Eye closure rate of Real-Time Driver-Drowsiness Detection System

A project report

Submitted in the partial fulfillment of the requirements for the

award of the degree of

Bachelor of Technology

in

Department of ELECTRONICS AND COMMUNICATION

BY

M.NVVSLD SANTHOSHI	170040519
R. VAMSI	170040735
U. TEJASRI	170040889

\

Under the supervision of

Mr. M. VENKATESWA RAO

(DEEMED TO BE UNIVERSITY)

Koneru Lakshmaiah Education Foundation

(Deemed to be University estd., u/s 3 of UGC Act 1956)

Greenfields, Vaddeswaram, Guntur (Dist.), Andhra Pradesh - 522502

November, 2019

Declaration

The Project Report entitled “Eye ratio of Real-Time Driver-Drowsiness Detection System “is a record of Bonafide work of M. SANTHOSHI (170040519), R. VAMSI (170040735), U. TEJASRI (170040889) submitted in partial fulfillment for the award of B.Tech in Electronics and Communication Engineering to the K L University. The results embodied in this report have not been copied from any other departments/University/Institute.

M.SANTHOSHI (170040519)

R. VAMSI (170040735)

U. TEJASRI (170040889)

Certificate

The Project Report entitled “Eye ratio of Real-Time Driver-Drowsiness Detection System “is a record of bonafide work of M. SANTHOSHI (170040519), R. VAMSI (170040735), U. TEJASRI (170040889) submitted in partial fulfillment for the award of B.TECH in Electronics and Communication Engineering to the K L University.

The results embodied in this report have not been copied from any other departments/ University/Institute.

Signature of the Supervisor

Signature of the HOD **Signature of the External Examiner**

ACKNOWLEDGEMENT

Apart from the efforts of ours, the success of any work depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this TERM PAPER based project report.

We express our gratitude to our Project Guide **Dr. M. VENKATESWA RAO**, Department of Electronics **&** Communication Engineering and**.** We can't thank him enough for tremendous support and help. We feel motivated and encouraged every time we attend his meeting. Without his encouragement and guidance this Project would not have materialized.

We are thankful to our Head of the Department **Dr. SUMAN MALOJI,** Professor, Department of Electronics **&** Communication Engineering; who modeled us both technically and morally for achieving greater success in life.

We would like to show our greatest appreciation to **Dr. Pranvir Singh Satvat**, Dean Academics, KLEF for his valuable suggestions and statements.

Finally, we owe a lot to the teaching and non-teaching staff of the Dept. of Electronics **&** Communication Engineering for their direct or indirect support in doing our Lab based project work.

ABSTRACT

The research work is on the driver drowsiness detection which is uniquely focused on eye conclusion, which improves traffic security. Sleepy driving is one of the primary driver of auto collisions. As per the National Sleep Foundation, about portion of U.S. grown-up drivers to reliably getting in the driver's seat while feeling lazy. About 20% drivers fall in lay down with over 40% drivers this has occurred at any rate once in their driving vocation. In this paper, we will utilize Artificial Intelligence-based progressed calculations to identify driver laziness and the rate at which the driver is tired. We propose a calculation that utilizations eye conclusion designed facial highlights to identify driver laziness.
By the inadequacies of past calculations, we present another face-following calculation to improve the following exactness. We extricated side effects from eye area are

(1) level of eye conclusion,

(2) eyelid distance changes as for the ordinary eyelid distance, and

(3) eye conclusion rate.

In these mishaps, weakness driving caused around 20% t0 30% car crashes. Consequently, exhausted driving is a huge and risk in car crashes.

The first and second side effects identified with eye area are utilized for weakness location. Exact outcomes show that the proposed model can accomplish precision of 84% utilizing Open CV

TABLE OF CONTENTS

CONTENT	PAGE NO
LITERATURE SURVEY	5
1. INTRODUCTION	
1.1. IMPACT OF DROWSINESS ON DRIVING	7
1.2. CNN	7
1.3. WHAT IS COMPUTER VISION?	7
1.4. FACE TRACKING	8
1.4.1. COMPARISON WITH STATE OF ART APPROACH	8
1.5. VISUAL OBJECT TRACKING	9
1.6. EYE DETECTION USING OPEN CV	9
2. IMAGE PROCESSING	9
2.1. CHANGING COLOR SPACES	10-11
2.2. RESIZING IMAGES	11-12
2.3. OUTLINING	12-13
2.4. SIMPLE IMAGE THRESHOLDING	13-14
2.5. IMAGE FILTERING	15
2.6. IMAGE SEGMENTATION (WATERSHED ALGORITHM)	16-17
2.7. SPEEDED-UP ROBUST FEATURES (SURF)	18-19
3. OBJECT DETECTION	19
3.1. VIDEO ANALYSIS	20
3.2. CORE FUNCTIONALITY	20
3.3. VIDEO I/O	20
3.4. OPENCV - STORING IMAGES	21
3.5. CREATING AND DISPLAYING THE MATRIX	21-22
3.6. LOADING IMAGE	22
4. AUTOMATIC MEMORY MANAGEMENT	23
4.1. AUTOMATIC ALLOCATION OF THE OUTPUT DATA	23-24
4.2. MULTI-THREADING AND RE-ENTERABILITY	24
5. FACE DETECTION	24
5.1. FACE DETECTION USE CASE	25-26
6. D LIB	27-29
6.1. ALGORITHM	30
6.2. CNN BASED FACE DETECTOR FROM DLIB	31-32
6.3. DETECTING FACES AT ODD ANGLES	32-34

7. THEORETICAL ANALYSIS 35
   7.1. EYE ASPECT RATIO 35-37
   7.2. INPUT ARRAY AND OUTPUT ARRAY 38
   7.3. ERROR HANDLING 38
8. EXPERIMENTAL INVESTIGATION 39-44
9. RESULTS 45
10. CONCLUSION 46
11. REFERENCES 47

LITERATURE SURVEY

The ongoing driving conduct checking assumes a huge part in clever transportation frameworks. Such observing expands traffic security by diminishing and wiping out the danger of potential auto collisions. Vision-based methodology including camcorders for perilous circumstance location is without a doubt one of the most point of view and regularly utilized in detecting driver climate. For this situation, the pictures of a driver, caught with camcorders, can portray its facial highlights, similar to head developments, eye state, mouth state, and, a short time later, distinguish a momentum level of weakness state. In this paper, we influence the underlying forward looking camera of the cell phone to constantly follow driving facial highlights and early perceive driver's sleepiness and interruption risky states. Risky state acknowledgment is characterized into on the web and disconnected modes. Because of proficiency and execution of cell phones in online mode, the driving risky states are resolved continuously on the cell phones with help of PC vision libraries OpenCV and Dlib while driving. Something else, the disconnected mode depends on the aftereffects of factual examination gave by a cloud administration, using the aggregated insights progressively, yet additionally the recently gathered, put away and created by AI devices

1. INTRODUCTION

One of the major causes behind the road accidents is driver's drowsiness. After continuous driving for long time, drivers easily get tired resulting into driver fatigue and drowsiness. As indicated by the National Highway Traffic Safety Administration, consistently around 100,000 police-revealed crashes include languid driving. These accidents bring about in excess of 1,550 fatalities and 71,000 wounds. The genuine number might be a lot higher, notwithstanding, as it is hard to decide if a driver was tired at the hour of an accident.

Exhaustion at the driving controls is regularly the main driver of genuine mishaps. Coming up next are signs and side effects of sluggish driving, as indicated by the American Academy of Sleep Medicine:(1) Frequent yawning or trouble keeping your eyes open (2)"Nodding off" or experiencing difficulty keeping your head up (3) Inability to recall traveling the last couple of miles (4) Missing street signs or turns (5) Difficulty keeping up your speed (6) Drifting out of your path. In any case, as indicated by our perspective we are focusing just on flickering of eyes [1].

These surprising figures show how predominant lazy driving is. What drivers may not understand is how much languid driving puts themselves – and others – in danger. Indeed, an expected 5,000 individuals kicked the bucket in 2015 in accidents including tired driving, as indicated by a Governors Highway Safety Association report.

The first and second indications related to eye district are used for depletion acknowledgment. The proposed model can achieve accuracy of 84% using open cv computation and dlib library. which get the face air and gives the gathering yet like we require simply eye end movement of action. So we get eye end educational assortment for our specific.

OpenCV, or Opensource Computer Vision library, begun as an exploration venture at Intel. It's at present the biggest PC vision library regarding the sheet number of capacities it holds. OpenCV contains executions of in excess of 2500 calculations.
Open CV is one of the profound learning methods with wide and profound structure. Where pixels separated from pictures and move those pixels to neural organizations. we measure the pixels of the picture for less unpredictability of the model.

