



**5.1.1 FLOW GRAPH**

****

**5.1.2 ACCURENCY**

• Detection disappointment it displays the disappointment pace of the framework

• Warning pace of the framework is given by as follows

**Figure 5: correct rate**

Figure 5 portrays the determined estimations of the right rate for each tried example the right admonition pace of tiredness identification. Here we can accomplish higher than 99.2% and the normal right rate can accomplish 99.45%.

Figure 6 portrayed the determined proportion of exactness for each occasion:

**Figure 6: Precision rate**

The outcome portrays the exactness of the proposed arrangement of learning. The outcome differs with the accompanying viewpoints:

• No. of caught outlines

• Size of the eye

• Eye leeway (with or without eyeglass)

With expansion, the preparation dataset assume the most significant job in indicating the exhibition of the system.

**Figure 5: Correct rate for each experiment instance**

**Figure 6: Precision rate for each experiment instance**

**5.2 AUTOMATIC NUMBER PLATE DETECTION**

We gathered 30 pictures of Indian vehicles having tags in various light conditions. We partitioned those pictures in to 3 of the gatherings and every one of the gathering has 10 pictures. We have named those gatherings A, B and C respectively. We tried our own experiments on these gatherings which are additionally talked about in following segments.

**5.2.1 KNN ALGORTHIM**

It uses  KNN algorithm  to recognize the alphabets and digits of normalized font. The program takes an image of the license plate as input, Converts it to gray-scale giving a threshold of the image and finally extracts the plate area.

**5.2.2 License Plate Detection**

As we experience the beforehand in Architecture and Implementation sections, we have actualized two calculations for tag identification: Haar-Training and KNN. We tried our own experiments on these techniques independently to analyse their presentation

**License Plate Detection through Haar-Training**

|  |  |  |  |
| --- | --- | --- | --- |
| **Groups Plates** | **Total License Plates** | **No. of Detected License** | **Detection Rate (%)** |
| **Group A** | 10 | 8 | 0.8 |
| **Group B** | 10 | 9 | 0.9 |
| **Group C** | 10 | 9 | 0.9 |
| **Results** | 30 | 26 | 0.86 |

**Table 1: License Plate Detection through Haar-Training**

**Figure 8: License Plate Detection through Haar-Training**

|  |  |  |  |
| --- | --- | --- | --- |
| **Groups Plates** | **Total License Plates** | **No. of Detected License** | **Detection Rate (%)** |
| **Group A** | 10 | 9 | 0.9 |
| **Group B** | 10 | 10 | 1 |
| **Group C** | 10 | 10 | 1 |
| **Results** | 30 | 29 | 0.96 |

**Table 2: License Plate Detection through KNN-Training**

**Figure 9: License Plate Detection through KNN**

Table 1 shows the consequences of tag identification through Haar-training.We can examination through the table that Group A has less discovery when contrasted with different gatherings yet recognition pace of haar-preparing for 3 gatherings is (0.96).

**License Plate Detection Results through KNN**

Results that we traversed KNN calculation we showed in the Table 2. Gathering A has lower recognition rate when contrasted with different gatherings however identification pace of KNN calculation against 3 gatherings is 0.86 which is very agreeable.

**Precision of Tesseract-OCR**

We utilized the KNN calculation and tesseract-ocr library with the blend for LPR..The second segment contains those tags that were extricated in past segment utilizing KNN calculation.

**Total Numbers of Extracted License Plates**=26

**Correct OCR Results**=22

**Precision of OCR**= (22/26)\*100= 84.6%

We can realize that the accuracy of tesseract-ocr results is just about 85 percent, which is very worthy. All in all, we see from the above outcomes that haar preparing calculation has higher discovery rate when contrasted with KNN calculation however KNN calculation likewise indicated great outcomes.

Our experiments results for KNN likewise have given us that it has 85% precision for character acknowledgment. Since we will effectively compute the exactness of our License Plate Recognition framework. .

**Correct OCR Results**=22

**No. of License Plate used in test cases**=30

**Precision of LRP System**= (22/30)\*100= 73%

The calculation shows that the precision of our License Plate Recognition system is 73%.

**License Plate Recognition in Practice, Real Time**

We have picked 20 autos from rurak territory having standard Foreign tags and test our tag acknowledgment application.[11] We have tried to find that what number of attempts it expected to perceive tag. We have given greatest 3 attempts to perceive tag.

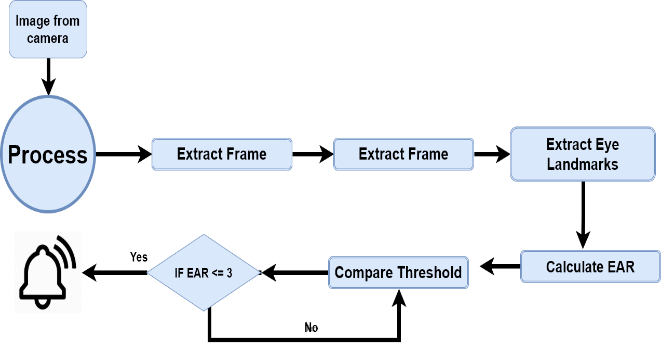
**Total No. of Cars**=20

**Detected License Plates**=17

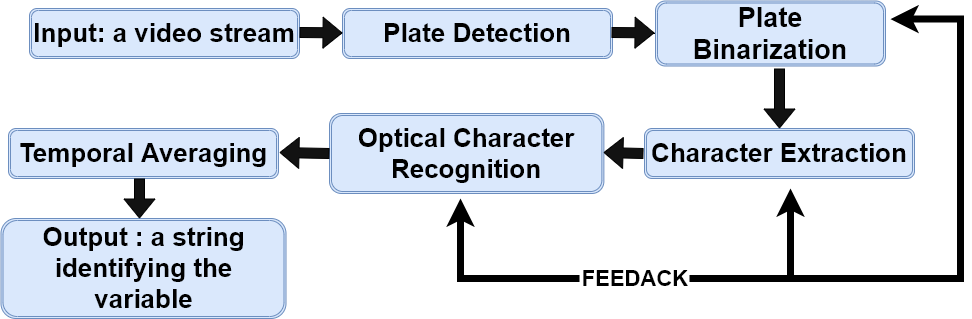
**Precision of LPR in Real Time**= (17/20)\*100=85%

The calculation shows that the precision of our LPR system in real time is 85%.

