iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** V **Month of publication:** May 2021

DOI: https://doi.org/10.22214/ijraset.2021.34586

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

Implementation of Drowsiness Detection and Alert System using Parameters as Drivers Head and Eyes

Prof. Ashwini Yerlekar[1], Siddhesh Pachkawade[2], Anjali Patil[3], Pratiksha Shende[4]
[1]*Guide,* [2, 3, 4]*Student, Department of Computer Science and Engineering, Rajiv Gandhi College of Engineering and Research, Wanadongri, Nagpur, INDIA - 441 110*

Abstract: ***Driver drowsiness is the momentous factor in a huge number of vehicle accidents. This driver drowsiness detection system has been valued highly and applied in various fields recently such as driver visual attention monitoring and driver activity tracking. Drowsiness can be detected through the driver face monitoring system. In this paper, a detailed review of driver drowsiness detection techniques implemented in the face camera has been reviewed. It combines off-the-shelf software components for face detection, human skin colour detection, and eye state (open vs. closed) classification in a novel way. Preliminary results show that the system is reliable and tolerant to many real-world constraints .An important application of machine vision and image processing could be driver drowsiness detection systems due to its high importance. In recent years there have been many research projects reported in the literature in this field. In this paper, unlike conventional drowsiness detection methods, which are based on the eye states alone, we used facial expressions to detect drowsiness. The review has also been focused on insight into recent and state-of-the-art techniques .And the most important thing is this paper helps others to decide better techniques for effective drowsiness detection. This paper describes the steps involved in designing and implementing a driver drowsiness detection system based on visual input (driver's face , head and eyes).***
Keyword: drowsiness, alert system, detection, dlib (Python library)

I. INTRODUCTION

Now road accidents have become a major concern. Everyday a lot of people are dying because of road accidents. Matter of fact, a large percentage of accidents occur due to inadvertent driving. It was found that one reason for driving inadvertently is driver fatigue and drowsiness. A lot of data revealed road accidents that were caused by driver fatigue and drowsiness. Drowsiness is a term that can be defined as feeling of sleepiness. Due to drowsiness, a driver may fall asleep while driving. Various techniques proposed in the literature to detect drowsiness. The driver face monitoring system is one of them. Driver face monitoring systems include imaging and intelligence software parts with the participation of various hardware parts. System overview that detects if a person is dry while driving and if so, alerts them using a voice alarm in real time. To identify if a person is asleep, if his eyes are closed, or if he is yawning, to recognize that he predicts eye spaces. Improvements in public safety and reduction of accidents are important goals of Intelligent Transportation Systems (ITS). One of the most important factors in accidents, especially on rural roads, is driver fatigue and monotony. Fatigue reduces the ability to control the vehicle and the ability to make decisions. Research shows that the driver is usually fatigued 1 hour after driving. In the early hours of the afternoon, after eating lunch and midnight, driver fatigue and drowsiness are much higher than at other times. In addition, alcohol, drug addiction, and use of hypnotic drugs can cause loss of consciousness.

II. LITERATURE REVIEW

According to "www.pyimagesearch.com/eye-blink-detection-with-opencv-python-dlib/" - author Adrian Rosebroke , the driver drowsiness system has been implemented by using library code present in the Python Coding environment.

Describe Road Accidents. Road accidents occur due to the driver not being noticed. In this paper we describe a real-time system for analyzing image sequences of a driver and determining the level of attention. For this purpose, we use the calculation of the percentage of eyelid closure. Closing the eye acts as an indicator to detect drowsiness. Driver fatigue and drowsiness are the major causes of traffic accidents on the road. It is very necessary to monitor the alertness level of the driver and to issue an alert when he / she is not paying enough attention on the road, to issue a promising way to reduce accidents caused by the driver's factors. Fatigue monitoring can be initiated with the removal of visual parameters. This can be done through a computer vision system. In the proposed work, we have robust methods for eye tracking under intentional lighting conditions and facial bending. It is based on the texture of the eyes. Visual information is obtained using a specially designed solution by combining a video camera with an IR illumination system. The system is fully automated and detects eye position and eye closure and corrects eye gaze. The experimental results using real images demonstrate the accuracy and robustness of the proposed solution.