1.1. Impact of Drowsiness on Driving:

Driving while languid is like driving under impact of liquor:

Drivers' response times, consciousness of dangers and capacity to support consideration all demolish the drowsier the driver

Driving subsequent to going over 20 hours without rest is what could be compared to driving with a blood-liquor grouping of 0.08% – the U.S. legitimate cut-off You are multiple times bound to be in a fender bender on the off chance that you are exhausted

A driver probably won't know when the person is exhausted on the grounds that indications of weakness are difficult to recognize. A few people may likewise encounter miniature rest – short, compulsory times of distractedness[2]. In the 4 or 5 seconds a driver encounters miniature rest, at parkway speed, the vehicle will venture to every part of the length of a football field.

1.2. CNN

Convolution Neural Networks (CNN) extricate the highlights of a picture and diminish the size without loss of its qualities. In any case, by utilizing Convolution neural organizations (CNN) we can diminish the quantity of neurons with the assistance of the conv and pooling layers, where we can lessen the quantity of calculations in a completely associated network[3]. In profound learning, a convolutional neural organization is a class of profound neural organizations, most regularly applied to examining visual symbolism. They are otherwise called move invariant or space invariant fake neural networks, based on their shared-loads engineering and interpretation invariance characteristics

1.3. What is Computer Vision?

Let me rapidly clarify what PC vision is before we jump into OpenCV. It's acceptable to have an instinctive comprehension of what we'll be discussing through the remainder of the article. The capacity to see and see the world falls into place without any issues for us people. It's natural for us to assemble data from our environmental factors through the endowment of vision and insight.

1.4. FACE TRACKING:

A face following camera catches video pictures that are sent to the face following programming. The product at that point utilizes AI calculations to distinguish faces. When a face has been recognized, the product can chase after that face inside the video transfer and to break down facial highlights and demeanors continuously.

Face Tracking Technology identifies and tracks the presence of a human face in a computerized video outline. This innovation can be consolidated into PC and portable applications, and can even be utilized in advanced mechanics. Face Tracking Technology can be utilized on the web or disconnected.

1.4.1. Comparison with state of art approach:

We have utilized the EAR (eye perspective proportion) and proposed an approach to figure ECR (Eye Closure Ratio). Contrasted with different strategies from the writing, our proposed calculation gives better precision and lessens reaction season of ascertaining the EAR at worker as the worker is locally arrangement and furthermore the returned EAR esteem is privately checked in the android gadget of the driver consequently improving the aftereffects of readiness when the driver feels languid. Additionally, in other meddlesome methods(T. Hwang, M. Kim, S. Hong, and K. S. Park,2016)[4], (S. Junawane, S. Jagtap, P. Deshpande, and L. Soni,2017), various machines and gadgets should be connected to the driver's body were required, consequently making it awkward for the driver to focus on his driving. Additionally, in past methodologies an arrangement should be played out without fail, at whatever point the driver begins the ride. Notwithstanding, these meddling strategies include a decent measure of cost to quantify beat rate, pulses, and so forth In our proposed measure, we have recently utilized an android gadget and a neighborhood worker to identify languor that eliminates the components of cost of machines and break in driver's focus. In contrast with the utilization of outer camera in the meddlesome strategies, we have utilized the android gadget, which is frequently utilized by individuals for route and different purposes. The proposed calculation has functioned admirably in conditions having great lightning. It additionally works for individuals wearing scenes. Following area depicts the presentation assessment of the proposed approach.

1.5. VISUAL OBJECT TRACKING:

Visual Object Tracking is one of the chief difficulties in Computer Vision, where the undertaking is to find a specific article in all edges of a video, given just its area in the primary casing. Some model edges demonstrating the scale variation of our methodology.

1.6. EYE DETECTION USING OPEN CV:

The full type of Open CV is Open-Source Computer Vision Library. It is an Open Source-library that incorporates a few several Computer Vision Algorithms.

Open CV is a library of programming capacities primarily focused on continuous PC vision.

Open CV has a few modules[5].

2. Image Processing:

An image Processing module contains direct and non – linear image sifting and image changes.

Image Processing is most usually named as 'computerized Image Processing' and the space wherein it is habitually utilized is 'PC Vision’. Both Image Processing calculations and Computer Vision (CV) calculations accept a picture as info, notwithstanding, in picture preparing the yield is additionally a picture, while in PC vision the yield can be some data about the image.

Face acknowledgment with OpenCV, Python, and profound learning:

Inside this instructional exercise, you will figure out how to perform facial acknowledgment utilizing OpenCV, Python, and profound learning.

We'll begin with a concise conversation of how profound learning-based facial acknowledgment functions, including the idea of "profound measurement learning".

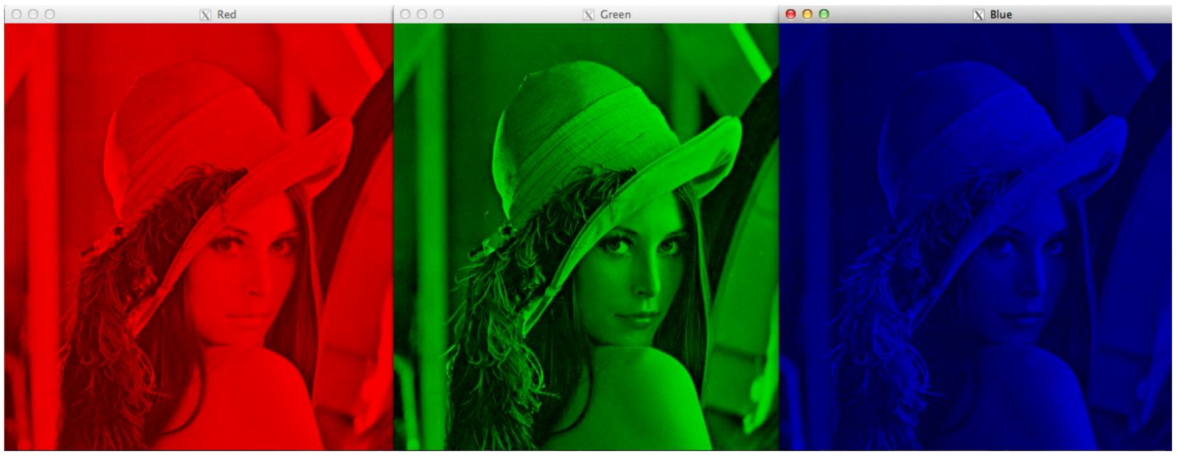
From that point, I will assist you with introducing the libraries you need to really perform face acknowledgment.

2.1. Changing Color Spaces:

A shading space is a convention for speaking to colours such that makes them effectively reproducible. We realize that grayscale pictures have single pixel esteems and shading pictures contain 3 qualities for every pixel – the powers of the Red, Green and Blue channels.

Most PC vision use cases measure pictures in RGB plan. In any case, applications like video weight and contraption self-ruling limit – these are vivaciously dependent on other concealing spaces, like the Hue-Saturation-Value or HSV concealing space[6]. As you comprehend a RGB picture comprises of the shading power of various shading channels, i.e., the force and shading data are blended in RGB shading space however in HSV shading space the shading and power data are isolated from one another. This makes HSV shading space more hearty to lighting changes.

```
#import the required libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline
image = cv2.imread('index.jpg')
#converting image to Gray scale
gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#plotting the grayscale image
plt.imshow(gray_image)
#converting image to HSV format
hsv_image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
#plotting the HSV image
plt.imshow(hsv_image)
```



2.2. Resizing Images:

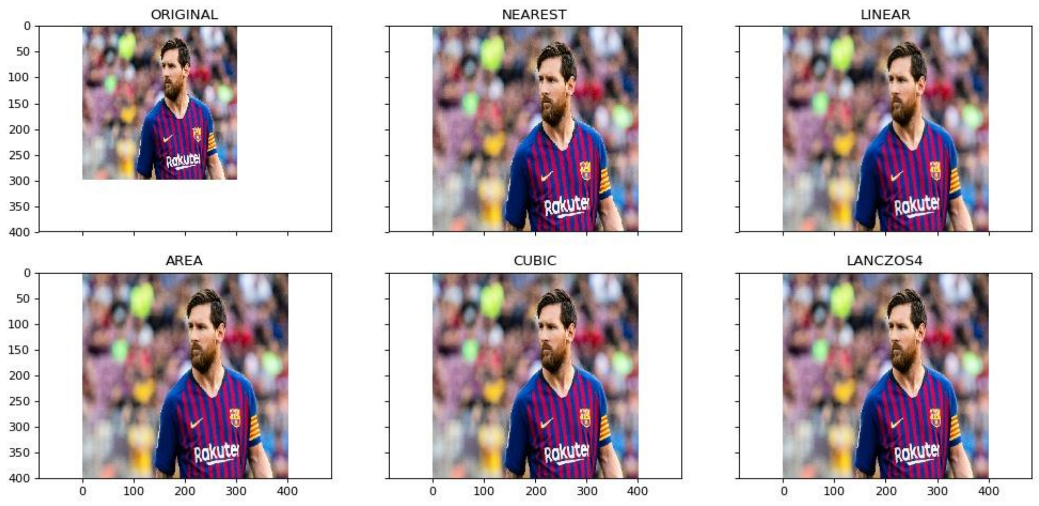
AI models work with a fixed measured info. A similar thought applies to PC vision models also. The pictures we use for preparing our model should be of a similar size.