**5.2.3 ACCURANCY**



**5.2.4 FLOW GRAPH**

****

**5.2.5 SIMPLE FLOW CHART**



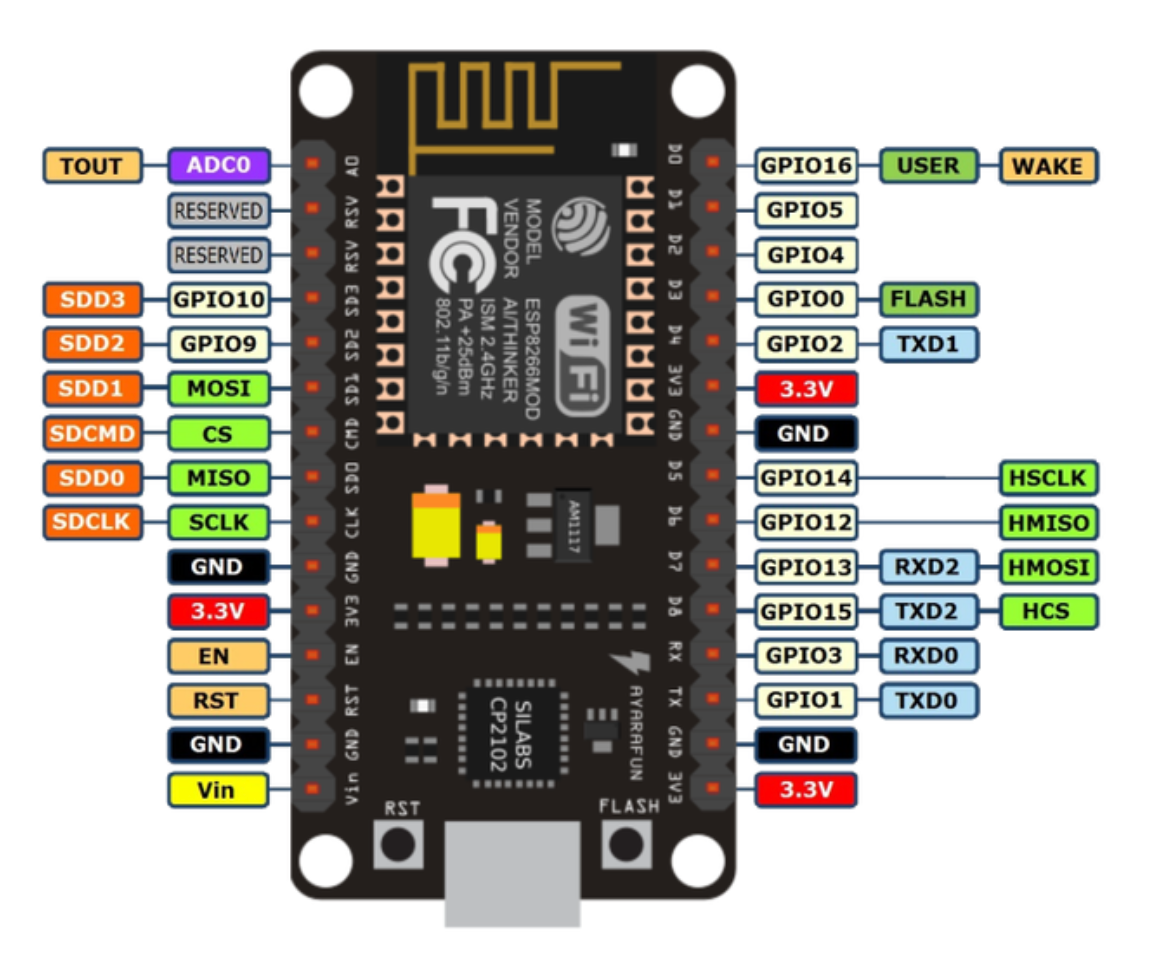
## **SMART HELMET BIKE STARTER AND ALCOHOL DETECTION**

According to study result, most instances of mishaps are cause by engine bicycles because of alcoholic and driving. It's mostly because of the nonappearance of head protector and the alcoholic and drive.In our proposed framework we have an answer for build up an electronic brilliant cap framework that effectively checks the wearing of cap and tanked driving.By executing this framework progressively for riders a sheltered excursion was guaranteed and a few passing because of the bike mishaps can be maintained a strategic distance from .We have an answer for present propelled sensors procedures and radio recurrence remote correspondences are remembered for this undertaking to make it a decent one. This task guarantees whether the individual wearing head protector and maintaining a strategic distance from the tanked driving. By effectively executing this undertaking continuously a protected bike venture is conceivable and it lessen the wounds that were abstained from during the mishaps.

An accident is an unexpected, unusual, unintended external action which occurs in particular time and place. Carelessness of the driver is the major factor for accident. The government has made rules that rider should compulsory wear the helmet and not consume alcohol and drive. Still the riders do not obey the rules. These accidents are caused due to negligence of the rider. Not wearing the helmet causes the rider with head injuries which may lead to death of the rider. In order to overcome this an intelligent system, smart helmet is proposed, it detects the helmet and also the alcohol present in rider’s breath. This system has a pair of transmitter and receiver, the transmitter is placed in the helmet and the receiver is placed at the bike ignition. There are different sensors to ensure the helmet is on the head. These vibration sensors are placed in helmet where the probability of hitting is more. An alcohol sensor is placed near mouth of the rider. The alcohol sensor detects the presence of alcohol in rider’s breath. The data of the detection of helmet and alcohol is coded with RF encoder and then transmitted through radio frequency transmitter. The receiver at the bike receives the data and the data is decoded using RF decoder. The result of presence of helmet and the alcohol detection is analyzed on the smart phone. The proposed system will be so designed that if one of the two conditions are violated then also the bike won’t start. The bike will start only if the both conditions are followed. This smart helmet will help the rider to compulsory wear helmet and restrict drink and drive condition. MCU controls the function of relay and the ignition, it control the engine through a relay and a relay interfacing circuit.

* + 1. **IC’S AND SENSORS**
* **NODE MCU**

**NodeMCU** is an open-source firmware and development kit that helps you to prototype or build IoT products. It includes firmware that runs on the **ESP8266** Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. ... It is based on the eLua project and built on the Espressif Non-OS SDK for **ESP8266**

****

**Figure: ESP8266**

* **Alcohol Sensor**

Alcohol sensors are used for checking the concentration of alcohol in a sample. Now a days they are mainly used by traffic police. To check the alcohol concentration users have to blow in the straw for some minutes. The air blown contains the vapours which are used for calculating the alcohol content.  
This happens because alcohol doesnt get digested instead it is absorbed by the mouth and stomach.Thus some traces are always left even after several hours of drinking.

****

**Figure:Alcohol sensor**

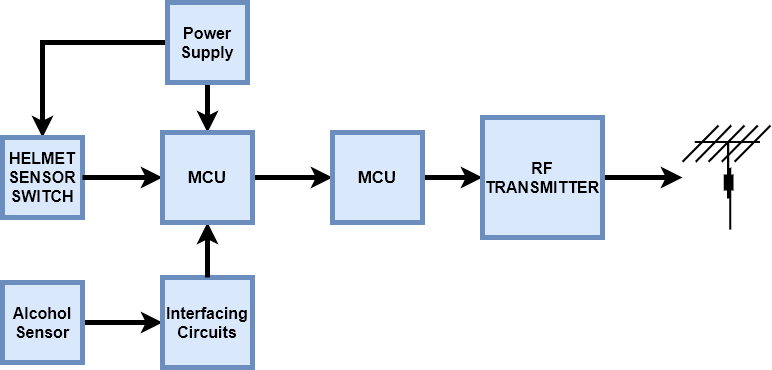
* **Force sensing resistor**

**F**SRs are sensors that allow you to detect physical pressure, squeezing and weight.



**Figure: Force Sensing Resistor**

**5.3.2 FLOW CHART**

****

**Bike Section**

****

**CHAPTER 6**

**CONCLUSION**

**6.1 CONCLUSION**

The proposed framework is to deliver an answer for one of the significant reasons for the street mishap, the driver laziness; the proposed arrangement does following the driver's face and eyes and afterward the framework will tell him when his eyes get shut so as to abstain from losing the control of the vehicle and causing auto collisions.