It can become an important part in the development of advanced safety vehicles. People are scared about their future employment, livelihood and their future. Many proponents of autonomous, self-propelled vehicles argue that the first industry that will fully and completely overheat self-driving cars / trucks (even before consumer vehicles). Self-driving becomes a reality in the next few years. We have good reason to be concerned - drivers are out of jobs, one that he has been spending his entire life. They are also approaching retirement and need to strengthen their years of work. This is not speculation either: NVIDIA recently announced a partnership with PACCAR, a leading truck manufacturer. The goal of this partnership is to make self-driving vehicles a reality.

III. PROPOSED WORK

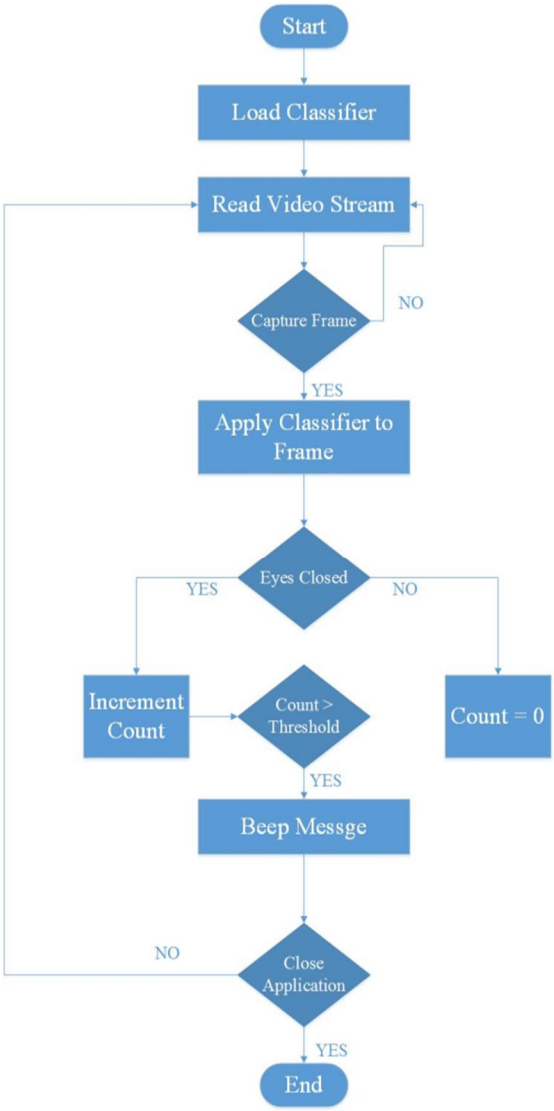


Fig1: working of the project

In this Python project, we will use OpenCV to assemble
Images from a webcam and feed them into a deep learning model that will classify whether a person's eyes are 'open' or 'closed'. The approach we will use for this Python project is as follows:

1) *Step 1* - Take an image as input from a camera.
2) *Step 2* - Locate the face in the image and create a region of interest (ROI).
3) *Step 3* - Locate the eye from the ROI and feed it to the classifier.
4) *Step 4* - The classifier will classify whether eyes are open or closed.
5) *Step 5* - Calculate the score to check if the person is drying up.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

IV. IMPLEMENTATION

Algorithm of drowsiness detection and warning system:

A. Take Image as Input from a Camera

With a webcam, we will take images as input. So to access the webcam, we created an infinite loop that would occupy each frame. We use the method provided by OpenCV, cv2.VideoCapture (0), to access the camera and set the capture object (cap). Cap .read () will read each frame and we will store the image in a frame variable.