Presently this may get tricky on the off chance that we are making our own dataset by scratching pictures from different sources. That is the place where the capacity of resizing pictures goes to the front.

Pictures can be effortlessly scaled all over utilizing OpenCV. This activity is helpful for preparing profound learning models when we need to change pictures over to the model's information shape. Diverse interjection and down sampling techniques are upheld by OpenCV, which can be utilized by the accompanying boundaries:

INTER_NEAREST: Nearest neighbour addition
INTER_LINEAR: Bilinear addition
INTER_AREA: Resampling utilizing pixel territory connection
INTER_CUBIC: Bicubic insertion over 4×4 pixel area
INTER_LANCZOS4: Lanczos introduction over 8×8 area

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#reading the image
image = cv2.imread('index.jpg')
#converting image to size (100,100,3)
smaller_image
cv2.resize(image,(100,100),inerpolation='linear')
#plot the resized image
plt.imshow(smaller_image)
```



2.3. Outlining:

We have changed over video into outlines utilizing the OpenCV library of python. We have caught the video utilizing the Video Capture technique for OpenCV and developed each edge by the reading strategy remembered for the library, composed these pictures into an envelope utilizing compose technique from the OpenCV library[7]. The dataset we had was videos(30fps) of tired individuals. By outlining the recordings we changed over each video into 1500+ casings(recordings had shifted span, so we accepting 1500 examples as breaking point).

We have utilized Eye angle Ratio(EAR) for recognizing the casings wherein the subject is sluggish, Mouth perspective ratio(MAR) is considered for mouth district which is another highlight in the model proposed. EAR is determined by adding the two good ways from the top finish to the base finish of the eye, which is then separated by the even distance of the eye. Mouth angle ratio(MAR) is likewise determined comparatively, where both are clarified underneath.

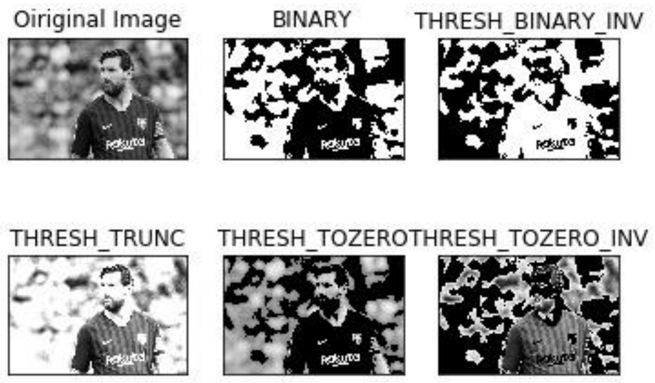
2.4. Simple Image Thresholding:

Thresholding is a picture division technique. It contrasts pixel esteems and a limit worth and updates it likewise. OpenCV underpins different varieties of thresholding. A basic thresholding capacity can be characterized this way:

if Image(x,y) > threshold , Image(x,y) = 1

otherwise, Image(x,y) = 0

Thresholding can only be applied to grayscale images.

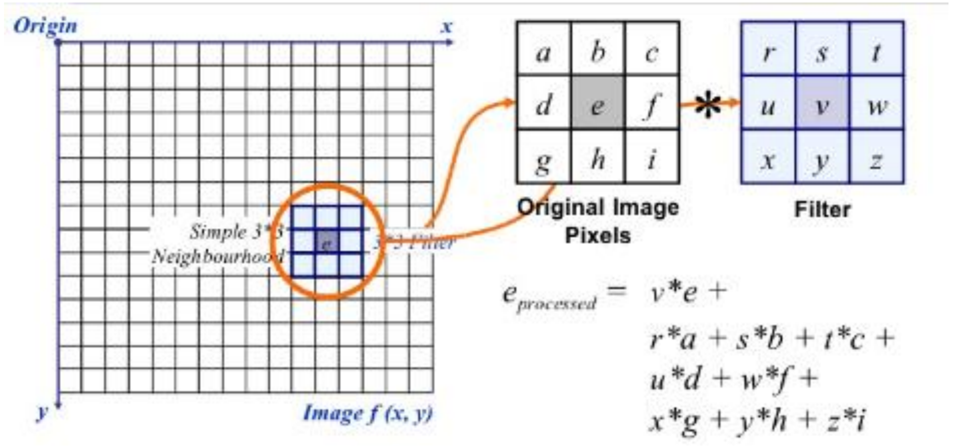


```
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
#here 0 means that the image is loaded in gray scale format
gray_image = cv2.imread('index.png',0)

ret,thresh_binary =
cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY)
ret,thresh_binary_inv =
cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY_INV)
ret,thresh_trunc =
cv2.threshold(gray_image,127,255,cv2.THRESH_TRUNC)
ret,thresh_tozero =
cv2.threshold(gray_image,127,255,cv2.THRESH_TOZERO)
ret,thresh_tozero_inv =
cv2.threshold(gray_image,127,255,cv2.THRESH_TOZERO_INV)
#DISPLAYING THE DIFFERENT THRESHOLDING STYLES
names = ['Oiriginal
Image','BINARY','THRESH_BINARY_INV','THRESH_TRUNC','THRESH_TOZER
O','THRESH_TOZERO_INV']
images =
gray_image,thresh_binary,thresh_binary_inv,thresh_trunc,thresh_t
ozero,thresh_tozero_inv
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(names[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

2.5. Image Filtering:

In image filtering, a pixel esteem is refreshed utilizing its neighbouring qualities. Yet, how are these qualities refreshed in any case?

Indeed, there are various methods of refreshing pixel esteems, for example, choosing the greatest incentive from neighbours, utilizing the normal of neighbours, and so on Every technique has its own employments. For instance, averaging the pixel esteems in an area is utilized for picture.



Gaussian filtering is additionally utilized for picture obscuring that gives various loads to the neighboring pixels dependent on their separation from the pixel viable.

For picture separating, we use parts. Portions are frameworks of quantities of various shapes like 3 x 3, 5 x 5, and so forth A portion is utilized to compute the spot item with a piece of the picture. While figuring the new estimation of a pixel [8], the part community is covered with the pixel. The neighboring pixel esteems are duplicated with the comparing esteems in the piece. The determined worth is relegated to the pixel agreeing with the focal point of the bit.

2.6. Image Segmentation (Watershed Algorithm):

Picture division is the endeavor of describing every pixel in the image to some class. For example, gathering every pixel as nearer view or establishment. Picture division is critical for isolating the appropriate parts from a picture.

The watershed estimation is a praiseworthy picture division figuring. It considers the pixel regards in an image as topography. For finding as far as possible, it acknowledges starting markers as information. The count by then starts flooding the bowl from the markers till the markers meet at as far as possible [9].

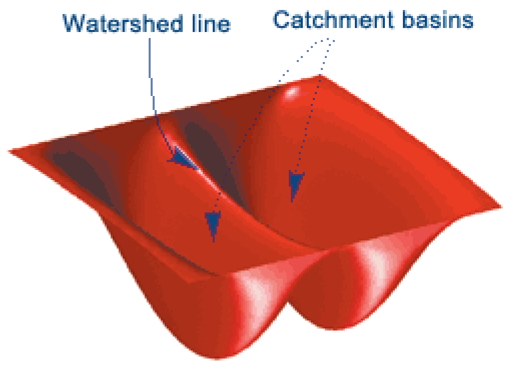


Image Source :- Mathworks

Suppose we have a geography with various bowls. Presently, in the event that we fill various bowls with water of various shading, at that point the convergence of various tones will give us the item limits. This is the instinct behind the watershed calculation.

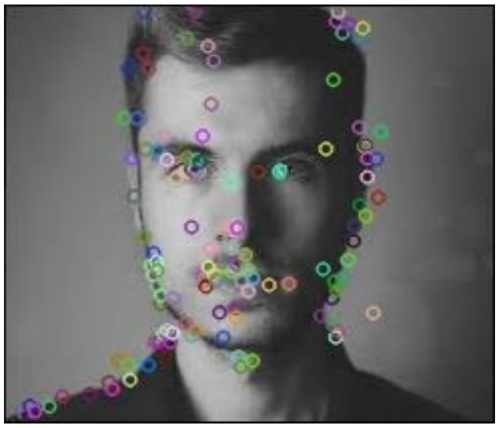
```
#importing required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt

#reading the image
image = cv2.imread('coins.jpg')
#converting image to grayscale format
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#apply thresholding
ret,thresh =
cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#get a kernel
kernel = np.ones((3,3),np.uint8)
opening =
cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel,iterations = 2)
#extract the background from image
sure_bg = cv2.dilate(opening,kernel,iterations = 3)
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret,sure_fg =
cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_bg)
ret,markers = cv2.connectedComponents(sure_fg)

markers = markers+1

markers[unknown==255] = 0
markers = cv2.watershed(image,markers)
image[markers==-1] = [255,0,0]
plt.imshow(sure_fg)
```

2.7. Speeded-Up Robust Features (SURF):

Speeded-Up Robust Features (SURF) is an improved type of SIFT. It works significantly snappier and is more incredible to picture changes. In SIFT, the scale space is approximated using Laplacian of Gaussian. Interruption – that sounds unreasonably perplexing. What is Laplacian of Gaussian?