Continuous number plate location and acknowledgment framework that permits to "read" permit place data in a computerized way and perceive tag data with an exactness of over 70%, basically in a flash by essentially pointing and recognizing the gadget at a car. Our first and principle objective is to build up a LPR framework that ought to have accuracy over 70%. We have tried rightness of our framework against pictures put away in database and constant . The outcomes from contextual analysis part show that exactness of LPR framework utilizing pictures from the continuous database is 73 % which is very agreeable. The constant testing shows that exactness of our LPR framework is 85%. the outcomes shows that haar-preparing has better location rate (96%) when contrasted with KNN calculation (83%) however the measurements of KNN calculation In Table 2 likewise show that discovery pace of this calculation isn't terrible. Other goal was to utilize standard libraries, so we utilized tesseract-ocr for Optical Character Recognition and the outcomes from tests in Table 1 show that it has 86% accuracy. By actualizing this the demise rate can be diminished in our country.The head protector may not be a 100% lifeline yet at the same time it can the primary line of barrier for the rider in the event of a mishap to get shields the riders. .

The created venture proficiently guarantees:

• Ensuring rider is wearing protective cap all through the ride.

• Rider ought not be affected by liquor.

• Accident location.

**6.2 FUTURE WORKS**

Currently our LPR framework is utilizing KNN calculation for tag extraction. We can utilize KNN-preparing calculation rather than this calculation as results from contextual investigation shows that it has better discovery rate. We are utilizing nearby framework database for information stockpiling. For testing execution of remote information base server can be utilized through web administrations.

• It can be utilized in the genuine world by making it into the littler size.

• It can be utilized in four wheeler to guarantee the security of the driver.

**CHAPTER 7**

**APPENDIX**

**7.1 SOURCE CODE**

**7.1.1 DROWSINESS DETECTION**

'''This script detects if a person is drowsy or not,using dlib and eye

aspect ratio calculations. Uses webcam video feed as input.'''

|  |
| --- |
|  |
|  | calculations. Uses webcam video feed as input.''' |
|  |  |
|  | #Import necessary libraries |
|  | from scipy.spatial import distance |
|  | from imutils import face\_utils |
|  | import numpy as np |
|  | import pygame #For playing sound |
|  | import time |
|  | import dlib |
|  | import cv2 |
|  |  |
|  | #Initialize Pygame and load music |
|  | pygame.mixer.init() |
|  | pygame.mixer.music.load('audio/alert.wav') |
|  |  |
|  | #Minimum threshold of eye aspect ratio below which alarm is triggerd |
|  | EYE\_ASPECT\_RATIO\_THRESHOLD = 0.3 |
|  |  |
|  | #Minimum consecutive frames for which eye ratio is below threshold for alarm to be triggered |
|  | EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES = 25 |
|  |  |
|  | #COunts no. of consecutuve frames below threshold value |
|  | COUNTER = 0 |
|  |  |
|  | #Load face cascade which will be used to draw a rectangle around detected faces. |
|  | face\_cascade = cv2.CascadeClassifier("haarcascades/haarcascade\_frontalface\_default.xml") |
|  |  |
|  | #This function calculates and return eye aspect ratio |
|  | def eye\_aspect\_ratio(eye): |
|  | A = distance.euclidean(eye[1], eye[5]) |
|  | B = distance.euclidean(eye[2], eye[4]) |
|  | C = distance.euclidean(eye[0], eye[3]) |
|  |  |
|  | ear = (A+B) / (2\*C) |
|  | return ear |
|  |  |
|  | #Load face detector and predictor, uses dlib shape predictor file |
|  | detector = dlib.get\_frontal\_face\_detector() |
|  | predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat') |
|  |  |
|  | #Extract indexes of f acial landmarks for the left and right eye |
|  | (lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS['left\_eye'] |
|  | (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS['right\_eye'] |
|  |  |
|  | #Start webcam video capture |
|  | video\_capture = cv2.VideoCapture(0) |
|  |  |
|  | #Give some time for camera to initialize(not required) |
|  | time.sleep(2) |
|  |  |
|  | while(True): |
|  | #Read each frame and flip it, and convert to grayscale |
|  | ret, frame = video\_capture.read() |
|  | frame = cv2.flip(frame,1) |
|  | gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) |
|  |  |
|  | #Detect facial points through detector function |
|  | faces = detector(gray, 0) |
|  |  |
|  | #Detect faces through haarcascade\_frontalface\_default.xml |
|  | face\_rectangle = face\_cascade.detectMultiScale(gray, 1.3, 5) |
|  |  |
|  | #Draw rectangle around each face detected |
|  | for (x,y,w,h) in face\_rectangle: |
|  | cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2) |
|  |  |
|  | #Detect facial points |
|  | for face in faces: |
|  |  |
|  | shape = predictor(gray, face) |
|  | shape = face\_utils.shape\_to\_np(shape) |
|  |  |
|  | #Get array of coordinates of leftEye and rightEye |
|  | leftEye = shape[lStart:lEnd] |
|  | rightEye = shape[rStart:rEnd] |
|  |  |
|  | #Calculate aspect ratio of both eyes |
|  | leftEyeAspectRatio = eye\_aspect\_ratio(leftEye) |
|  | rightEyeAspectRatio = eye\_aspect\_ratio(rightEye) |
|  |  |
|  | eyeAspectRatio = (leftEyeAspectRatio + rightEyeAspectRatio) / 2 |
|  |  |
|  | #Use hull to remove convex contour discrepencies and draw eye shape around 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) |
|  |  |
|  | #Detect if eye aspect ratio is less than threshold |
|  | if(eyeAspectRatio < EYE\_ASPECT\_RATIO\_THRESHOLD): |
|  | COUNTER += 1 |
|  | #If no. of frames is greater than threshold frames, |
|  | if COUNTER >= EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES: |
|  | pygame.mixer.music.play(-1) |
|  | cv2.putText(frame, "You are Drowsy!!!", (150,200), cv2.FONT\_HERSHEY\_SIMPLEX, 1.5, (0,0,255), 2) |
|  | else: |
|  | pygame.mixer.music.stop() |
|  | COUNTER = 0 |
|  |  |
|  | #Show video feed |
|  | cv2.imshow('Video', frame) |
|  | if(cv2.waitKey(1) & 0xFF == ord('q')): |
|  | break |
|  |  |
|  | #Finally when video capture is over, release the video capture and destroyAllWindows |
|  | video\_capture.release() |
|  | cv2.destroyAllWindows() |