B. Detect and Create face in the Image

1) *Area of Interest (ROI):* To detect faces in an image, we must first convert the image to grayscale because the OpenCV algorithm for object detection takes a gray image at the input. We do not need color information to detect objects. We will use every Cascade classifier to detect faces. This line is used to set our classifier face = cv2.CascadeClassifier (ade path to higher cascade xml file '). Then we find out using face = face.detectMultiScale (gray). This gives an array of constraints along with x, y coordinates, and height, the width of the object's border box. Now we can iterate over the faces and create a boundary box for each face. For (x, y, w, h) faces: cv2.rectangle (frame, (x, y), (x + w, y + h), (100,100,100), 1)

C. Locate the Eye with ROI and Feed It

1) *To Classify:* The same procedure is used to detect the eyes, to detect the face. First, we set the cascade classifier for the eyes in L-I and R-I, respectively, then detect the eyes using left_ye = LiAttectMulticel (gray). Now we only need to extract the eye data from the full image. This can be achieved by taking out the border box of the eye and then we can extract the eye image from the frame with this code. L _eye = frame [y: y + h, x: x + w] l_eye contains image data of the eye only. It will be fed into the classifier which will predict whether the eyes are open or closed. Similarly, we are removing the right eye in r_eye.

D. Whether or not the Classifier will Categorize

1) *Eyes open or Closed:* We are using a classifier to predict eye position. To feed our image into the model, we need to perform some operations because the correct dimensions are needed to initialize the model. First, we convert the color image to grayscale using r_eye = cv2.cvtColor (r_eye, cv2.COLOR_BGR2GRAY). Then, we resize the image to 24 * 24 pixels because our model was trained on a 24 * 24 pixel image cv2.resize (r_eye, (24,24)). We normalize our data for better convergence r_eye = r_eye / 255 (all values will be between 0–1). Expand the dimensions to feed into our classification. We loaded our model using model = load_model (c model / cnnCat2.h5 '). Now we predict each eye with our model lpred = model.predict_classes (l_eye). If lpred is a value of [0] = 1, it indicates that the eyes are open, if the value of lpred is [0] = 0, it indicates that the eyes are closed.

E. Calculate the Score to Check Whether

1) *The Person is Courageous:* The score is basically a value we will use to determine how long a person has closed their eyes. So if both eyes are closed, we will keep. When we raise the score and when the eyes are open, we reduce the score. We are using the result cv2.putText () function on the screen which will display the real-time status of the person. cv2.putText (frame, "open", (10, height-20), font, 1, (255,255,255), 1, cv2.LINE_AA) A range is defined for example if the score is greater than 15 which means That the person's eyes remained closed for a long time. This is when we beep the alarm using sound.play ().

F. How are Facial Landmark Indices for Facial Areas?

The face landmark detector implemented inside Dlib maps 68 (x, y) -coordinates that map to specific facial structures. These 68 point mappings were obtained by training a size prescriptor on the label iBUG 300-W dataset. Examining the image, we can see that the facial regions can be accessed through simple Python indexing.

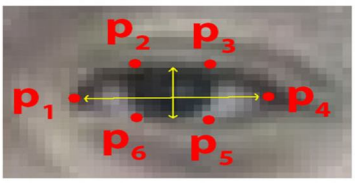


Fig2: mapping on the eyes

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

G. Are Encoded Inside the Mapping

FACIAL_LANDMARKS_IDXS

Dictionary inside the face_utils of the Emutils library:

FACIAL_LANDMARKS_IDXS = ordered ([

("Mouth", (48, 68)),

("right_eyebrow", (17, 22)),

("Left_eyebrow", (22, 27)),

("right_eye", (36, 42)),

("Left_eye", (42, 48)),

("Nose", (27, 35)),

("Jaw", (0, 17))

Using this dictionary we can easily index faces in landscape arrays and easily extract various features of the face by supplying a string as a key.

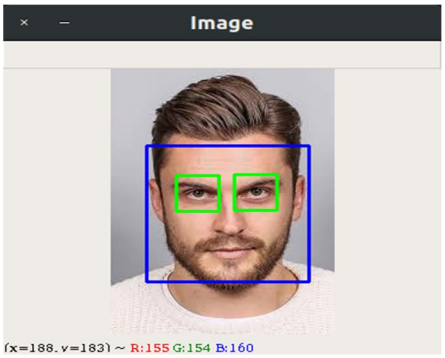


Fig3: region of area(ROA);facial landmarks

H. Visualizing Facial Locations with OpenCV and Python

It is a bit harder to visualize each of these faces and overlay the results on an input image.