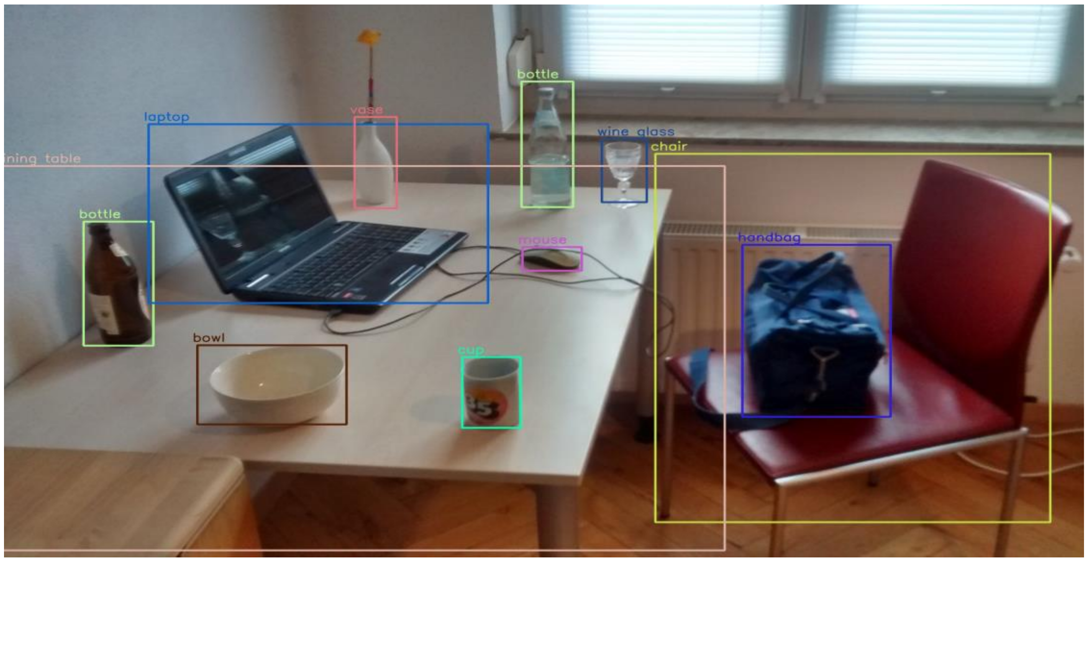
Laplacian is a bit used for figuring the edges in an image. The Laplacian bit works by approximating a second subordinate of the image. Subsequently, it is extraordinarily tricky to uproar. We generally apply the Gaussian piece to the image before Laplacian partition as such giving it the name Laplacian of Gaussian.

```
#import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#show OpenCV version
print(cv2.__version__)
#read the iamge and convert to grayscale
image = cv2.imread('index.png')
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#create sift object
sift  = cv2.xfeatures2d.SIFT_create()
#calculate keypoints and their orientation
keypoints,descriptors = sift.detectAndCompute(gray,None)
#plot keypoints on the image
with_keypoints = cv2.drawKeypoints(gray,keypoints)
#plot the image
plt.imshow(with_keypoints)
```



3. **Object Detection:**

This module incorporates recognition of articles. For instance, eyes, faces[10], individuals, etc. Item recognition is a PC vision strategy that permits us to recognize and find objects in a picture or video. With this sort of recognizable proof and confinement, object recognition can be utilized to include objects in a scene and decide and track their exact areas, all while precisely naming them.



3.1. Video Analysis:

A Video examination module incorporates movement assessment, foundation and article following calculations.

Video content examination (likewise video content investigation, VCA) or video examination (VA) is the ability of naturally breaking down video to identify and decide fleeting and spatial occasions.

This specialized ability is utilized in a wide scope of areas including entertainment, video recovery and video perusing, medical services, retail, car, transport, home mechanization, fire and smoke location, wellbeing and security [11]. The calculations can be executed as programming on broadly useful machines, or as equipment in specific video handling units.

3.2. Core Functionality:

This module defines basic data structures and basic functions used by all other modules.

3.2.1. Camera Calibration and 3D Re-construction:

This module includes basic multiple-view geometry algorithms, object pose estimations and 3D Reconstruction elements.

3.3. Video I/O:

This module is an easy-to-use interface to video capturing.

These are the couple of modules which are predominantly utilized.
Open CV uses special cases for signal basic mistakes. At the point when the info information has a right configuration and has a place with the predetermined worth reach, however the calculation can't prevail for reasons unknown, it restores an uncommon mistake code.

Open CV de-distributes, just as dispenses the memory consequently for yield work boundaries more often than not. Along these lines, if a capacity has at least one information exhibits and some yield clusters, the yield exhibits are naturally apportioned or redistributed. Open CV is written in python and its essential interface is in python.

Open CV sudden spikes in demand for the accompanying working frameworks Windows, Linux, MacOS. The size and kind of the yield clusters are resolved from the size and sort of information exhibits.

3.4. OpenCV - Storing Images:

To catch a picture, we use gadgets like cameras and scanners. These gadgets record mathematical estimations of the picture (Ex: pixel esteems). OpenCV is a library which measures the computerized pictures, in this manner we need to store these pictures for handling.

The Mat class of OpenCV library is utilized to store the estimations of a picture [12]. It speaks to a n-dimensional cluster and is utilized to store picture information of grayscale or shading pictures, voxel volumes, vector fields, point mists, tensors, histograms, and so on

This class comprises of two data parts: the **header** and a **pointer**

- **Header** − Contains information like size, method used for storing, and the address of the matrix (constant in size).
- **Pointer** − Stores the pixel values of the image (Keeps on varying).

3.5. Creating and Displaying the Matrix:

In this segment, we will talk about our first OpenCV model. We will perceive how to make and show a basic OpenCV lattice.
Given underneath are the means to be followed to make and show a framework in OpenCV.

Step 1: Load the OpenCV native library

While composing Java code utilizing OpenCV library, the initial step you need to do is to stack the local library of OpenCV utilizing the loadLibrary (). Burden the OpenCV local library as demonstrated as follows.

```
//Loading the core library
System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
```

Step 2: Instantiate the Mat class

Instantiate the Mat class using any of the functions mentioned in this chapter earlier.

```
//Creating a matrix
Mat matrix = new Mat (5, 5, CvType.CV_8UC1, new Scalar (0));
```

Step 3: Fill the matrix using the methods

You can retrieve particular rows/columns of a matrix by passing index values to the methods **row ()/col ()**.

And, you can set values to these using any of the variants of the **setTo()** methods.

On executing the above program, you will get the following output −

```
OpenCV Mat data:
[  1,   1,   1,   3,   1;
   0,   0,   0,   3,   0;
   0,   0,   0,   3,   0;
   0,   0,   0,   3,   0;
   0,   0,   0,   3,   0]
```

3.6. Loading Image:

The Buffered Image class of the java.awt.image.BufferedImage bundle is utilized to store a picture and the ImageIO class of the bundle import javax.imageio gives techniques to peruse and compose Images.

4. Automatic Memory Management:

Open CV handles all the memory subsequently. Regardless of anything else, std::vector, cv::Mat and other data structures used by the limits and systems have destructors that deallocate the shrouded memory upholds when required. This infers that the destructors don't for the most part deallocate the backings as though there ought to emerge an event of Mat. They consider possible data sharing. A destructor decrements the reference counter associated with the system data support. The support is deallocated if and just if the reference counter shows up at zero, that is, where the same structures imply a comparative pad. Additionally, when a Mat event is copied, no genuine data is really imitated. Taking everything into account, the reference counter is expanded to hold that there is another owner of a comparative data. There is more over the Mat::clone procedure that makes a full copy of the system data.

4.1. Automatic Allocation of the Output Data:

Open CV deallocates the memory naturally, just as consequently designates the memory for yield work boundaries more often than not. Along these lines, if a capacity has at least one information clusters and some yield exhibits, the yield clusters are consequently dispensed or redistributed. The size and kind of the yield clusters are resolved from the size and sort of information exhibits. If necessary, the capacities take additional boundaries that help to sort out the yield exhibit properties

The bunch diagram is normally dispersed by the >> overseer since the video layout objective and the spot significance is known to the video getting module. The bunch edges is subsequently assigned by the cut Color work. It has a comparative size and the spot significance as the data bunch. The amount of channels is 1 in light of the fact that the concealing change cv:: COLOR_BGR2GRAY is passed, which infers a concealing to grayscale change. Note that edge and edges are alloted simply a solitary time during the principle execution of the hover body since all the accompanying video traces have a comparative objective. If you somehow change the video objective, the bunches are normally rearranged.

