“ Driver Drowsiness Detection ”

Mini Project Report

Submitted in partial fulfillment of the requirements of the subject Minor Project

by

Anirudh Bhattacharya
Saumya Shah
Mink Shethia

Supervisor

Prof. Mamta Borle

Department of Computer Engineering

K.J. Somaiya Institute of Engineering and Information Technology

Ayurvihar, Sion Mumbai-400022

2021-22

This is to certify that the project entitled ***"DRIVER DROWSINESS SYSTEM"*** *is a bona fide work of Anirudh Bhattacharya, Saumya Shah and Mink Shethia submitted as mini project in the subject of* ***Minor Project*** *in* ***"Computer Engineering".***

_____

Prof. Mamta Borle

(Project Guide)

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. we also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. we understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Anirudh Bhattacharya _____

Saumya Shah _____

Mink Shethia _____

Date:

ACKNOWLEDGEMENT

Before presenting our seminar work entitled "**DRIVER DROWSINESS SYSTEM**" we would like to convey our sincere thanks to many people who guided us throughout the course for this seminar work. First, we would like to express our sincere thanks to our beloved Principal **DR. SURESH K. UKARANDE** for providing various facilities to carry out this report.

We would like to express our sincere thanks to **Prof. Mamta Borle** for her guidance, encouragement, co-operation and suggestions given to us at progressing stages of the report.

Finally, we would like to thank our **H.O.D. Dr. SARITA AMBADEKAR** and all teaching, non-teaching staff of the college and friends for their moral support rendered during the course of the report work and for their direct and indirect involvement in the completion of our report work, which made our endeavor fruitful.

Anirudh Bhattacharya

Saumya Shah

Mink Shethia

ABSTRACT

About 40% of car accidents on highways occur due to lack of sleep or drowsiness. This is a major cause for concern on busy highways with heavy vehicles driving at high speeds.

Thus, our project aims to detect the drowsiness level of a driver in a vehicle by identifying their facial features and using those to detect whether they're capable of driving or not.

This system will be most useful to law enforcement agencies and can be equipped on traffic signals or signboards with a powerful camera or it can also be mounted to the dashboard of a car by the manufacturer to help reduce the possibility of accidents.

INDEX

REFERENCES

CHAPTER 1
INTRODUCTION

1.1 Introduction:

Driver Drowsiness Detection system is a local application that aims to detect the drowsiness level of a driver and whether or not they're capable of driving their vehicle at the time.

This project aims to help law enforcement agencies and car manufacturers reduce the probability of accidents by warning the driver about their condition.

This project makes use of the Open Computer Vision library to detect the driver's facial features and analyze them. Machine learning libraries Keras and Tensorflow were also used to train the models.

1.2 Problem Introduction:

Driver Drowsiness System helps detect the level of fatigue in a vehicle operator.

Objective:-

1. Detects facial features of a driver.
2. Warn the driver if they're not in suitable physical condition to drive the vehicle.
3. Help law enforcement agencies prevent accidents on the road.
4. Help car manufacturers keep the vehicle and its passengers stay safe inside the vehicle.
5. Prevent possibly fatal mishaps and save lives.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 INTRODUCTION:

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

2.2 HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

HARDWARE REQUIREMENTS FOR PRESENT PROJECT:

PROCESSOR : Intel Pentium dual core or above.

RAM : 2 GB

HARD DISK : 160 GB

Integrated Webcam / External Camera

2.3 SOFTWARE REQUIREMENTS:

Software Requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

SOFTWARE REQUIREMENTS FOR OUR PROJECT:

OPERATING SYSTEM : Windows XP and above, Ubuntu v12.04 and above.

FOR DETECTION : Python 3.0 or above, OpenCV

FOR ANALYSIS: Keras, Matplotlib, Numpy

CHAPTER 3

ANALYSIS

3.1 PROPOSED SYSTEM:

Driver Drowsiness system can be implemented inside the vehicle or on traffic signals or signboards to detect the level of fatigue in the driver and prevent accidents.

3.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

3.2.1 Economic Feasibility

The project is aimed to be used mainly by Law Enforcement Agencies and Car Manufacturers and will require an integrated webcam or an external camera to be connected with the device on which the system will be running.