'''This script uses OpenCV'shaarcascade (face and eye cascade) to detect face

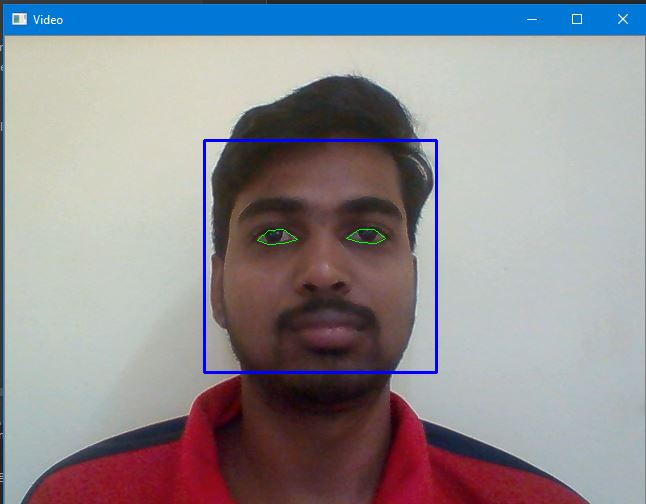
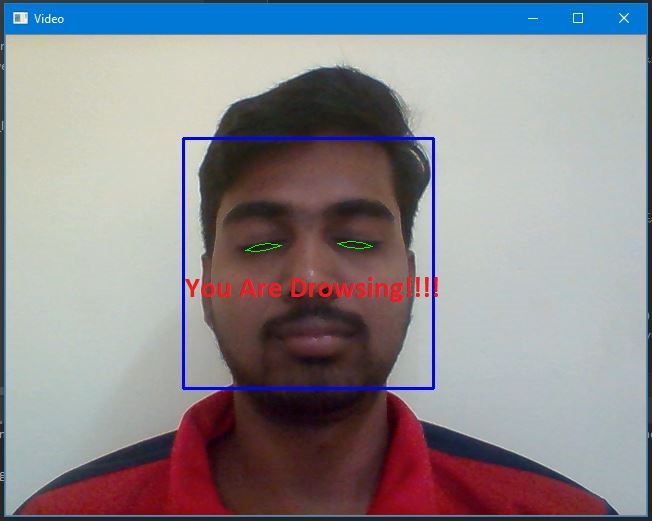
|  |
| --- |
|  |
|  | and eyes in a given input image.''' |
|  |  |
|  | #Import necessary libraries |
|  | import cv2 as cv |
|  | import numpy as np |
|  |  |
|  | #Load face cascade and hair cascade from haarcascades folder |
|  | face\_cascade = cv.CascadeClassifier("haarcascades/haarcascade\_frontalface\_default.xml") |
|  | eye\_cascade = cv.CascadeClassifier("haarcascades/haarcascade\_eye.xml") |
|  |  |
|  | #Read image in img and convert it to grayscale and store in gray. |
|  | #Image is converted to grayscale, as face cascade doesn't require to operate on coloured images. |
|  | img = cv.imread('images/test.jpg') |
|  | gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY) |
|  |  |
|  | #Detect all faces in image. |
|  | faces = face\_cascade.detectMultiScale(gray, 1.3, 5) |
|  |  |
|  | #Draw a rectangle over the face, and detect eyes in faces |
|  | for (x,y,w,h) in faces: |
|  | cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2) |
|  |  |
|  | #ROI is region of interest with area having face inside it. |
|  | roi\_gray = gray[y:y+h, x:x+w] |
|  | roi\_color = img[y:y+h, x:x+w] |
|  |  |
|  | #Detect eyes in face |
|  | eyes = eye\_cascade.detectMultiScale(roi\_gray) |
|  |  |
|  | for (ex,ey,ew,eh) in eyes: |
|  | cv.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2) |
|  |  |
|  | cv.imshow('Image', img) |
|  | cv.waitKey(0) |
|  | cv.destroyAllWindows() |

|  |  |
| --- | --- |
|  | |
|  | |
| '''This script uses OpenCV'shaarcascade | | and eyes in a video feed which can be inputted through a webcam.''' |
|  | |  |
|  | |
|  | | #Import necessary  libraries  import cv2 as cv | |
|  | | import numpy as np | |
|  | |  | |
|  | | #Load face cascade and hair cascade from haarcascades folderq | |
|  | | face\_cascade = cv.CascadeClassifier("haarcascades/haarcascade\_frontalface\_default.xml") | |
|  | | eye\_cascade = cv.CascadeClassifier("haarcascades/haarcascade\_eye.xml") | |
|  | |  | |
|  | | #Capture video from webcam | |
|  | | video\_capture = cv.VideoCapture(0) | |
|  | |  | |
|  | | #Read all frames from webcam | |
|  | | while True: | |
|  | | ret, frame = video\_capture.read() | |
|  | | frame = cv.flip(frame,1) #Flip so that video feed is not flipped, and appears mirror like. | |
|  | | gray = cv.cvtColor(frame, cv.COLOR\_BGR2GRAY) | |
|  | |  | |
|  | | faces = face\_cascade.detectMultiScale(gray, 1.3, 5) | |
|  | |  | |
|  | | for (x,y,w,h) in faces: | |
|  | | cv.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2) | |
|  | | roi\_gray = gray[y:y+h, x:x+w] | |
|  | | roi\_color = frame[y:y+h, x:x+w] | |
|  | |  | |
|  | | eyes = eye\_cascade.detectMultiScale(roi\_gray) | |
|  | |  | |
|  | | for (ex,ey,ew,eh) in eyes: | |
|  | | cv.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2) | |
|  | |  | |
|  | | cv.imshow('Video', frame) | |
|  | |  | |
|  | | if(cv.waitKey(1) & 0xFF == ord('q')): | |
|  | | break | |
|  | |  | |
|  | | #Finally when video capture is over, release the video capture and destroyAllWindows | |
|  | | video\_capture.release() | |
|  | | cv.destroyAllWindows() | |

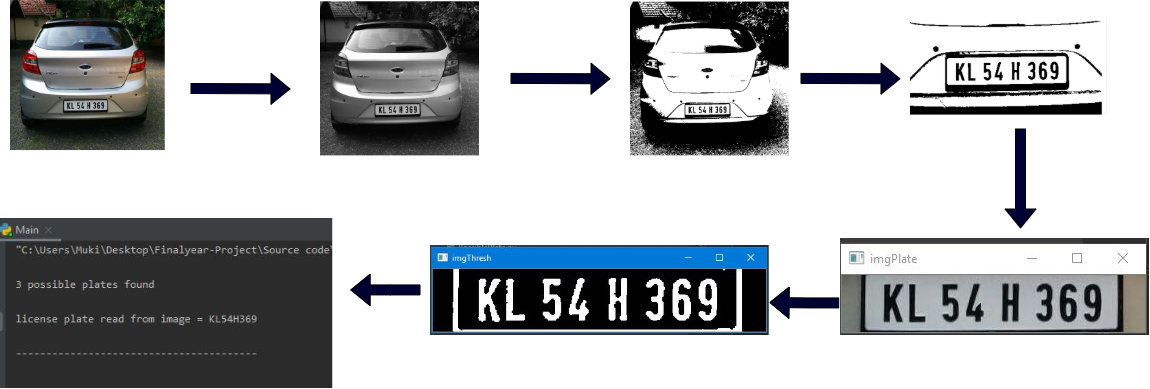
**7.1.2 NUMBER PLATE DETECTION**

|  |
| --- |
|  |
|  | import os |
|  |  |
|  | import cv2 |
|  | import numpy as np |
|  | import math |
|  | import random |
|  |  |
|  | import Main |
|  | import Preprocess |
|  | import PossibleChar |
|  |  |
|  | # module level variables ########################################################################## |
|  |  |
|  | kNearest = cv2.ml.KNearest\_create() |
|  |  |
|  | # constants for checkIfPossibleChar, this checks one possible char only (does not compare to another char) |
|  | MIN\_PIXEL\_WIDTH = 2 |
|  | MIN\_PIXEL\_HEIGHT = 8 |
|  |  |
|  | MIN\_ASPECT\_RATIO = 0.25 |
|  | MAX\_ASPECT\_RATIO = 1.0 |
|  |  |
|  | MIN\_PIXEL\_AREA = 80 |
|  |  |
|  | # constants for comparing two chars |
|  | MIN\_DIAG\_SIZE\_MULTIPLE\_AWAY = 0.3 |
|  | MAX\_DIAG\_SIZE\_MULTIPLE\_AWAY = 5.0 |
|  |  |
|  | MAX\_CHANGE\_IN\_AREA = 0.5 |
|  |  |
|  | MAX\_CHANGE\_IN\_WIDTH = 0.8 |
|  | MAX\_CHANGE\_IN\_HEIGHT = 0.2 |
|  |  |
|  | MAX\_ANGLE\_BETWEEN\_CHARS = 12.0 |
|  |  |
|  | # other constants |
|  | MIN\_NUMBER\_OF\_MATCHING\_CHARS = 3 |
|  |  |
|  | RESIZED\_CHAR\_IMAGE\_WIDTH = 20 |
|  | RESIZED\_CHAR\_IMAGE\_HEIGHT = 30 |
|  |  |
|  | MIN\_CONTOUR\_AREA = 100 |
|  |  |
|  | ################################################################################################### |
|  | def loadKNNDataAndTrainKNN(): |
|  | allContoursWithData = [] # declare empty lists, |
|  | validContoursWithData = [] # we will fill these shortly |
|  |  |
|  | try: |
|  | npaClassifications = np.loadtxt("classifications.txt", np.float32) # read in training classifications |
|  | except: # if file could not be opened |
|  | print("error, unable to open classifications.txt, exiting program\n") # show error message |
|  | os.system("pause") |
|  | return False # and return False |
|  | # end try |
|  |  |
|  | try: |
|  | npaFlattenedImages = np.loadtxt("flattened\_images.txt", np.float32) # read in training images |
|  | except: # if file could not be opened |
|  | print("error, unable to open flattened\_images.txt, exiting program\n") # show error message |
|  | os.system("pause") |
|  | return False # and return False |
|  | # end try |
|  |  |
|  | npaClassifications = npaClassifications.reshape((npaClassifications.size, 1)) # reshape numpy array to 1d, necessary to pass to call to train |
|  |  |
|  | kNearest.setDefaultK(1) # set default K to 1 |
|  |  |
|  | kNearest.train(npaFlattenedImages, cv2.ml.ROW\_SAMPLE, npaClassifications) # train KNN object |
|  |  |
|  | return True # if we got here training was successful so return true |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | def detectCharsInPlates(listOfPossiblePlates): |
|  | intPlateCounter = 0 |
|  | imgContours = None |
|  | contours = [] |
|  |  |
|  | if len(listOfPossiblePlates) == 0: # if list of possible plates is empty |
|  | return listOfPossiblePlates # return |
|  | # end if |
|  |  |
|  | # at this point we can be sure the list of possible plates has at least one plate |
|  |  |
|  | for possiblePlate in listOfPossiblePlates: # for each possible plate, this is a big for loop that takes up most of the function |
|  |  |
|  | possiblePlate.imgGrayscale, possiblePlate.imgThresh = Preprocess.preprocess(possiblePlate.imgPlate) # preprocess to get grayscale and threshold images |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | cv2.imshow("5a", possiblePlate.imgPlate) |
|  | cv2.imshow("5b", possiblePlate.imgGrayscale) |
|  | cv2.imshow("5c", possiblePlate.imgThresh) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | # increase size of plate image for easier viewing and char detection |
|  | possiblePlate.imgThresh = cv2.resize(possiblePlate.imgThresh, (0, 0), fx = 1.6, fy = 1.6) |
|  |  |
|  | # threshold again to eliminate any gray areas |
|  | thresholdValue, possiblePlate.imgThresh = cv2.threshold(possiblePlate.imgThresh, 0.0, 255.0, cv2.THRESH\_BINARY | cv2.THRESH\_OTSU) |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | cv2.imshow("5d", possiblePlate.imgThresh) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | # find all possible chars in the plate, |
|  | # this function first finds all contours, then only includes contours that could be chars (without comparison to other chars yet) |
|  | listOfPossibleCharsInPlate = findPossibleCharsInPlate(possiblePlate.imgGrayscale, possiblePlate.imgThresh) |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | height, width, numChannels = possiblePlate.imgPlate.shape |
|  | imgContours = np.zeros((height, width, 3), np.uint8) |
|  | del contours[:] # clear the contours list |
|  |  |
|  | for possibleChar in listOfPossibleCharsInPlate: |
|  | contours.append(possibleChar.contour) |
|  | # end for |
|  |  |
|  | cv2.drawContours(imgContours, contours, -1, Main.SCALAR\_WHITE) |
|  |  |
|  | cv2.imshow("6", imgContours) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | # given a list of all possible chars, find groups of matching chars within the plate |
|  | listOfListsOfMatchingCharsInPlate = findListOfListsOfMatchingChars(listOfPossibleCharsInPlate) |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | imgContours = np.zeros((height, width, 3), np.uint8) |
|  | del contours[:] |
|  |  |
|  | for listOfMatchingChars in listOfListsOfMatchingCharsInPlate: |
|  | intRandomBlue = random.randint(0, 255) |
|  | intRandomGreen = random.randint(0, 255) |
|  | intRandomRed = random.randint(0, 255) |
|  |  |
|  | for matchingChar in listOfMatchingChars: |
|  | contours.append(matchingChar.contour) |
|  | # end for |
|  | cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen, intRandomRed)) |
|  | # end for |
|  | cv2.imshow("7", imgContours) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | if (len(listOfListsOfMatchingCharsInPlate) == 0): # if no groups of matching chars were found in the plate |
|  |  |
|  | if Main.showSteps == True: # show steps ############################################### |
|  | print("chars found in plate number " + str( |
|  | intPlateCounter) + " = (none), click on any image and press a key to continue . . .") |
|  | intPlateCounter = intPlateCounter + 1 |
|  | cv2.destroyWindow("8") |
|  | cv2.destroyWindow("9") |
|  | cv2.destroyWindow("10") |
|  | cv2.waitKey(0) |
|  | # end if # show steps ################################################################# |
|  |  |
|  | possiblePlate.strChars = "" |
|  | continue # go back to top of for loop |
|  | # end if |
|  |  |
|  | for i in range(0, len(listOfListsOfMatchingCharsInPlate)): # within each list of matching chars |
|  | listOfListsOfMatchingCharsInPlate[i].sort(key = lambda matchingChar: matchingChar.intCenterX) # sort chars from left to right |
|  | listOfListsOfMatchingCharsInPlate[i] = removeInnerOverlappingChars(listOfListsOfMatchingCharsInPlate[i]) # and remove inner overlapping chars |
|  | # end for |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | imgContours = np.zeros((height, width, 3), np.uint8) |
|  |  |
|  | for listOfMatchingChars in listOfListsOfMatchingCharsInPlate: |
|  | intRandomBlue = random.randint(0, 255) |
|  | intRandomGreen = random.randint(0, 255) |
|  | intRandomRed = random.randint(0, 255) |
|  |  |
|  | del contours[:] |
|  |  |
|  | for matchingChar in listOfMatchingChars: |
|  | contours.append(matchingChar.contour) |
|  | # end for |
|  |  |
|  | cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen, intRandomRed)) |
|  | # end for |
|  | cv2.imshow("8", imgContours) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | # within each possible plate, suppose the longest list of potential matching chars is the actual list of chars |
|  | intLenOfLongestListOfChars = 0 |
|  | intIndexOfLongestListOfChars = 0 |
|  |  |
|  | # loop through all the vectors of matching chars, get the index of the one with the most chars |
|  | for i in range(0, len(listOfListsOfMatchingCharsInPlate)): |
|  | if len(listOfListsOfMatchingCharsInPlate[i]) > intLenOfLongestListOfChars: |
|  | intLenOfLongestListOfChars = len(listOfListsOfMatchingCharsInPlate[i]) |
|  | intIndexOfLongestListOfChars = i |
|  | # end if |
|  | # end for |
|  |  |
|  | # suppose that the longest list of matching chars within the plate is the actual list of chars |
|  | longestListOfMatchingCharsInPlate = listOfListsOfMatchingCharsInPlate[intIndexOfLongestListOfChars] |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | imgContours = np.zeros((height, width, 3), np.uint8) |
|  | del contours[:] |
|  |  |
|  | for matchingChar in longestListOfMatchingCharsInPlate: |
|  | contours.append(matchingChar.contour) |
|  | # end for |
|  |  |
|  | cv2.drawContours(imgContours, contours, -1, Main.SCALAR\_WHITE) |
|  |  |
|  | cv2.imshow("9", imgContours) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | possiblePlate.strChars = recognizeCharsInPlate(possiblePlate.imgThresh, longestListOfMatchingCharsInPlate) |
|  |  |
|  | if Main.showSteps == True: # show steps ################################################### |
|  | print("chars found in plate number " + str( |
|  | intPlateCounter) + " = " + possiblePlate.strChars + ", click on any image and press a key to continue . . .") |
|  | intPlateCounter = intPlateCounter + 1 |
|  | cv2.waitKey(0) |
|  | # end if # show steps ##################################################################### |
|  |  |
|  | # end of big for loop that takes up most of the function |
|  |  |
|  | if Main.showSteps == True: |
|  | print("\nchar detection complete, click on any image and press a key to continue . . .\n") |
|  | cv2.waitKey(0) |
|  | # end if |
|  |  |
|  | return listOfPossiblePlates |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | def findPossibleCharsInPlate(imgGrayscale, imgThresh): |
|  | listOfPossibleChars = [] # this will be the return value |
|  | contours = [] |
|  | imgThreshCopy = imgThresh.copy() |
|  |  |
|  | # find all contours in plate |
|  | contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE) |
|  |  |
|  | for contour in contours: # for each contour |
|  | possibleChar = PossibleChar.PossibleChar(contour) |
|  |  |
|  | if checkIfPossibleChar(possibleChar): # if contour is a possible char, note this does not compare to other chars (yet) . . . |
|  | listOfPossibleChars.append(possibleChar) # add to list of possible chars |
|  | # end if |
|  | # end if |
|  |  |
|  | return listOfPossibleChars |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | def checkIfPossibleChar(possibleChar): |
|  | # this function is a 'first pass' that does a rough check on a contour to see if it could be a char, |
|  | # note that we are not (yet) comparing the char to other chars to look for a group |
|  | if (possibleChar.intBoundingRectArea > MIN\_PIXEL\_AREA and |
|  | possibleChar.intBoundingRectWidth > MIN\_PIXEL\_WIDTH and possibleChar.intBoundingRectHeight > MIN\_PIXEL\_HEIGHT and |
|  | MIN\_ASPECT\_RATIO < possibleChar.fltAspectRatio and possibleChar.fltAspectRatio < MAX\_ASPECT\_RATIO): |
|  | return True |
|  | else: |
|  | return False |
|  | # end if |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | def findListOfListsOfMatchingChars(listOfPossibleChars): |
|  | # with this function, we start off with all the possible chars in one big list |
|  | # the purpose of this function is to re-arrange the one big list of chars into a list of lists of matching chars, |
|  | # note that chars that are not found to be in a group of matches do not need to be considered further |
|  | listOfListsOfMatchingChars = [] # this will be the return value |
|  |  |
|  | for possibleChar in listOfPossibleChars: # for each possible char in the one big list of chars |
|  | listOfMatchingChars = findListOfMatchingChars(possibleChar, listOfPossibleChars) # find all chars in the big list that match the current char |
|  |  |
|  | listOfMatchingChars.append(possibleChar) # also add the current char to current possible list of matching chars |
|  |  |
|  | if len(listOfMatchingChars) < MIN\_NUMBER\_OF\_MATCHING\_CHARS: # if current possible list of matching chars is not long enough to constitute a possible plate |
|  | continue # jump back to the top of the for loop and try again with next char, note that it's not necessary |
|  | # to save the list in any way since it did not have enough chars to be a possible plate |
|  | # end if |
|  |  |
|  | # if we get here, the current list passed test as a "group" or "cluster" of matching chars |
|  | listOfListsOfMatchingChars.append(listOfMatchingChars) # so add to our list of lists of matching chars |
|  |  |
|  | listOfPossibleCharsWithCurrentMatchesRemoved = [] |
|  |  |
|  | # remove the current list of matching chars from the big list so we don't use those same chars twice, |
|  | # make sure to make a new big list for this since we don't want to change the original big list |
|  | listOfPossibleCharsWithCurrentMatchesRemoved = list(set(listOfPossibleChars) - set(listOfMatchingChars)) |
|  |  |
|  | recursiveListOfListsOfMatchingChars = findListOfListsOfMatchingChars(listOfPossibleCharsWithCurrentMatchesRemoved) # recursive call |
|  |  |
|  | for recursiveListOfMatchingChars in recursiveListOfListsOfMatchingChars: # for each list of matching chars found by recursive call |
|  | listOfListsOfMatchingChars.append(recursiveListOfMatchingChars) # add to our original list of lists of matching chars |
|  | # end for |
|  |  |
|  | break # exit for |
|  |  |
|  | # end for |
|  |  |
|  | return listOfListsOfMatchingChars |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | def findListOfMatchingChars(possibleChar, listOfChars): |
|  | # the purpose of this function is, given a possible char and a big list of possible chars, |
|  | # find all chars in the big list that are a match for the single possible char, and return those matching chars as a list |
|  | listOfMatchingChars = [] # this will be the return value |
|  |  |
|  | for possibleMatchingChar in listOfChars: # for each char in big list |
|  | if possibleMatchingChar == possibleChar: # if the char we attempting to find matches for is the exact same char as the char in the big list we are currently checking |
|  | # then we should not include it in the list of matches b/c that would end up double including the current char |
|  | continue # so do not add to list of matches and jump back to top of for loop |
|  | # end if |
|  | # compute stuff to see if chars are a match |
|  | fltDistanceBetweenChars = distanceBetweenChars(possibleChar, possibleMatchingChar) |
|  |  |
|  | fltAngleBetweenChars = angleBetweenChars(possibleChar, possibleMatchingChar) |
|  |  |
|  | fltChangeInArea = float(abs(possibleMatchingChar.intBoundingRectArea - possibleChar.intBoundingRectArea)) / float(possibleChar.intBoundingRectArea) |
|  |  |
|  | fltChangeInWidth = float(abs(possibleMatchingChar.intBoundingRectWidth - possibleChar.intBoundingRectWidth)) / float(possibleChar.intBoundingRectWidth) |
|  | fltChangeInHeight = float(abs(possibleMatchingChar.intBoundingRectHeight - possibleChar.intBoundingRectHeight)) / float(possibleChar.intBoundingRectHeight) |
|  |  |
|  | # check if chars match |
|  | if (fltDistanceBetweenChars < (possibleChar.fltDiagonalSize \* MAX\_DIAG\_SIZE\_MULTIPLE\_AWAY) and |
|  | fltAngleBetweenChars < MAX\_ANGLE\_BETWEEN\_CHARS and |
|  | fltChangeInArea < MAX\_CHANGE\_IN\_AREA and |
|  | fltChangeInWidth < MAX\_CHANGE\_IN\_WIDTH and |
|  | fltChangeInHeight < MAX\_CHANGE\_IN\_HEIGHT): |
|  |  |
|  | listOfMatchingChars.append(possibleMatchingChar) # if the chars are a match, add the current char to list of matching chars |
|  | # end if |
|  | # end for |
|  |  |
|  | return listOfMatchingChars # return result |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | # use Pythagorean theorem to calculate distance between two chars |
|  | def distanceBetweenChars(firstChar, secondChar): |
|  | intX = abs(firstChar.intCenterX - secondChar.intCenterX) |
|  | intY = abs(firstChar.intCenterY - secondChar.intCenterY) |
|  |  |
|  | return math.sqrt((intX \*\* 2) + (intY \*\* 2)) |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | # use basic trigonometry (SOH CAH TOA) to calculate angle between chars |
|  | def angleBetweenChars(firstChar, secondChar): |
|  | fltAdj = float(abs(firstChar.intCenterX - secondChar.intCenterX)) |
|  | fltOpp = float(abs(firstChar.intCenterY - secondChar.intCenterY)) |
|  |  |
|  | if fltAdj != 0.0: # check to make sure we do not divide by zero if the center X positions are equal, float division by zero will cause a crash in Python |
|  | fltAngleInRad = math.atan(fltOpp / fltAdj) # if adjacent is not zero, calculate angle |
|  | else: |
|  | fltAngleInRad = 1.5708 # if adjacent is zero, use this as the angle, this is to be consistent with the C++ version of this program |
|  | # end if |
|  |  |
|  | fltAngleInDeg = fltAngleInRad \* (180.0 / math.pi) # calculate angle in degrees |
|  |  |
|  | return fltAngleInDeg |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | # if we have two chars overlapping or to close to each other to possibly be separate chars, remove the inner (smaller) char, |
|  | # this is to prevent including the same char twice if two contours are found for the same char, |
|  | # for example for the letter 'O' both the inner ring and the outer ring may be found as contours, but we should only include the char once |
|  | def removeInnerOverlappingChars(listOfMatchingChars): |
|  | listOfMatchingCharsWithInnerCharRemoved = list(listOfMatchingChars) # this will be the return value |
|  |  |
|  | for currentChar in listOfMatchingChars: |
|  | for otherChar in listOfMatchingChars: |
|  | if currentChar != otherChar: # if current char and other char are not the same char . . . |
|  | # if current char and other char have center points at almost the same location . . . |
|  | if distanceBetweenChars(currentChar, otherChar) < (currentChar.fltDiagonalSize \* MIN\_DIAG\_SIZE\_MULTIPLE\_AWAY): |
|  | # if we get in here we have found overlapping chars |
|  | # next we identify which char is smaller, then if that char was not already removed on a previous pass, remove it |
|  | if currentChar.intBoundingRectArea < otherChar.intBoundingRectArea: # if current char is smaller than other char |
|  | if currentChar in listOfMatchingCharsWithInnerCharRemoved: # if current char was not already removed on a previous pass . . . |
|  | listOfMatchingCharsWithInnerCharRemoved.remove(currentChar) # then remove current char |
|  | # end if |
|  | else: # else if other char is smaller than current char |
|  | if otherChar in listOfMatchingCharsWithInnerCharRemoved: # if other char was not already removed on a previous pass . . . |
|  | listOfMatchingCharsWithInnerCharRemoved.remove(otherChar) # then remove other char |
|  | # end if |
|  | # end if |
|  | # end if |
|  | # end if |
|  | # end for |
|  | # end for |
|  |  |
|  | return listOfMatchingCharsWithInnerCharRemoved |
|  | # end function |
|  |  |
|  | ################################################################################################### |
|  | # this is where we apply the actual char recognition |
|  | def recognizeCharsInPlate(imgThresh, listOfMatchingChars): |
|  | strChars = "" # this will be the return value, the chars in the lic plate |
|  |  |
|  | height, width = imgThresh.shape |
|  |  |
|  | imgThreshColor = np.zeros((height, width, 3), np.uint8) |
|  |  |
|  | listOfMatchingChars.sort(key = lambda matchingChar: matchingChar.intCenterX) # sort chars from left to right |
|  |  |
|  | cv2.cvtColor(imgThresh, cv2.COLOR\_GRAY2BGR, imgThreshColor) # make color version of threshold image so we can draw contours in color on it |
|  |  |
|  | for currentChar in listOfMatchingChars: # for each char in plate |
|  | pt1 = (currentChar.intBoundingRectX, currentChar.intBoundingRectY) |
|  | pt2 = ((currentChar.intBoundingRectX + currentChar.intBoundingRectWidth), (currentChar.intBoundingRectY + currentChar.intBoundingRectHeight)) |
|  |  |
|  | cv2.rectangle(imgThreshColor, pt1, pt2, Main.SCALAR\_GREEN, 2) # draw green box around the char |
|  |  |
|  | # crop char out of threshold image |
|  | imgROI = imgThresh[currentChar.intBoundingRectY : currentChar.intBoundingRectY + currentChar.intBoundingRectHeight, |
|  | currentChar.intBoundingRectX : currentChar.intBoundingRectX + currentChar.intBoundingRectWidth] |
|  |  |
|  | imgROIResized = cv2.resize(imgROI, (RESIZED\_CHAR\_IMAGE\_WIDTH, RESIZED\_CHAR\_IMAGE\_HEIGHT)) # resize image, this is necessary for char recognition |
|  |  |
|  | npaROIResized = imgROIResized.reshape((1, RESIZED\_CHAR\_IMAGE\_WIDTH \* RESIZED\_CHAR\_IMAGE\_HEIGHT)) # flatten image into 1d n |
|  |  |
|  |  |
|  |  |
|  | # finally we can call findNearest !!! |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | if Main.showSteps == True: # show steps ####################################################### |
|  | cv2.imshow("10", imgThreshColor) |
|  | # end if # show steps ######################################################################### |
|  |  |
|  | return strChars |
|  | # end function |

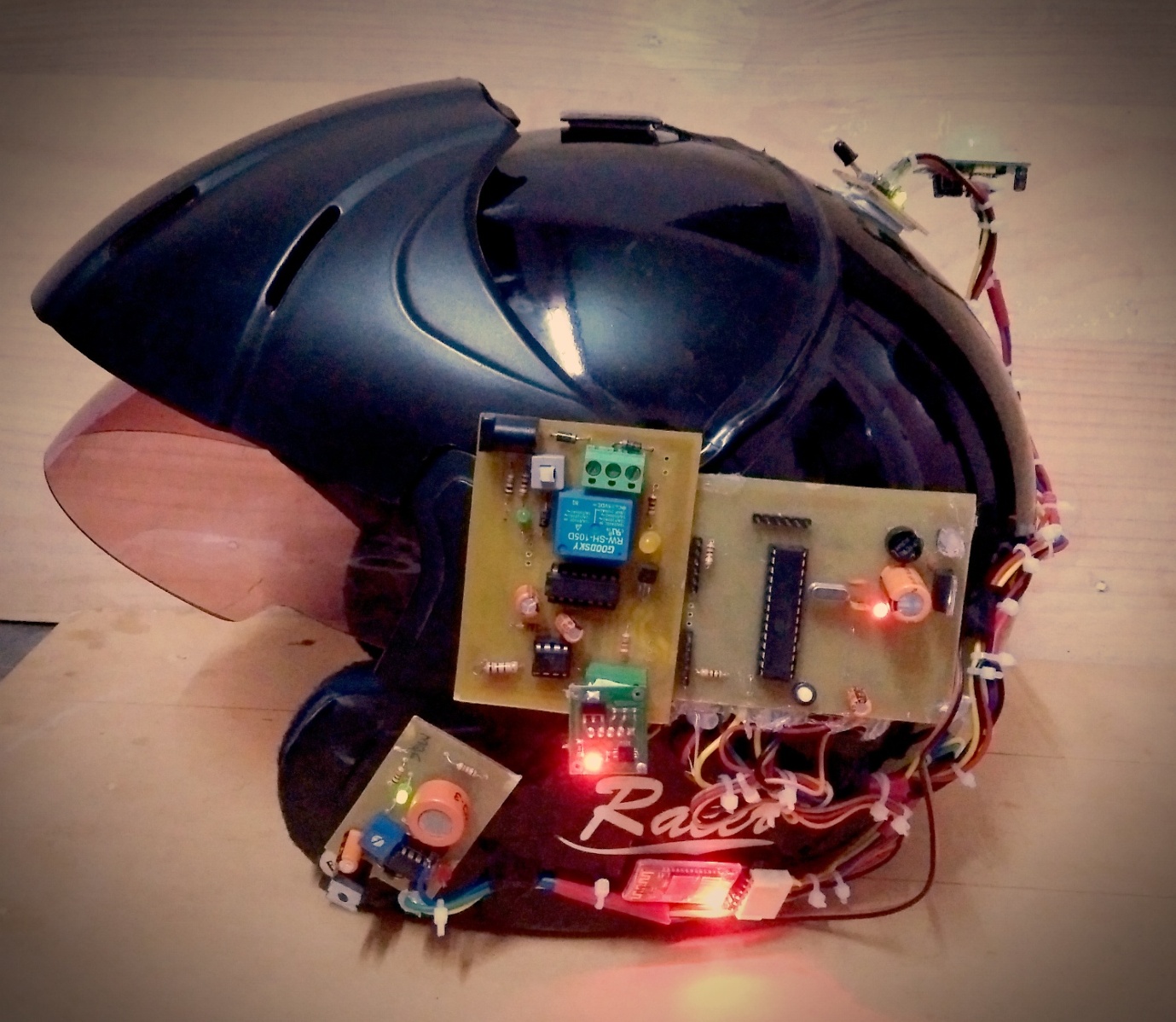
**7.2 SCREENSHOTS**

** **

****

****

****

****

**REFERENCES**

1. Prof.Pallavi khare, Rupesh Dudhe, Amolsing Chungade, and Ajit Naykinde, “Advanced license number plate recognition system International Journal of Engineering Research & Technology (IJERT), ICONECT-2015.
2. Anish ANISH LAZRUS1, SIDDHARTHA CHOUBEY2, SINHA G.R.3 , IEEE” International Journal of Machine Intelligence ISSN: 0975–2927 & E-ISSN: 0975–9166, Volume 3, Issue 3, 2011, pp-134-137.
3. Rajneesh, Anudeep Goraya, Gurmeet Singh, “Geert Real Time Drivers Drowsiness Detection and alert System by Measuring EAR,”International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 25, November- 2018.
4. Haoyang Sayeed and A. Perrig, “Secure Wireless Communications: Secret Keys through Multipath,” Proc. IEEE Int’l Conf. Acoustics, SpeechSignal Processing, pp. 3013-3016, Apr.2008
5. One Millisecond Face Alignment with an Ensemble of Regression Trees by Vahid Kazemi and Josephine Sullivan KTH, Royal Institute of Technology Computer Vision and Active Perception Lab Teknikringen 14, Stockholm, Sweden.
6. Real-Time Eye Blink Detection using Facial Landmarks by Tereza Soukupova and Jan Cech, Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague .
7. MarcoJavier Flores, JoséMaría Armingol and Arturo de la Escalera, ―Driver Drowsiness Warning System Using Visual Information for Both Diurnal and Nocturnal Illumination Conditions‖, Springer, EURASIP Journal on Advances in Signal Processing, 2010.
8. Belhassen AkroutWalid Mahdi, ―A Blinking Measurement Method for Driver Drowsiness Detection‖, Springer, Proceedings of the 8th International Conference on Computer Recognition Systems CORES, pp 651-660, 2013.
9. Krajewski J, Sommer D, Trutschel U, Edwards D, Golz M. Steering wheel behavior based estimation of fatigue. The fifth international driving symposium on human factors in driver assessment, training and vehicle design 2009;118-124.
10. Mardi Z, Ashtiani SN, Mikaili M. EEG-based drowsiness detection for safe driving using chaotic features and statistical tests. Journal of medical signals and sensors 2011;1:130–137 [Dino Ienco](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Dino%20Ienco.QT.&newsearch=true), [Raffaele Gaetano](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Raffaele%20Gaetano.QT.&newsearch=true), [Claire Dupaquier](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Claire%20Dupaquier.QT.&newsearch=true),” Land Cover Classification via Multitemporal Spatial Data by Deep Recurrent Neural Networks“ IEEE geoscience and remote sensing letters, vol. 14, no. 10, 2017,pp. 1685 – 1689.
11. National Highway Traffic Safety Administration (NHTSA): Drowsy driving and automobile crashes.
12. International Journal of Operations and Logistics Management www.absronline.org/journals p-ISSN: 2310-4945; e-ISSN: 2309-8023 Volume: 3, Issue: 3, Pages: 222-240 (September 2014) © Academy of Business & Scientific Research .
13. Method of detecting drowsiness level by utilizing blinking duration Kazuhiko Sugiyama a, Tomoaki Nakano a, Shin Yamamoto a, Toshikazu Ishihara a, Hiroyuki Fujii b, Eisaku Akutsu c.
14. Association for Safe International Road Travel (ASIRT), Road Crash Statistics.http://asirt.org/initiatives/informingroadusers/road-safety-facts/road-crash-statistics,2016.
15. Journal of VLSI Signal Processing 23, 497–511 (1999) c °1999 Kluwer Academic Publishers. Manufactured in The Netherlands.
16. Eye Detection Using Morphological and Color Image Processing Tanmay Rajpathaka, Ratnesh Kumar and Eric Schwartzb .
17. A Robust Algorithm for Eye Detection on Grey Intensity Face without Spectacles- JCS&T Vol. 5 No. 3
18. Froba Kebbuck: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.
19. Driver Drowsiness Detection using Eye-Closeness Detection (2016 12th International Conference on Signal-Image Technology & Internet-Based Systems).
20. Eye Detection Using Morphological and Color Image Processing Tanmay Rajpathaka, Ratnesh Kumar and Eric Schwartzb
21. A Robust Algorithm for Eye Detection on Grey Intensity Face without Spectacles- JCS&T Vol. 5 No. 3
22. Froba Kebbuck: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.
23. Driver Drowsiness Detection using Eye-Closeness Detection (2016 12th International Conference on Signal-Image Technology & Internet-Based Systems)
24. S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (ALPR): A state-of-the-art review,” IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 2, pp. 311-325, 2013.
25. Christos Nikolaos E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Vassili Loumos, and Eleftherios Kayafas, “A License Plate Recognition Algorithm for Intelligent Transportation System Applications”, IEEE, Transactions on Intelligent Transportation Systems. Vol. 7, No.3, September 2006. pp. 377-392.

**CONFERENCE**

* Dr. C R Dhivyaa, K Mugunthan, Aaroon D Sanju, P Naveen,

(MARCH 2020),” Enhancing Road Safety System Using Machine Learning

Techniques”,International Coneference on power Engneering(Powercoin’20) at

Sona College of Technology, Salem-5