The vital segment of this innovation is the Mat::create technique. It takes the ideal cluster size and type. In the event that the cluster as of now has the predetermined size and type, the strategy sits idle. Else, it delivers the recently distributed information, assuming any (this part includes decrementing the reference counter and contrasting it and zero), and afterward dispenses another cushion of the necessary size. Most capacities call the Mat::create technique for each yield cluster, thus the programmed yield information assignment is actualized.

Some eminent special cases from this plan are cv::mix Channels, cv::RNG::fill, and a couple of different capacities and techniques. They can't dispense the yield exhibit, so you need to do this ahead of time .

4.2. Multi-threading and Re-enterability:

The current Open CV execution is completely re-enterable. That is, similar capacity or similar techniques for various class occasions can be called from various strings. Additionally, a similar Mat can be utilized in various strings on the grounds that the reference-tallying tasks utilize the design explicit nuclear guidelines.

5. Face Detection:

OpenCV upholds haar course-based item identification. Haar falls are AI based classifiers that compute various highlights like edges, lines, and so forth in the picture. At that point, these classifiers train utilizing various positive and negative examples.

Prepared classifiers for various articles like faces, eyes and so on are accessible in the OpenCV GitHub repo, you can likewise prepare your own haar course for any item.

5.1. Face Detection Use Case:

The fun doesn't stop there! Another cool thing we can do – create an all out use case around the above code. Additionally, you don't need to start without any planning. We can make a couple little changes to the code and we're good to go .

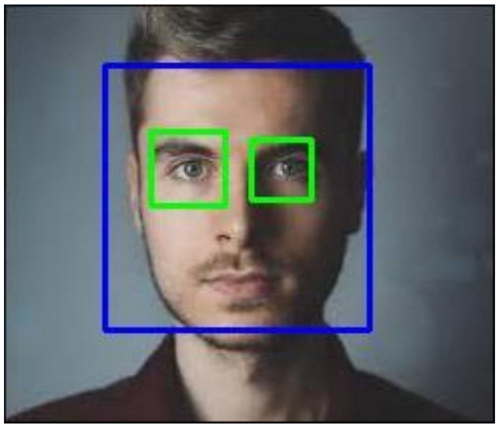
Assume, for instance, you need to assemble a robotized camera-based framework to follow where the speaker is continuously. As per his position, the framework turns the camera so the speaker is consistently in the centre of the video.

```
#import required libraries
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
%matplotlib inline

#load the classifiers downloaded
face_cascade = cv.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv.CascadeClassifier('haarcascade_eye.xml')
#read the image and convert to grayscale format
img = cv.imread('rotated_face.jpg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#calculate coordinates
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
for (x,y,w,h) in faces:
    cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    #draw bounding boxes around detected features
    for (ex,ey,ew,eh) in eyes:
        cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
#plot the image
plt.imshow(img)
#write image
cv2.imwrite('face_detection.jpg',img)
```

```
import cv2

import face_recognition
```

Then, read the video and get the length:

```
input_movie = cv2.VideoCapture("sample_video.mp4")

length = int(input_movie.get(cv2.CAP_PROP_FRAME_COUNT))
```



This is a base class for all more or less complex algorithms in OpenCV.

particularly for classes of calculations, for which there can be various usage. The models are sound system correspondence (for which there are calculations like square coordinating, semi-worldwide square coordinating, chart cut and so forth), foundation deduction (which should be possible utilizing combination of-gaussians models, codebook-based calculation and so on), optical stream (block coordinating, Lucas-Kanade, Horn-Schunck and so on)

6. D LIB:

Dlib is a toolkit for making real world machine learning and data analysis applications in C++. While the library is originally written in C++, it has good, easy to use Python bindings. I have majorly used **Dlib** for **face detection** and **facial** landmark **detection**.

Characteristics of DLIB:

- A digital library is a collection of texts, images, or data that has been digitized.
- It should have a way to navigate and retrieve information.
- It should have at least one specified audience

Detecting facial landmarks is therefore a two step process:

- **Step #1:** Localize the face in the image.
- **Step #2:** Detect the key facial structures on the face ROI.

Face location (Step #1) can be accomplished in various ways.

We could utilize OpenCV's implicit Haar falls.

We may apply a pre-prepared HOG + Linear SVM object identifier explicitly for the errand of face identification.

Or then again we may even utilize profound learning-based calculations for face confinement.

In one or the other case, the genuine calculation used to recognize the face in the picture doesn't make a difference. All things being equal, what's significant is that through some strategy we get the face jumping box (i.e., the (x, y)- directions of the face in the picture).Given the face region we can then apply

Step #2: detecting key facial structures in the face region.

There are a variety of facial achievement finders, yet all strategies fundamentally endeavor to restrict and check the going with facial regions: Mouth, Right eyebrow, Left eyebrow, Right eye, Left eye, Nose, Jaw

The facial milestone identifier remembered for the dlib library is an execution of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).

This method starts by using:

A preparing set of named facial tourist spots on a picture. These pictures are physically named, determining explicit (x, y)- directions of locales encompassing every facial structure.

Priors, of all the more explicitly, the likelihood on distance between sets of info pixels.

Given this readiness data, a gathering of backslide trees are set up to survey the facial achievement positions clearly from the pixel powers themselves (i.e., no "incorporate extraction" is happening).

The result is a facial achievement locater that can be used to recognize facial places of interest consistently with first rate gauges.

For more information and nuances on this specific technique, make sure to examine the paper by Kazemi and Sullivan associated with above, close by the authority dlib statement.

Installation:

Step 1: Install Visual Studio 2015.

Step 2: Install CMake v3.8.2.

Step 3: Install Anaconda 3.

Step 4: Download **Dlib**. Download **Dlib** v19.6 from http://**dlib**.net/files/**dlib**-19.6.zip.

Step 5: **Build Dlib library**. ...

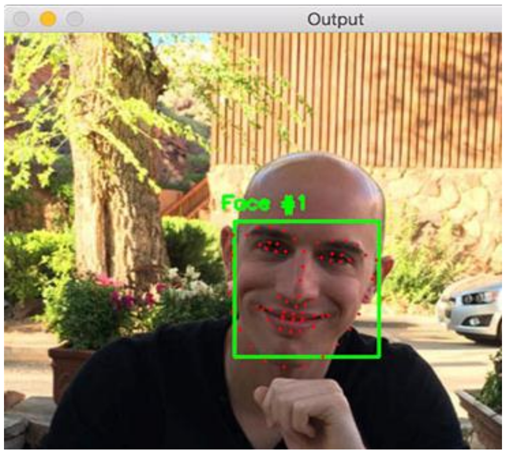
Step 6: Update user environment variable – dlib_DIR. ...

Step 7: **Build Dlib** examples. ...

Step 8: Test **Dlib**'s C++ example.

Facial landmarks with dlib, OpenCV, and Python

```
$ python facial_landmarks.py --shape-predictor shape_predictor_68_face_landmarks.dat \
--image images/example_01.jpg
```



Introduced underneath is a depiction of the Dlib limit's code, with the record centers contrasting with parts of the face.

Despite which dataset is used, the identical dlib structure can be used to set up a shape marker on the information planning data — this is significant in case you should get ready facial achievement locaters or custom shape pointers of your own.

In the excess of this blog entry, I'll exhibit how to identify these facial tourist spots in pictures.

Future blog entries in this arrangement will utilize these facial milestones to remove explicit locales of the face, apply face arrangement, and even form a squint location framework.

6.1. ALGORITHM:

```
import dlib
import cv2
face_detector = dlib.get_frontal_face_detector()
#face_detector = dlib.cnn_face_detection_model_v1("mmod_human_face_detector.dat")
def scale_faces(face_rects, down_scale=1.5):
faces = []
for face in face_rects:
scaled_face = dlib.rectangle(int(face.left() * down_scale),
int(face.top() * down_scale),
int(face.right() * down_scale),
int(face.bottom() * down_scale))
faces.append(scaled_face)
return faces
def detect_faces(image, down_scale=1.5):
image_scaled = cv2.resize(image, None, fx=1.0/down_scale, fy=1.0/down_scale,
interpolation=cv2.INTER_LINEAR)
faces = face_detector(image_scaled, 0)
# faces = [face.rect for face in faces]
faces = scale_faces(faces, down_scale)
return faces
```

6.2. CNN based face detector from dlib:

On the off chance that you are into such a picture preparing, PC vision or AI, odds are high that you may have run over/utilized dlib some place in your excursion.

As per dlib's github page, dlib is a toolbox for making true AI and information investigation applications in C++. While the library is initially written in C++, it has great, simple to utilize Python ties.