3.2.2 Technical Feasibility

The technical feasibility assessment meets with the expected needs of the proposed system. It has evaluated that hardware and software meets the need of the proposed system. The assessment based on the project of online testing consist of an interactive interface between student and teachers reveals the following outline design of system requirements:

->Python 3.0

->Matplotlib

->Numpy

->OpenCV

->Keras

->AJAX

To deal with requirements to handle completion of the project we are having strong resource of knowledge over the required technologies among our group members. Furthermore, these technologies are being thought in depth in WT tutorials to overcome any of the difficulties.

Also the technologies required are economically and legally feasible for implementation purpose.

3.2.3 Operational Feasibility

Driver Drowsiness Detection system to detect whether a vehicle's operator is physically capable of driving or not. Online pictures are also provided so that practically the users will understand what is the actual product.

3.3 SOFTWARE SPECIFICATION

OPENCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows,
Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A
full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The

document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposed to the C-based OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases)

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

KERAS:

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Keras is:

- Simple -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.

- Flexible -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

TensorFlow 2 is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

Guiding principles

- Modularity. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.
- Minimalism. Each module should be kept short and simple. Every piece of code should be transparent upon first reading. No black magic: it hurts iteration speed and ability to innovate.
- Easy extensibility. New modules are dead simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
- Work with Python. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

CHAPTER 4

IMPLEMENTATION

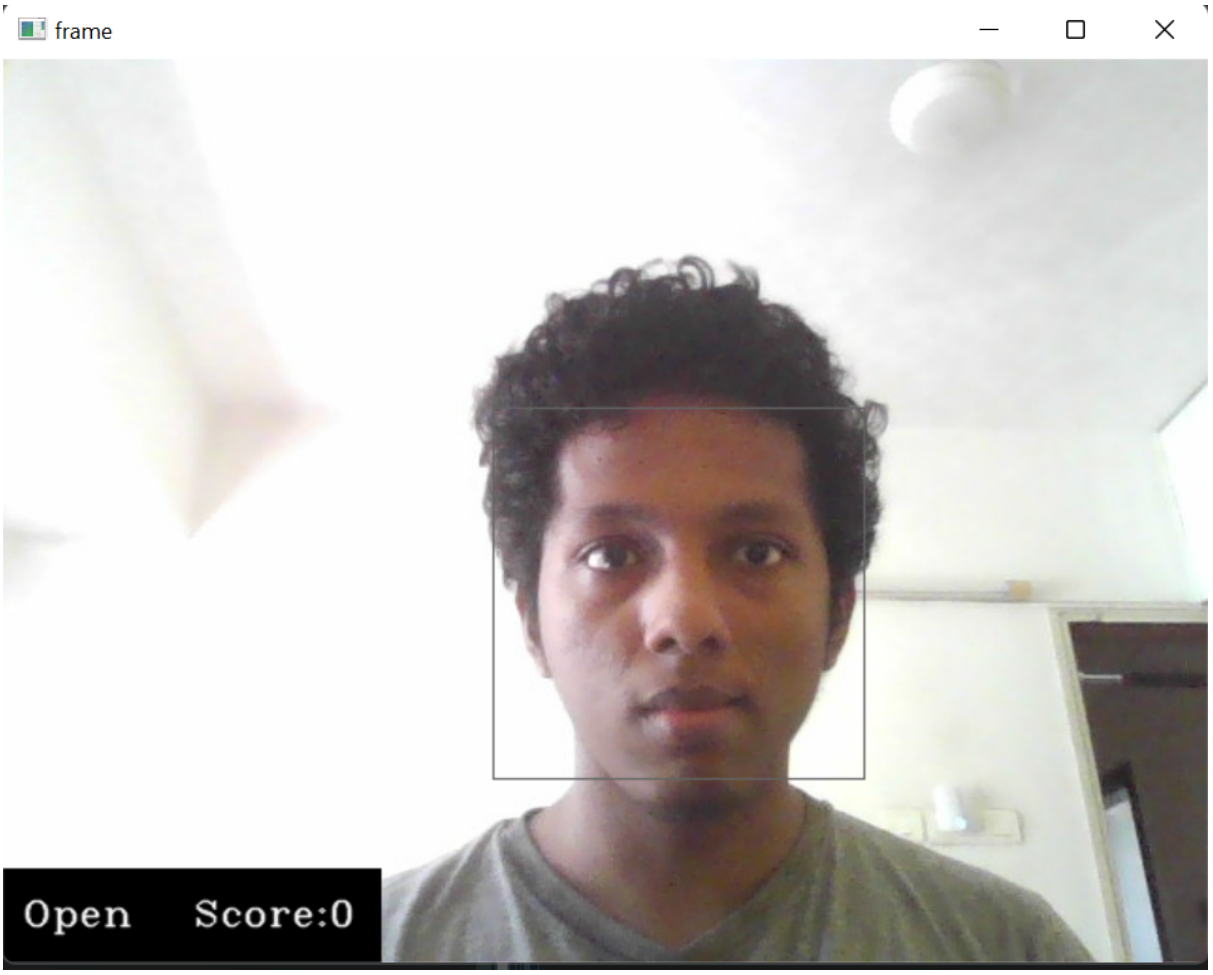
4.1 Introduction:

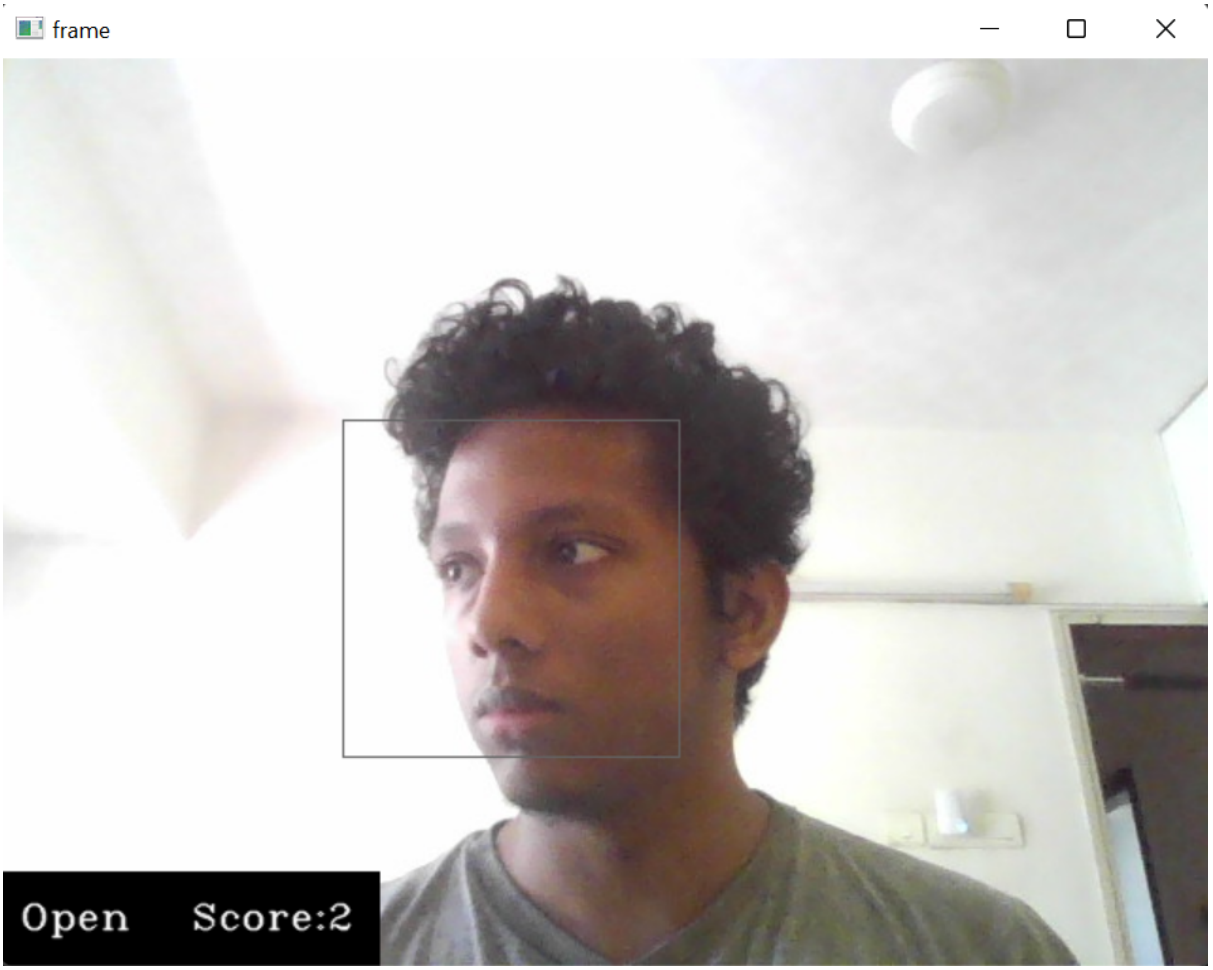
We plan to Implement our system in a slow and gradual manner. We fill first test the detection of the eyes then the detection of drowsiness in the eyes.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

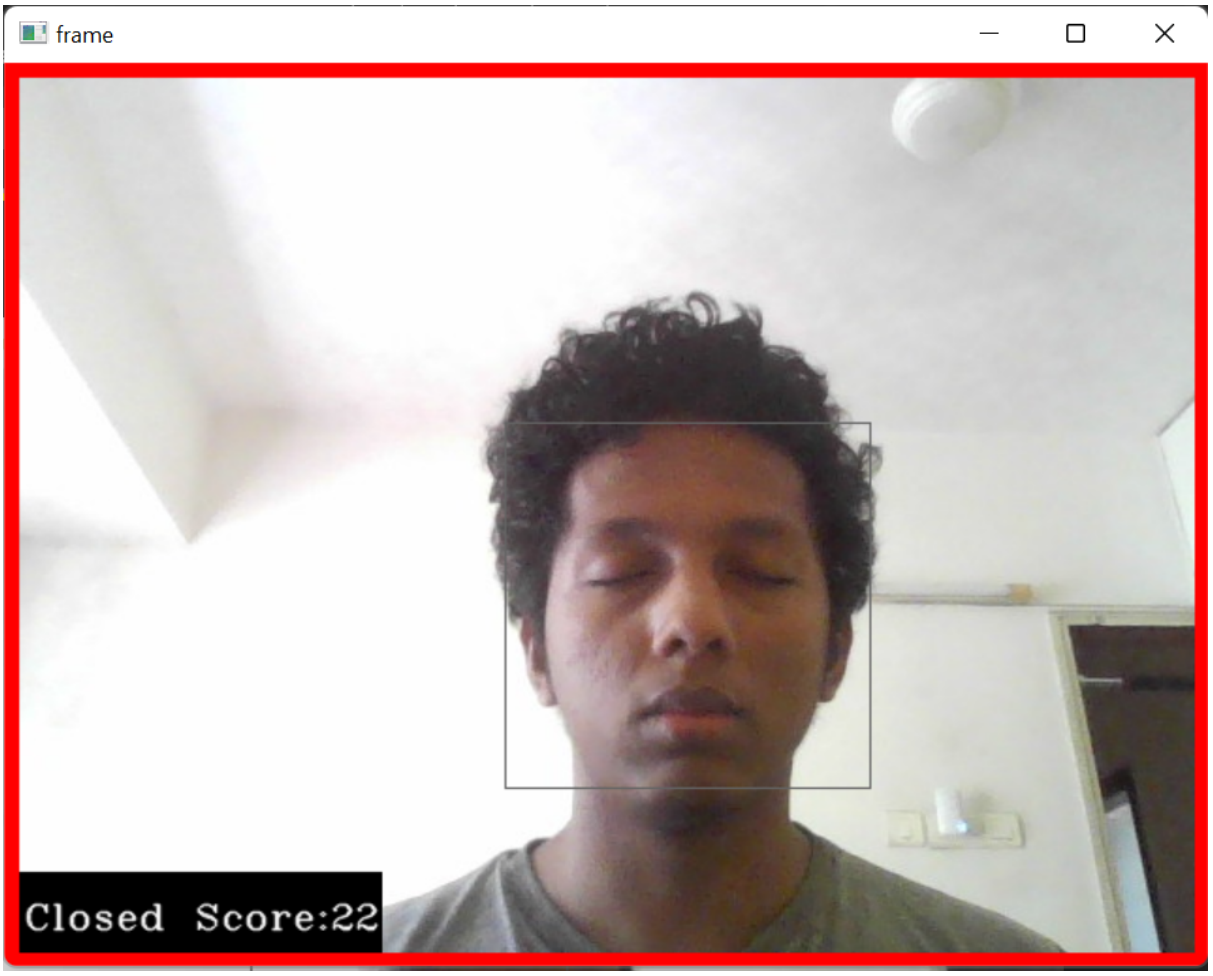
GUI Screenshots

Face Detected:

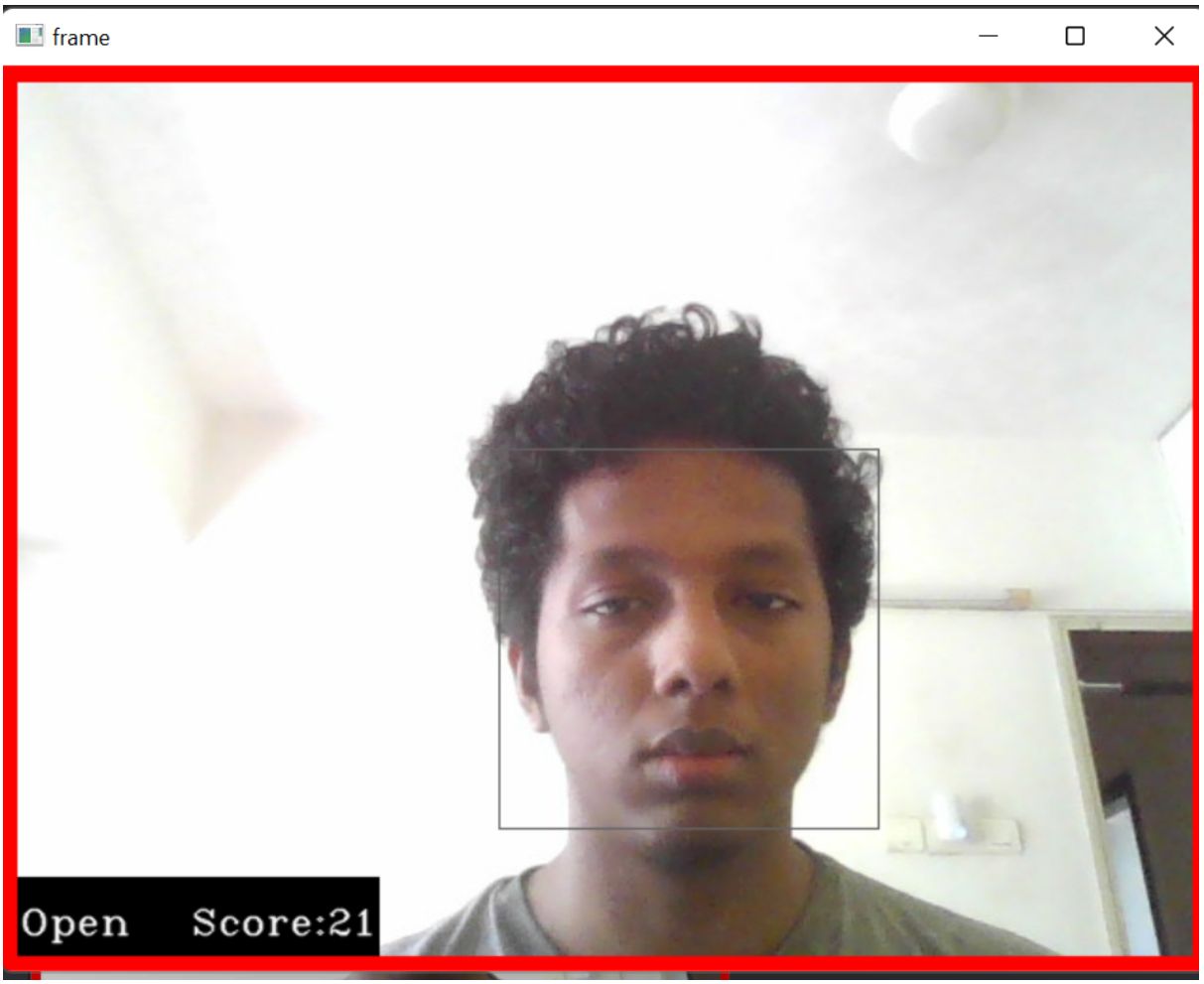




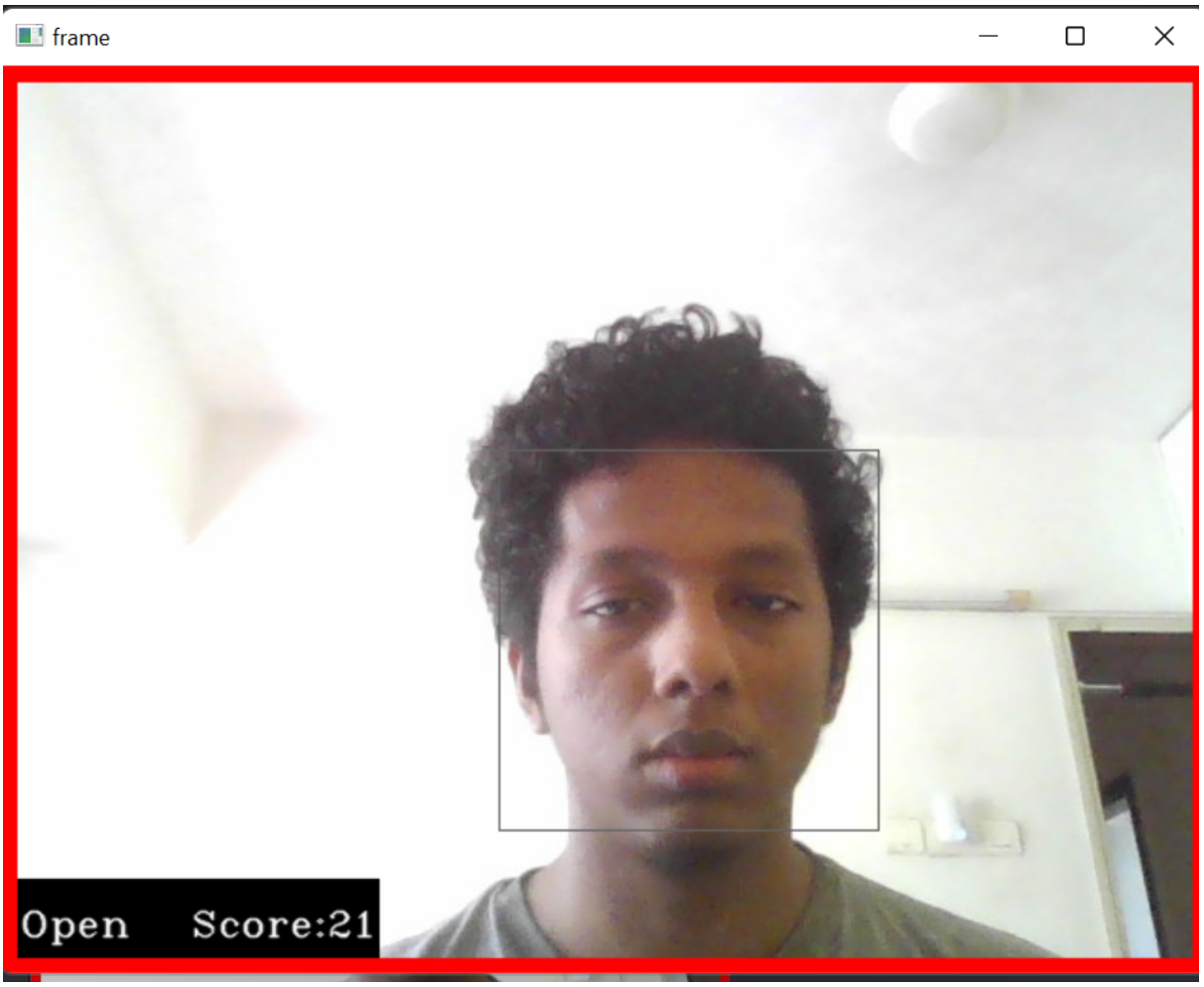
Eyes Full Closed:



Eyes Half Closed:



Tired State:



Thus, our system can successfully detect the drowsiness level of a driver.

Functioning:

The OpenCV program manages to launch the system's camera and by using the haar cascade files-models to detect facial features-the system can recognize different parts of the face, most importantly, the eyes.

This input is passed through the machine learning model which has been programmed to give the open or closed score of the eyes. At a certain score, the alarm will trigger and the whole picture frame will be encased by a red border. This is the signal to the system operator, notifying them that the driver is drowsy.

CHAPTER 5

TESTING

5.1 INTRODUCTION TO SYSTEM TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.2 TYPES OF TESTING:

1. **Unit testing:**

   Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2.Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3.Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for

testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4. System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5. **White Box Testing:**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6. **Black Box Testing:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7. **Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8. **Integration Testing:**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

9. **Acceptance Testing:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

5.3 Testing of Project

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.

- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 6

CONCLUSION

We have made this project to help reduce accidents, particularly on highways and also help officials identify violations of the laws. With this, we hope to make driving a safer experience for all involved.

CHAPTER

REFERENCES

We have referred the following websites:

1) https://pythonprogramming.net/
2) https://github.com/opencv/opencv
3) https://www.geeksforgeeks.org/opencv-overview/
4) https://github.com/keras-team/keras
5) https://keras.io/guides/
6) https://faroit.com/keras-docs/1.2.0/