To accomplish this, we'll need visualize_facial_landmark

Functions, already included in the imutils library. The visualize_facial_landmarks function requires two arguments, followed by two optional ones, each detailed below:

1) Image: The image on which we are going to create a landmark view of our face.

2) Size: NumPy array in a landmark with 68 faces coordinates that map for different face parts.

3) Color: A list of BGR tuples that are used to color-code each of the facial landmark areas.

4) Alpha: A parameter that is used to control the opacity of the overlay on the original image.

If the list of colors is "none", and if so, it starts with a predetermined list of BGR tuples (remember, OpenCV stores the color / pixel intensity in BGR order instead of RGB).

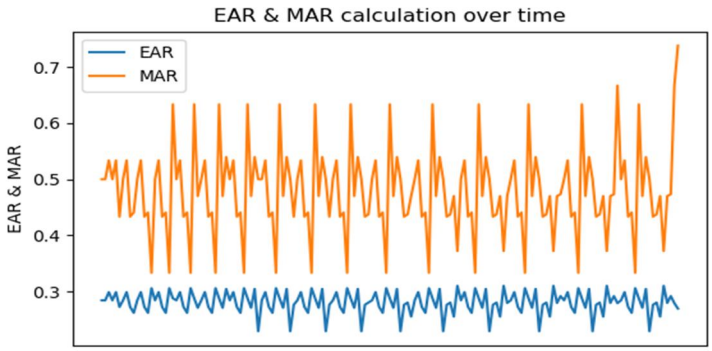


Fig4: graph of relation between EAR and MAR

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

I. Real-time Facial Recognition with OpenCV, Python, and dlib

We are also using the ImageStream implementation inside Emutil, allowing you to access your webcam / USB camera module in a more efficient, faster, commercial way. The script requires a command line argument, followed by a second optional one, below each elaborate:

Size-Indicator: The route to a landmark detector with pre-trained faces of dlib.

Now that our command line arguments are over, we have to initialize dlib's HOG + linear SVM-based face detector and then load the face landmark predictor from the disk:

(Detector = dlib.get_frontal_face_detector ()

Predictor = dlib.shape_predictor (args ["shape_predictor"])

Time_sleep () allows us to initialize our ImageStream and heat the camera sensor. The heart of our video processing pipeline can be found while inside the loop, and then grabs the next frame from our video stream. Then preprocesses to precise this frame. 400 pixel width and convert it to grayscale.

Before we can locate the face locations in our frame, we must first localize the face - this is accomplished through the detector at line 40 which is the bounding box (x, y) for each face in the image - Instructs.

Now that we have detected the faces in the video stream, the next step is to apply the facial landmark predictor for each face ROI. It applies the facial landmark detector to the face field, returning a shape object that we convert to a NumPy array then creating a series of circles on the output frame, visualizing each of the facial landmarks. Does. To understand the area of the face (ie, eyes, mouth, etc.) each (x, y) to display the output-frame on our screen. If

The Q key is pressed, we break through the loop and stop the script.

J. Eye Blink Detection

In the first part we will discuss the aspect ratio of the eye and how it can be used to determine whether a person is blinking in a video frame. From there, we will write Python, OpenCV and dlib code to (1) face landmark detection and (2) detect blinks in the image stream. Based on this implementation, we will implement our method to detect blinks in the example webcam stream with video files. We can locate facial landmarks to locate important areas of the face, including the eye, eyebrow, nose, ears, and mouth. This also implies that we can extract specific facial structures by knowing the index of the specific facial part. In the context of eyelid detection, we are only interested in two sets of facial structures - the eyes.