I have significantly utilized dlib for face location and facial milestone recognition. The frontal face identifier in dlib functions admirably. It is basic and just works out of the crate.

This locator depends on histogram of situated angles (HOG) and direct SVM. (Clarifying how this identifier functions is past the extent of this blog entry. Presumably a subject to examine for one more day)

While the HOG+SVM based face locator has been around for some time and has assembled a decent measure of clients, I don't know the number of us saw the CNN (Convolutional Neural Network) based face identifier accessible in dlib. Truly, I didn't. I inadvertently ran over it while perusing dlib's github archive.

In the event that you have ever utilized the HOG based face indicator in dlib, you presumably realize that it won't recognize faces at odd points. It is intended to be a decent "frontal" face finder and it is, in reality.

It identifies faces in any event, when they are not totally frontal to a decent expand. Which is truly useful for a frontal face indicator. In any case, you can unfortunately anticipate a limited amount of much from it.

In the interim, the CNN based identifier is equipped for recognizing faces practically in all points. Sadly, it isn't appropriate for ongoing video. It is intended to be executed on a GPU. To get a similar speed as the HOG based locator you may have to run on an incredible Nvidia GPU.

By the by, this ought not prevent us from giving it a shot actually pictures.

In the rest of this post, I am demonstrating how you can utilize the CNN put together face indicator from dlib with respect to pictures and contrast the outcomes and HOG based locator with prepared to utilize Python code.

6.3. Detecting faces at odd angles:

Could the CNN based indicator identify faces at odd points (read non-frontal) which the HOG based finder may neglect to distinguish?

Indeed, the appropriate response is "nearly". (I have not tried it thoroughly to give a sure "yes". However, to the extent I have tried, it is functioning admirably for non-frontal pictures).

How about we take a gander at a portion of the models where HOG based finder flops yet CNN can recognize.

Understanding deep learning face recognition embeddings:

All in all, how does profound learning + face acknowledgment work?

The mystery is a strategy called profound measurement learning.

On the off chance that you have any related knowledge with profound learning you realize that we regularly train an organization to:

Acknowledge a solitary info picture

What's more, yield a grouping/mark for that picture

Nonetheless, profound measurement learning is unique.

All things considered, of attempting to yield a solitary name (or even the directions/jumping box of items in a picture), we are rather yielding a genuine esteemed element vector.

For the dlib facial acknowledgment organization, the yield include vector is 128-d (i.e., a rundown of 128 genuine esteemed numbers) that is utilized to measure the face.

Here we give three pictures to the organization:

Two of these pictures are model countenances of a similar individual.

The third picture is an arbitrary face from our dataset and isn't similar individual as the other two pictures.

For instance, we should again consider Figure 1 above where we gave three pictures: one of Chad Smith and two of Will Ferrell.

Our organization measures the appearances, building the 128-d inserting (evaluation) for each.

From that point, the overall thought is that we'll change the loads of our neural organization so the 128-d estimations of the two Will Ferrel will be nearer to one another and farther from the estimations for Chad Smith.

At that point, we apply this square shape as a convolutional portion, over our entire picture. To be comprehensive, we should apply every single imaginable measurement and places of every part. A straightforward 24*24 pictures would normally result in over 160'000 highlights, each made of an entirety/deduction of pixels esteems. It would computationally be unimaginable for live face recognition. Along these lines, how would we accelerate this cycle ?

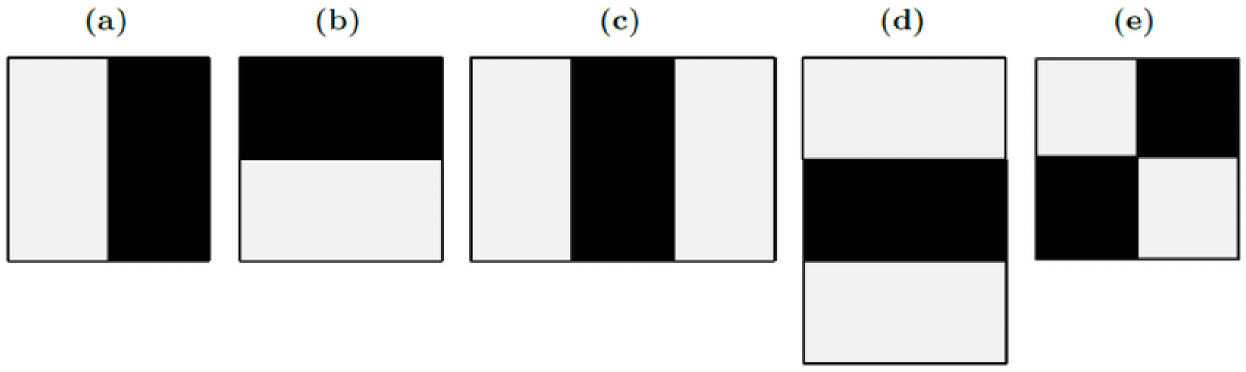
when the great district has been recognized by a square shape, it is pointless to run the window over a totally extraordinary area of the picture. This can be accomplished by Adaboost.

process the square shape highlights utilizing the fundamental picture standard, which is way quicker. We'll cover this in the following area.

There are a few kinds of square shapes that can be applied for Haar Features extraction. As indicated by the first paper :

- the two-square shape include is the distinction between the amount of the pixels inside two rectangular districts, utilized principally for distinguishing edges (a,b)
- the three-square shape include figures the whole inside two external square shapes deducted from the aggregate in a middle square shape, utilized fundamentally for identifying lines (c,d)

• the four-square shape highlight registers the contrast between corner to corner sets of square shape (e)



Since the highlights have been chosen, we apply them on the arrangement of preparing pictures utilizing Ad boost characterization, that joins a bunch of feeble classifiers to make a precise group model. With 200 highlights (rather than 160'000 at first), a precision of 95% is accomplished. The creators of the paper have chosen 6'000 highlights.

n spite of the fact that the cycle portrayed above is very effective, a significant issue remains. In a picture, a large portion of the picture is a non-face locale. Giving equivalent significance to every area of the picture looks bad, since we ought to primarily zero in on the areas that are well on the way to contain an image. Viola and Jones accomplished an expanded discovery rate while diminishing calculation time utilizing Cascading Classifiers.

The key thought is to dismiss sub-windows that don't contain faces while distinguishing areas that do. Since the errand is to recognize appropriately the face, we need to limit the bogus negative rate, i.e the sub-windows that contain a face and have not been distinguished all things considered.

Any negative outcome eventually prompts a dismissal of the sub-window as possibly containing a face. The underlying classifier dispenses with most negative models at a low computational expense, and the accompanying classifiers kill extra negative models yet require more computational exertion.

7. THEORITICAL ANALYSIS:

Saturation Arithmetic's:

As a PC vision library, Open CV game plans a ton with picture pixels that are routinely encoded in a decreased, 8-or 16-cycle per channel, outline and as needs be have a confined worth reach. Moreover, certain method on pictures, like concealing space changes, quality/contrast changes, sharpening, complex interposition (bi-cubic, Lanczos) can make esteems out of the available reach. In case you just store the most insignificant 8 (16) bits of the result, this results in visual relics and may impact a further picture assessment. To deal with this issue, the supposed submersion science is used. For example, to store r, the delayed consequence of a movement, to a 8-digit picture, you find the nearest motivator inside the 0 to 255 domain:

I(x,y)=min(max(round(r),0),255)

Close to principles are applied to 8-cycle stepped, 16-digit checked and unsigned sorts. This semantics is utilized any spot in the library. In code, it is finished utilizing the cv::saturate cast<> limits that take after standard cast works out. See under the execution of the equation gave previously:

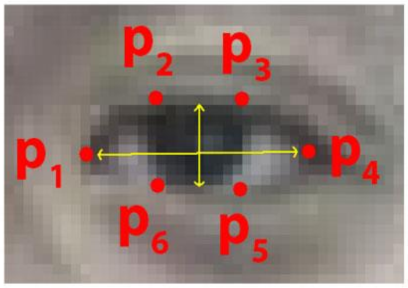
I.at<uchar>(y, x) = soak cast<uchar>(r);

where cv::uchar is an Open CV 8-cycle unsigned entire number sort. In the improved SIMD code, such SSE2 rules as paddusb, packuswb, and so on are used. They help achieve exactly the same lead as in code.

7.1. Eye Aspect Ratio:

In terms of blink detection we are interested in eye structures.

In this we are representing each eye by 6 (x , y) coordinates. These coordinates start from left corner of the eye to remaining portion of the eye in clockwise direction.