Each eye is represented by 6 (x, y)-coordinates, starting at the left corner of the eye (as if you were looking at the person), and then working clockwise around the rest of the field: the basis of this image But, we should take away the main point:

There is a relationship between the width and height of these coordinates.

Sokupova in his 2016 paper:

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

Equation1: The eye aspect ratio equation.

Where p1,…, p6 are 2D facial landmark locations.

The numerator of this equation calculates the distance between vertical eye sites, while each horizontal calculates the distance between horizontal spots, weighting each appropriately because there is only one set of horizontal points but no vertical points. There are two sets.

1) Why is this Equation so Interesting?

Well, as we find out, the aspect ratio of the eye is almost constant while the eye is open, but will rapidly decrease to zero when the eyelid blinks.

Using this simple equation, we can avoid image processing techniques and simply rely on the ratio of the eye's landmark distance to determine whether the person is blinking an eyelid.

The eye aspect ratio is constant, then rapidly falls to near zero, increasing again, indicating that a single blink has occurred.

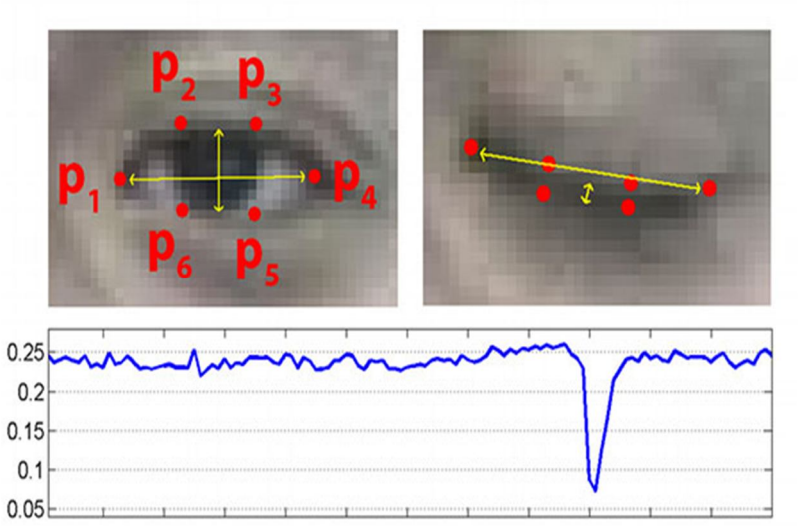


Fig5: facial landmark location and EAR graph

K. Improve Measured for Ear

Using the eye aspect ratio as a quantitative metric to determine if a person has blinked in a video stream.

However, due to noise in a video stream, sub facial landmark detection, or rapid changes in the viewing angle, a simple limit on the aspect ratio of the eye can produce a false-positive detection, reporting that the person is in reality as soon as the blink of an eye. I do not blink.

To make our blink detector more robust to these challenges, Soukupova and reportech recommend:

Calculate the eye aspect ratio for the N-th frame, along with the eye aspect ratios for the N - 6 frame and N + 6 frame, then compress these eye aspect ratios to create a 13 dimensional feature vector.

Training of support vector machines (SVM) on these feature vectors. Soukupova and reportech report that the combination of temporal-based feature vector and SVM classifier helps reduce false-positive blink detection and improve the overall accuracy of the blink detector. Open), then rapidly drops to zero, rising again, indicating that a nap has occurred. In our drowsiness detector case, we will monitor the eye aspect ratio to see if the value falls, but does not increase again, this means that the person has closed their eyes. Our boring detector requires a command line logic followed by two optional ones, each of which is detailed below. Alarm: Here you can optionally specify the path of an input audio file that can be used as an alarm. Webcam: This integer controls the index of your built-in webcam / USB camera. We can then visualize each eye region on our frame using the cv2.drawContours function below - this is often helpful when we are trying to debug our script and want to make sure that The eyes can be accurately detected and localized. Check to see if the aspect ratio of the eye is below the "eyelid / closed" eye boundary

EYE _AR _THRESH. If it is, we increase COUNTER, the total number of consecutive frames where the person has closed their eyes. If COUNTER exceeds EYE_AR_CONSEC_FRAMES, we assume that the person is starting to shut down.

Another check is done, this time to see if the alarm is on - if it is not, we turn it on. The arg alarm handle plays an alarm sound, provided the --alarm path is supplied after the script is executed. We take special care to create a separate thread responsible for calling the sound alarm to ensure that our main event is not blocked until the sound is over. Cv2.put_text draws text ALERT! On our frame- again, this is often helpful for debugging, especially if you are not using the Play Sound library. Finally, it handles the case where the eye aspect ratio is larger than EYE_AR_THRESH, indicating that the eyes are open. If the eyes are open, we reset the COUNTER and ensure that the alarm is off. The last code block in our drowsiness detector displays the output frame on our screen. Also, to keep the evidence, we have saved the frames in which the person was drowning.

V. APPLICATIONS

Driver drowsiness / fatigue is an important cause of combination-unit truck accidents. Recent analysis of the problem estimates that 15% - 36% of all accidents are fatal to combination-unit-truck drivers with associated drowsiness. The cost of these accidents is estimated to be $ 2,060 per vehicle over the lifetime of a combination-unit truck.

Drowsy driver detection methods can form the basis of a system that can potentially reduce the number of decreasing driving-related accidents. Recently, significant progress has been made in the development and application of a real-time drowsiness monitor. The monitor employs a novel dual-image video processing technique, measuring PERCLOS, a scientifically valid measure of drowsiness. Uses for PERCLOS monitors include:

A. Providing real-time feedback to the driver,
B. Providing performance feedback for a fatigue management program, and / or
C. Providing regulatory compliance information to enforcement officers.

VI. ADVANTAGES

A. Detects drowsiness.
B. Decreasing road accidents.
C. System implemented without using database storage.
D. No wires, cameras ,monitor or other devices are to be attached or aimed at the driver.
E. Due to the non intrusive nature of these methods they are more practically applicable.

VII. RESULT & CONCLUSION

If the person is detected to be in the drowsy state then the system algorithm will show the following results,if not then it will start the process of image capturing again.