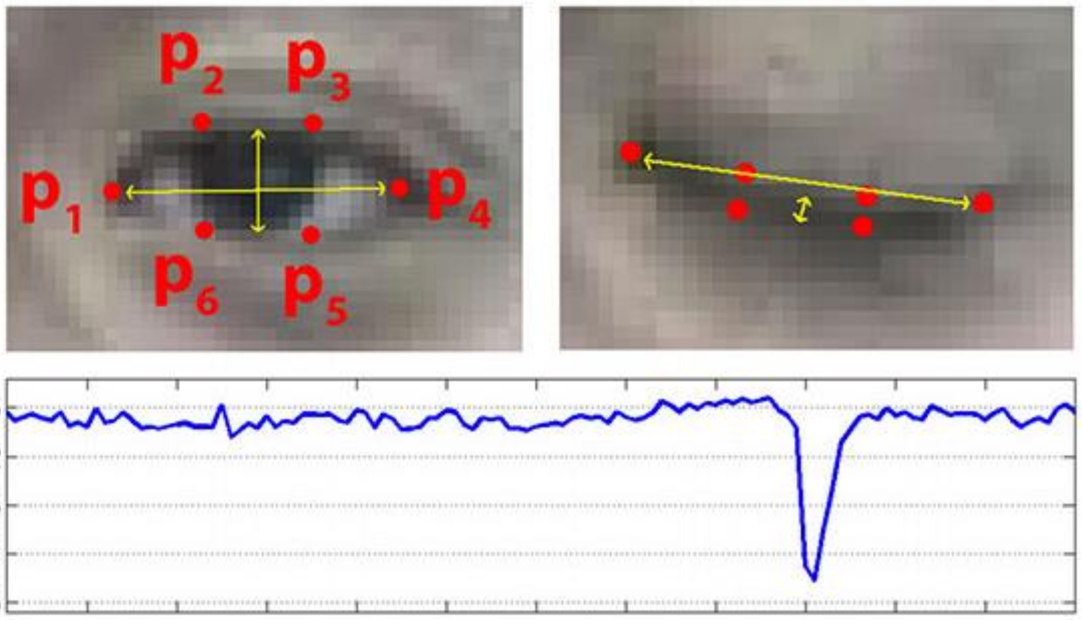


Figure 7: *Top-left:* A visualization of eye landmarks when then the eye is open. *Top-right:* Eye landmarks when the eye is closed. *Bottom:* Plotting the eye aspect ratio over time. The dip in the eye aspect ratio indicates a blink

Based on the image there is a relation between width and height of the coordinates.

There is an equation which derives Eye aspect ratio (EAR) :

$$EAR = \frac{\| p2-p6 \| + \| p3-p5 \|}{2\| p1-p4\|}$$

Where p1, p2, p3, p4, p5, p6 are facial landmark locations.

The numerator of this condition portrays the distance between the vertical eye milestones though the denominator of this condition depicts the distance between level eye landmrks. Subsequently we have one bunch of level focuses and two arrangements of vertical focuses.

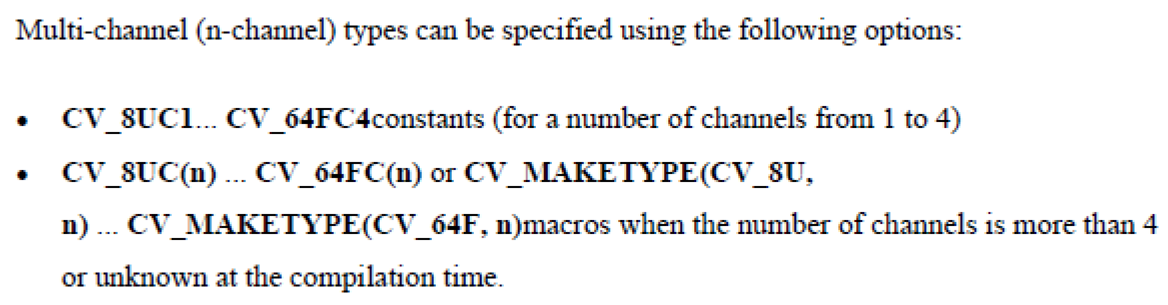
At the point when the eye is open the Eye perspective proportion is steady, however when it is flickering the eye viewpoint proportion will tumble to zero.

By using this condition we can evade picture taking care of methodology and we can depend upon eye achievement distances to find if an individual is glimmering.

A cluster whose components are such tuples, are called multi-channel exhibits, as inverse to the single-channel clusters, whose components are scalar qualities. The greatest conceivable number of channels is characterized by the CV_CN _MAX steady, which is at present set to 512.

For these basic types, the following enumeration is applied:

Enum { CV_8U=0, CV_8S=1, CV_16U=2, CV_16S=3, CV_32S=4, CV_32F=5, CV_64F=6 };



Multi-channel (n-channel) types can be specified using the following options:

- **CV_8UC1**... **CV_64FC4**constants (for a number of channels from 1 to 4)
- **CV_8UC(n)** ... **CV_64FC(n)** or **CV_MAKETYPE(CV_8U, n)** ... **CV_MAKETYPE(CV_64F, n)**macros when the number of channels is more than 4 or unknown at the compilation time.

7.2. Input Array and Output Array:

Many Open CV capacities measure thick 2-dimensional or multi-dimensional mathematical clusters. Generally, such capacities take cppMat as boundaries, however at times it's more advantageous to utilize std::vector<> (for a point set, for instance) or cv::Matx<> (for 3x3 homograph network and such). To maintain a strategic distance from numerous copies in the API, uncommon "intermediary" classes have been presented. The base "intermediary" class is cv::Input Array It is utilized for passing read-just clusters on a capacity input. The got from Input Array class cv::Output Array is utilized to determine a yield exhibit for a capacity. Regularly, you ought not mind of those middle of the road types - it will all work consequently. You can accept that rather than Input Array/Output Array you can generally utilize Mat,std::vector<>,cv::Max<>,cv::Vec<> or cv::Scalar. At the point when a capacity has a discretionary info or yield cluster, and you don't have or don't need one, pass cv::no Array().

7.3. Error Handling:

OpenCV utilizes special cases for signal basic blunders. At the point when the information has a right arrangement and has a place with the predetermined worth reach, however the calculation can't prevail for reasons unknown (for instance, the enhancement calculation didn't join), it restores an uncommon blunder code (ordinarily, simply a Boolean variable).

The special cases can be examples of the cv::Exception class or its subordinates. In its turn, cv::Exception is a subsidiary of std::exception. So, it tends to be nimbly dealt with in the code utilizing other standard library parts.

The exemption is commonly tossed either utilizing the CV_Error(errorcode,description) large scale, or its printf-like CV_Error_(errcode,(printf-spec,printf-args)) variation, or utilizing the CV_Assert(condition) full scale that checks the condition and tosses a special case when it isn't fulfilled. For execution basic code, there is CV_DbgAssert(condition) that is just held in the Debug arrangement. Because of the programmed memory the executives, all the middle supports are naturally deallocated if there should be an occurrence of an unexpected mistake. You just need to add an attempt explanation to get special cases, if necessary.

8. EXPERIMENTAL INVESTIGATION

code:

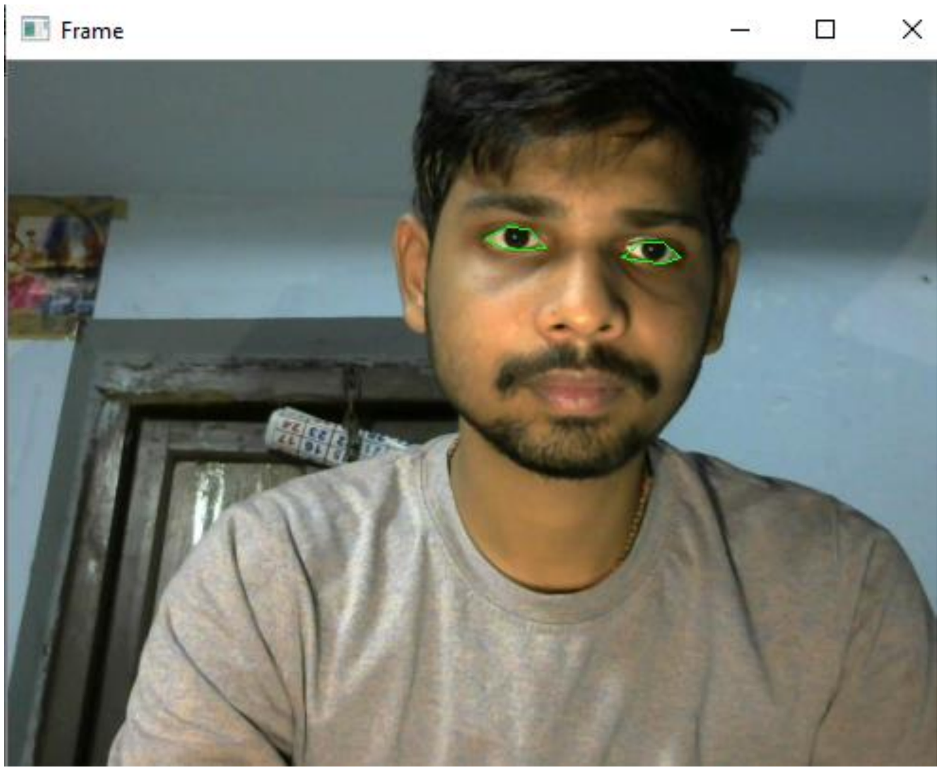
```
from scipy.spatial import distance
from imutils import face_utils
import imutils
import dlib
import cv2
from playsound import playsound
def eye_aspect_ratio(eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear
thresh = 0.25
frame_check = 20
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor(".\shape_predictor_68_face_landmarks.dat")  # Dat file is the crux
of the code
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
cap = cv2.VideoCapture(0)
flag = 0
while True:
from playsound import playsound
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])
```

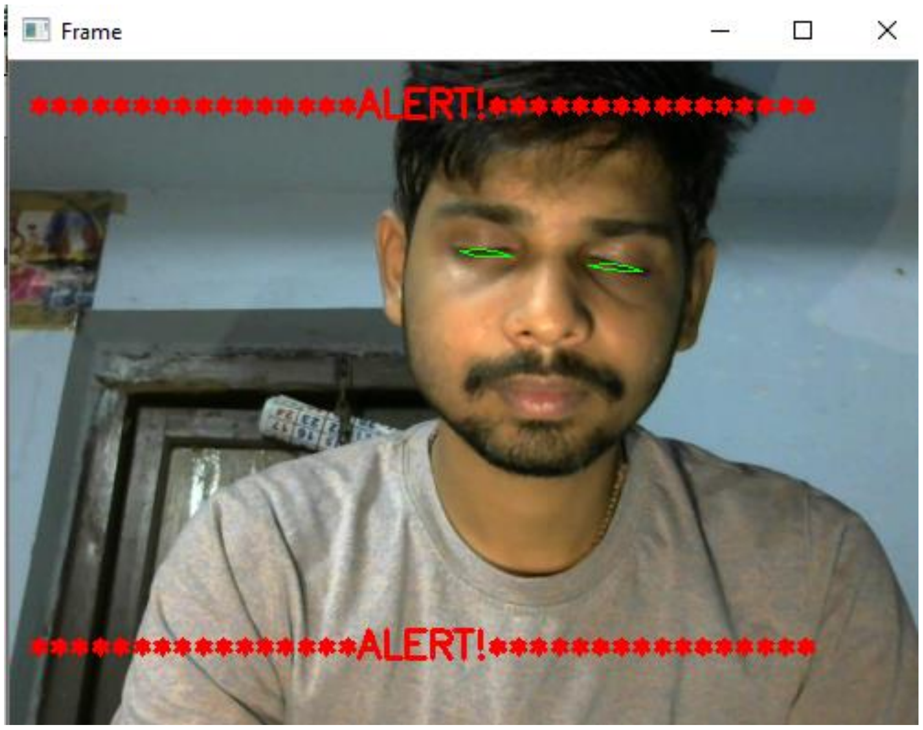
```
ear = (A + B) / (2.0 * C)
return ear
ret, frame = cap.read()
frame = imutils.resize(frame, width=500)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
subjects = detect(gray, 0)
for subject in subjects:
    shape = predict(gray, subject)
    shape = face_utils.shape_to_np(shape)  # converting to NumPy Array
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
    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)
    if ear < thresh:
        flag += 1
        print(flag)
        if flag >= frame_check:
            cv2.putText(frame, "****************ALERT!****************", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            cv2.putText(frame, "****************ALERT!****************", (10, 325),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            playsound('newalarm.mp3')
        # print ("Drowsy")
        else:
  flag = 0
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
for subject in subjects:
        shape = predict(gray, subject)
        shape = face_utils.shape_to_np(shape)  # converting to NumPy Array
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        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)
    if key == ord("q"):
        break
        if ear < thresh:
            flag += 1
            print(flag)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
cv2.destroyAllWindows()
cap.stop()
```

```
#import required libraries
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
%matplotlib inline

#load the classifiers downloaded
face_cascade = cv.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv.CascadeClassifier('haarcascade_eye.xml')
#read the image and convert to grayscale format
img = cv.imread('rotated_face.jpg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#calculate coordinates
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
for (x,y,w,h) in faces:
    cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    #draw bounding boxes around detected features
    for (ex,ey,ew,eh) in eyes:
        cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
#plot the image
plt.imshow(img)
#write image
cv2.imwrite('face_detection.jpg',img)
#importing required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

```
#reading the image
image = cv2.imread('coins.jpg')
#converting image to grayscale format
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#apply thresholding
ret,thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#get a kernel
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel,iterations = 2)
#extract the background from image
sure_bg = cv2.dilate(opening,kernel,iterations = 3)

dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret,sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)

sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_bg)

ret,markers = cv2.connectedComponents(sure_fg)

markers = markers+1

markers[unknown==255] = 0

markers = cv2.watershed(image,markers)
image[markers==-1] = [255,0,0]

plt.imshow(sure_fg)


import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

#reading images in grayscale format
image1 = cv2.imread('messi.jpg',0)
image2 = cv2.imread('team.jpg',0)

#finding out the keypoints and their descriptors
keypoints1,descriptors1 = cv2.detectAndCompute(image1,None)
keypoints2,descriptors2 = cv2.detectAndCompute(image2,None)

#matching the descriptors from both the images
bf = cv2.BFMatcher()
matches = bf.knnMatch(ds1,ds2,k = 2)


#selecting only the good features
good_matches = []
for m,n in matches:
   if m.distance < 0.75*n.distance:
       good.append([m])

image3 = cv2.drawMatchesKnn(image1,kp1,image2,kp2,good,flags = 2)
```

9. RESULTS:




Frame
*****************ALERT!*****************
*****************ALERT!*****************

10. CONCLUSION:

The drowsiness detection algorithm that takes live picture from the camera to identify the contrast between the open and shut eyes of the driver has been effectively actualized. To identify the laziness, at least three continuous open or shut eyes states are thought of. On the off chance that this happens, it means that driver is sluggish and afterward framework gives cautioning sign to the driver. Framework can recognize condition of the eyes with or without the normal glasses on. The framework may give incorrect outcomes in specific cases because of the impact of light, position of driver.the capacity to decide if a driver's eyes are appropriately open with an eye screen. This paper is proposing another technique recognizing the languid eyes to actualize the weariness driving identifying framework. Eyes are dynamic items and have profoundly inconstancy that is the reason recognizing eyes is truly troublesome. Here, we present another technique for distinguishing the sleepy eyes which is entirely reasonable for equipment execution and some different cycles.

11. REFERENCES:

[1] International Organization of Motor Vehicle Manufacturers. (2018). *Pro- visional Registrations or Sales of New Vehicles*. [Online]. Available: http://www.oica.net/wp-content/uploads/

[2] Wards Intelligence. (2018).*World Vehicles in Operation by Country, 2013_ 2017*. [Online]. Available: http://subscribers.wardsintelligence.com/databrowse- world

[3] National Highway Traf_c Safety Administration. (2018). *Traf_c Safety Facts 2016*. [Online]. Available: https://crashstats.nhtsa.dot.gov

[4] Y. Jiang, S. Guo, and S. Deng, ``Denoising and chaotic feature extraction of electrocardial signals for driver fatigue detection by Kolmogorov entropy," *J. Dyn. Syst., Meas. Control, Trans. ASME*, vol. 141, no. 2, 2019, Art. no. 0210131, doi: 10.1115/1.4041355.

[5] “Road Accidents in India 2016,” 2016. S. Sangle, B. Rathore, R. Rathod, A. Yadav, and A. Yadav, “Real Time Drowsiness Detection System,” pp. 87–92, 2018.

[6]V. Varghese, A. Shenoy, S. Ks, and K. P. Remya, “Ear Based Driver Drowsiness Detection System,” vol. 2018, pp. 93–96, 2018.

[7]A. Kumar and R. Patra, “Driver drowsiness monitoring system using visual behaviour and machine learning,” *ISCAIE 2018 - 2018 IEEE Symposium on Computer Applications and Industrial Electronics*, pp. 339–344, 2018.

[8]International Organization of Motor Vehicle Manufacturers, “Provisional registrations or sales of new vehicles,” http://www.oica.net/wpcontent/ uploads/, 2018.

[9] Wards Intelligence, “World vehicles in operation by country, 2013-2017,” http://subscribers.wardsintelligence.com/data-browse-world, 2018.

[10] Borghini Gianluca and Astolfi Laura et.al, “Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness,” NEUROSCIENCE AND BIOBEHAVIORAL REVIEWS, 2014.

[11] Attention Technologies, “S.a.m.g-3-steering attention monitor,” www.zzzzalert.com, 1999.

[12] Smart Eye, “Smarteye,” https://smarteye.se/, 2018.

[13] Henriques J F, Caseiro R, and Martins P et. al, “High-speed tracking with kernelized correlation filters,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.