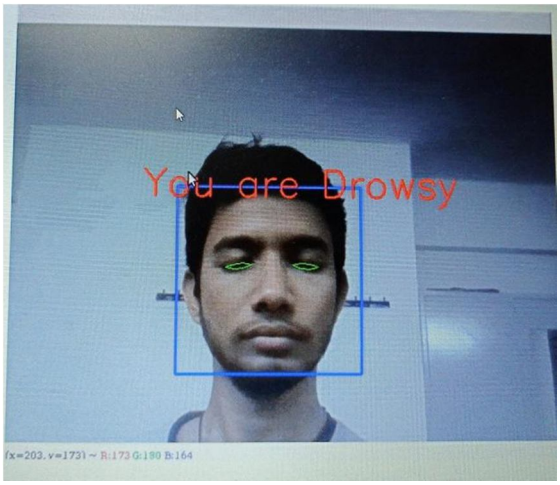
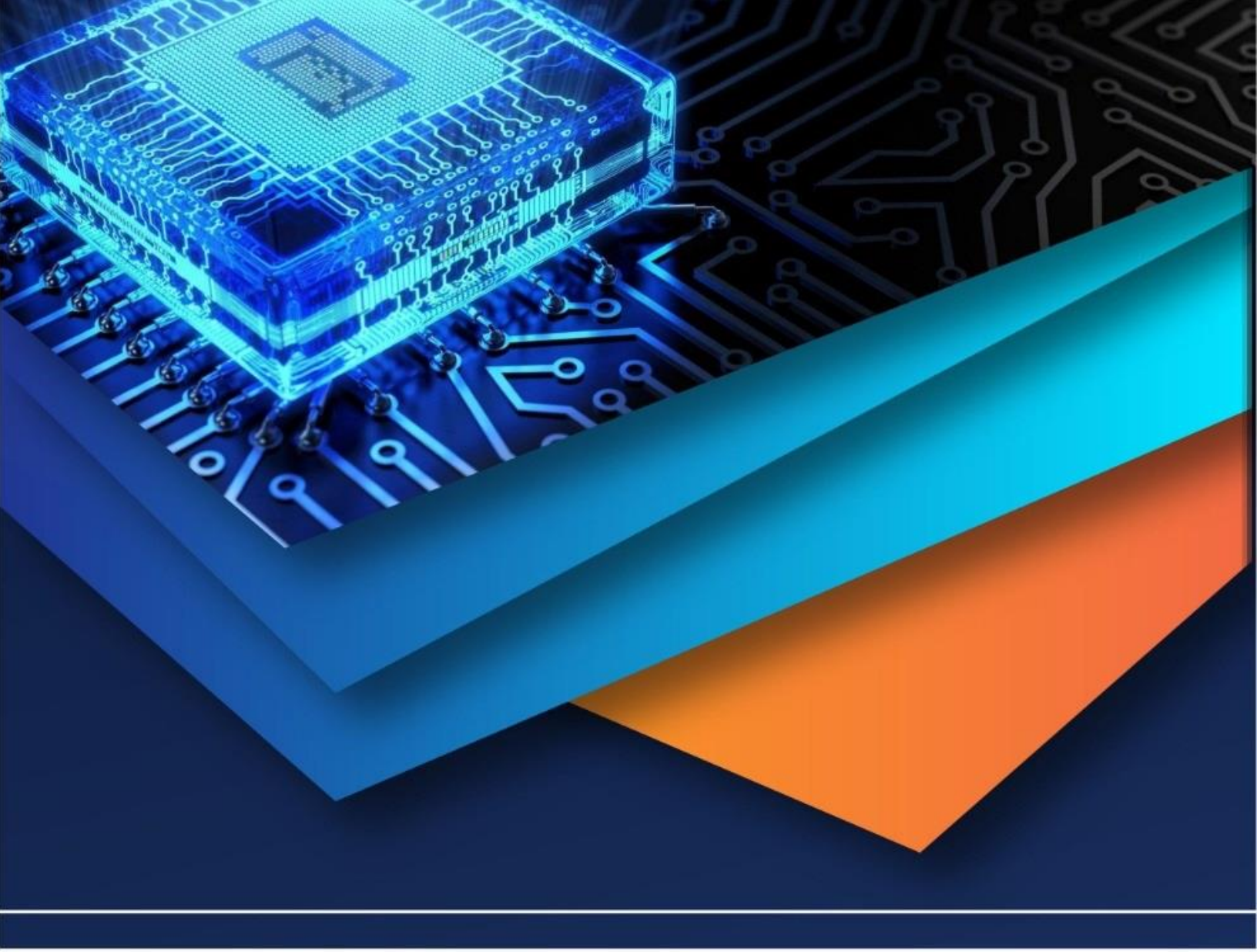


Fig6: “you are drowsy”-the person is in drowsy state

REFERENCES

[1] Drivers are falling asleep behind the wheels. Prevalence of drowsy driving crashes: https://www.nsc.org/road-safety/safety-topics/fatigued-driving
[2] Facial landmarks with dlib, OpenCV and Python: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/
[3] Eye blink detection with OpenCV, Python, and dlib: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
[4] Drowsiness Detection with OpenCV: https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/
[5] Real-Time Eye Blink Detection using Facial Landmarks: http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf
[6] Mitharwal Surendra Singh L., Ajgar Bhavana G., Shinde Pooja S., Maske Ashish M. “Eye Tracking Based Driver Drowsiness Monitoring & Warning System”.
[7] “Development of drowsiness detection system” Vehicle Research Laboratory, Nissan Center NISSAN MOTOR Co., Ltd. Hiroshi Ueno Masayuki Kaneda Masataka Tsukino
[8] Ingre, M.; ÅKerstedt, T.; Peters, B.; Anund, A.; Kecklund, G. Subjective sleepiness, simulated driving performance and blink duration: Examining individual differences. J. Sleep Res. 2006, 15, 47–53.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY