A

Project Report On

“DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIORS AND MACHINE LEARNING ”

submitted in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

19G31A0508	-	**E.ANIL KUMAR**
19G31A0531	-	**M.NIKHIL SAI**
19G31A0548	-	**S.NARASIMHULU**
19G31A0532	-	**M.MEHBOOB HASAN**
19G31A0506	-	**C.NAVEEN KUMAR**

Under the esteemed guidance of

Mr. M. JAGADEESH M. TECH.

Associate Professor,

Dept. of Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ST. JOHNS COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to JNTUA, Anantapuramu and Approved by AICTE, New Delhi)

Yerrakota, Yemmiganur, Kurnool (Dist.) – 518360, A.P.

2019- 2023

A

Project Report On

"DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIORS AND MACHINE LEARNING"

submitted in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

19G31A0508	**-**	**E.ANIL KUMAR**
19G31A0531	**-**	**M.NIKHIL SAI**
19G31A0548	**-**	**S.NARASIMHULU**
19G31A0532	**-**	**M.MEHBOOB HASAN**
19G31A0506	**-**	**C.NAVEEN KUMAR**

Under the esteemed guidance of

Mr.M.JAGADEESH M. TECH.

Associate Professor,

Department of Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ST. JOHNS COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to JNTUA, Anantapuramu and Approved by AICTE, New Delhi)

Yerrakota, Yemmiganur, Kurnool (Dist.) – 518360, A.P.

2019- 2023

ST. JOHNS COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to JNTUA, Anantapuramu and Approved by AICTE, New Delhi)

Yerrakota, Yemmiganur, Kurnool (Dist.) – 518360, A.P.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIORS AND MACHINE LEARNING"** submitted by E.ANIL KUMAR (19G31A0508), in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in St. Johns College of Engineering & Technology, Yerrakota,Yemmiganur, Kurnool, A. P. It is a record of bonafide work carried out under my guidance and supervision.

Project Guide

M.JAGADEESH
Associate Professor
Department of CSE

Head of the Department

Dr. P.VEERESH
Professor
Department of CSE

Internal Examiner

External Examiner

DECLARATION BY THE CANDIDATE

I, E.ANIL KUMAR (19G31A0508), hereby declare that the Project Report entitled **"DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIORS AND MACHINE LEARNING"**, under the guidance of **M.JAGADEESH** Associate professor, Department of Computer Science and Engineering is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

This is a Record of Bonafide work carried out by us and the results embodied in this Project Report have not been reproduced or copied from any source. The results embodied in this Project Report have not submitted to any other University or Institute for the Award of any other degree.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our **Project Guide, Mr. M.JAGADEESH**, Associate Professor, Department of Computer Science and Engineering, for his guidance with unsurpassed knowledge and immense encouragement.

We express our thanks to **Project Coordinator, Mrs. S.S.RAJAKUMARI M.Tech(Ph.D)**, for her Continuous support and encouragement.

We are grateful to **Dr. P. VEERESH**, **Head of the Department**, Computer Science and Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management**, St. Johns College of Engineering and Technology, **Dr. V. VEERANNA**, for their encouragement and cooperation to carry out this work.

We thank all **teaching faculty** of Department of CSE, whose suggestions during reviews helped us in accomplishment of our project.

We would like to thank our **parents, friends, and classmates** for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT MEMBERS:

19G31A0508	**-**	**E.ANIL KUMAR**
19G31A0531	**-**	**M.NIKHIL SAI**
19G31A0548	**-**	**S.NARASIMHULU**
19G31A0532	**-**	**M.MEHBOOB HASAN**
19G31A0506	**-**	**C.NAVEEN KUMAR**

ABSTRACT

Drowsy driving is one of the major causes of road accidents and death. Hence, detection of driver's fatigue and its indication is an active research area. Most of the conventional methods are either vehicle based, or behavioral based or physiological based. Few methods are intrusive and distract the driver, some require expensive sensors and data handling. Therefore, in this study, a low cost, real time driver's drowsiness detection system is developed with acceptable accuracy. In the developed system, a webcam records the video and driver's face is detected in each frame employing image processing techniques. Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio, mouth opening ratio and nose length ratio are computed and depending on their values, drowsiness is detected based on developed adaptive thresholding. Machine learning algorithms have been implemented as well in an offline manner.

INDEX

LIST OF FIGURES

LIST OF TABLES

ACRONYMS & ABBREVIATIONS

- **HTML:** Hyper Text Markup Language.
- **XML:** Extensible Markup Language.
- **IDE:** Integrated Development Environment
- **PHP:** Hyper Text Preprocessor
- **RDBMS:** Relational Database Management System.
- **GUI:** Graphical User Interface
- **HTTP:** Hyper Text Transfer Protocol
- **API:** Application Programming Interface
- **E-R:** Entity-Relationship
- **UML:** Unified Modeling Language
- **OOAD:** Object-Oriented Analysis & Design.

CHAPTER - 1
INTRODUCTION
The chapter gives brief introduction of the project.

CHAPTER 1
INTRODUCTION

Drowsy driving is one of the major causes of deaths occurring in road accidents. The truck drivers who drive for continuous long hours (especially at night), bus drivers of long distance route or overnight buses are more susceptible to this problem. Driver drowsiness is an overcast nightmare to passengers in every country. Every year, a large number of injuries and deaths occur due to fatigue related road accidents. Hence, detection of driver's fatigue and its indication is an active area of research due to its immense practical applicability. The basic drowsiness detection system has three blocks/modules; acquisition system, processing system and warning system. Here, the video of the driver's frontal face is captured in acquisition system and transferred to the processing block where it is processed online to detect drowsiness. If drowsiness is detected, a warning or alarm is send to the driver from the warning system.

Generally, the methods to detect drowsy drivers are classified in three types; vehicle based, behavioural based and physiological based. In vehicle based method, a number of metrics like steering wheel movement, accelerator or brake pattern, vehicle speed, lateral acceleration, deviations from lane position etc. are monitored continuously. Detection of any abnormal change in these values is considered as driver drowsiness. This is a nonintrusive measurement as the sensors are not attached on the driver. In behavioural based method, the visual behavior of the driver i.e., eye blinking, eye closing, yawn, head bending etc. are analyzed to detect drowsiness. This is also nonintrusive measurement as simple camera is used to detect these features. In physiological based method, the physiological signals like Electrocardiogram (ECG), Electooculogram (EOG), Electroencephalogram (EEG), heartbeat, pulse rate etc. are monitored and from these metrics, drowsiness or fatigue level is detected. This is intrusive measurement as the sensors are attached on the driver which will distract the driver. Depending on the sensors used in the system, system cost as well as size will increase. However, inclusion of more parameters/features will increase the accuracy of the system to a certain extent. These factors motivate us to develop a low-cost, real time driver's drowsiness detection system with acceptable accuracy. Hence, we have proposed a webcam based system to detect driver's fatigue from the face image only using image processing and machine learning techniques to make the system low-cost as well as portable.

1.2 Motivation For Work

Driver drowsiness is a significant factor in the increasing number of accidents on today's roads and has been extensively accepted . This proof has been verified by many researchers that have demonstrated ties between driver drowsiness and road accidents. Although it is hard to decide the exact number of accidents due to drowsiness, it is much likely to be underestimated. The above statement shows the significance of a research with the objective of reducing the dangers of accidents anticipated to drowsiness. So far, researchers have tried to model the behavior by creating links between drowsiness and certain indications related to the vehicle and to the driver .

Previous approaches to drowsiness detection primarily make pre-assumptions about the relevant behavior, focusing on blink rate, eye closure, and yawning . The automobile business also has tried to build several systems to predict driver drowsiness but there are only a few commercial products available today. The systems do not look at driver performance and overlook driver ability and characteristics. Naturally, most people would agree that different people drive differently. The system that being develop able to adapt to the changes of the driver's behaviour

1.3 Problem Statement

Drowsy driving is a significant public safety concern that can result in severe accidents and fatalities. According to the National Highway Traffic Safety Administration (NHTSA), drowsy driving is estimated to be a factor in up to 7,500 fatal crashes and 100,000 crashes with injuries in the United States each year. Many accidents caused by drowsy driving could be prevented if early warning systems were in place to detect and alert drivers when they are becoming drowsy.

The current methods for detecting drowsiness in drivers, such as self-assessment questionnaires, are subjective and unreliable. There is a need for a robust and accurate driver drowsiness detection system that can objectively and non-invasively monitor a driver's physiological and behavioral cues in real-time and provide timely alerts to prevent accidents due to drowsy driving.

The problem statement for this project is to develop a driver drowsiness detection system that can accurately and reliably detect the drowsiness level of a driver in real-time using a combination of physiological and behavioral cues. The system should be able to monitor the driver's facial expressions, eye movements, head movements, and other relevant physiological parameters to determine the drowsiness level. It should also be able to integrate with existing in-vehicle technologies and provide timely alerts, such as visual or auditory warnings, to alert the driver when their drowsiness level reaches a dangerous threshold.

The driver drowsiness detection system should be designed to work in various driving conditions, including day and night driving, different weather conditions, and different road types. It should be scalable and adaptable to different types of vehicles, including passenger cars, commercial trucks, and public transportation vehicles. The system should also be user-friendly, requiring minimal setup and calibration, and should not interfere with the normal driving experience or distract the driver.

The successful development of an accurate and reliable driver drowsiness detection system has the potential to significantly reduce the number of accidents caused by drowsy driving and save lives. It could be used in various applications, including passenger vehicles, commercial fleets, and public transportation, to improve road safety and prevent accidents due to driver drowsiness.

1.4 Introduction about Visual Studio Code

Visual Studio Code (VS Code) is a popular, lightweight, and free source code editor developed by Microsoft for Windows, macOS, and Linux. It has gained immense popularity among developers due to its rich features, extensibility, and cross-platform compatibility.

VS Code is designed to provide a fast and efficient coding experience with a modern and intuitive user interface. It offers a wide range of features, including syntax highlighting, code completion, debugging, version control integration, and built-in terminal, which make it a powerful tool for software development across various programming languages and frameworks.

One of the key features of VS Code is its extensibility. It has a large marketplace of extensions, which can be easily installed to enhance its functionality according to the specific needs of developers. These extensions provide support for different programming languages, frameworks, tools, and integrations with popular development services, making VS Code a versatile code editor for a wide range of development tasks.

Additionally, VS Code has a vibrant and active community of developers who contribute to its development, provide support, and continuously improve its features and performance. It also has a strong ecosystem of documentation, tutorials, and online resources, which makes it easy for developers to learn and get started with VS Code.

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

Visual Studio Code Insiders logo

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, debuggers, time travel debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where users can access version control settings and view changes made to the current project. To use the feature, Visual Studio Code must be linked to any supported version control system (Git, Apache Subversion, Perforce, etc.). This allows users to create repositories as well as to make push and pull requests directly from the Visual Studio Code program.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled. Due to the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected.

In summary, Visual Studio Code is a powerful, extensible, and widely used code editor that provides a modern and efficient coding experience for developers. Its features, extensibility, and community support make it a popular choice among developers for various software development tasks.

CHAPTER – 2

KEY TECHNOLOGIES USED

The chapter gives brief introduction of key technologies used in the project

2.1 Introduction about python

Python is a high-level, interpreted, and general-purpose programming language that is widely used for various applications such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was created by Guido van Rossum and first released in 1991, and has since become one of the most popular programming languages worldwide due to its simplicity, versatility, and extensive community support.

One of the key features of Python is its readability and ease of use, which makes it an excellent language for beginners as well as experienced developers. Python's syntax is designed to be simple and easy to understand, with a focus on code readability and minimizing the cost of program maintenance. Python uses indentation to denote blocks of code, which makes it visually appealing and helps in writing clean and organized code.

Python has a large standard library that provides a wide range of modules for handling various tasks such as file I/O, regular expressions, networking, databases, GUI development, and more. Additionally, Python has a vast ecosystem of third-party libraries and frameworks, such as NumPy for numerical computing, Pandas for data analysis, TensorFlow for machine learning, Django for web development, and Flask for building APIs, which makes it highly extensible and suitable for diverse applications.

Another notable aspect of Python is its strong community support. Python has a large and active community of developers who contribute to its development, create and maintain libraries, and provide support through forums, mailing lists, and online communities. This vibrant community ensures that Python remains up-to-date with the latest trends in technology and provides continuous improvements to the language.

Overall, Python's simplicity, versatility, and extensive community support make it a popular choice for a wide range of applications, from small scripts to complex software projects, and it continues to be one of the most widely used programming languages in the world.

Python is also known for its cross-platform compatibility, which means that Python code can be run on different operating systems such as Windows, macOS, Linux, and others without requiring any modifications to the code. This makes Python a highly portable language, allowing developers to write code once and run it on multiple platforms, which is particularly beneficial for large-scale applications and projects with diverse deployment environments.

Python's versatility also extends to its support for different programming paradigms, including procedural, object-oriented, and functional programming. This makes Python a flexible language that can be used for a wide range of programming styles, depending on the requirements of the project.

Furthermore, Python has a strong focus on code reusability and modularity, which allows developers to write modular and maintainable code. Python's package management system, such as pip (Python Package Index), makes it easy to install, manage, and share

third-party libraries, enabling developers to leverage existing code and build applications more efficiently.

Python also has a large and growing presence in the field of data science, machine learning, and artificial intelligence. With libraries such as NumPy, Pandas, Matplotlib, and scikit-learn, Python has become a popular choice for data analysis, visualization, and machine learning tasks. Its simplicity and ease of use, combined with powerful libraries, make Python a preferred language for many data scientists and machine learning practitioners.

In addition to its technical capabilities, Python has a supportive and welcoming community. The Python community is known for its inclusivity, diversity, and collaborative spirit, with a wealth of resources available, including comprehensive documentation, online tutorials, forums, conferences, and meetups. This strong community-driven approach fosters learning, sharing, and collaboration among Python developers worldwide.

In conclusion, Python is a versatile, readable, and powerful programming language that is widely used for a wide range of applications, from web development to scientific computing to machine learning. Its simplicity, cross-platform compatibility, extensive library support, and strong community make it a popular choice among developers of all levels of expertise. Whether you're a beginner or an experienced developer, Python provides a robust and enjoyable programming experience.

2.2 PACKAGES

1)Numpy

NumPy is a powerful Python library for numerical computing that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. It is one of the most commonly used libraries in data science, machine learning, and scientific computing due to its efficiency and versatility.

Some key features of NumPy include:

Arrays: NumPy provides the ndarray (N-dimensional array) object, which is a homogeneous, multi-dimensional array that can have any number of dimensions. Arrays in NumPy are more efficient and memory optimized compared to Python lists, making them ideal for numerical computations.

Mathematical Operations: NumPy offers a wide range of mathematical functions for performing various operations on arrays, such as mathematical operations (addition, subtraction, multiplication, etc.), statistical operations (mean, median, standard deviation, etc.), linear algebra, Fourier analysis, and more. These operations are vectorized, meaning they are performed element-wise on arrays, making computations fast and efficient.

Broadcasting: NumPy allows for broadcasting, which is a powerful feature that enables element-wise operations on arrays of different shapes and sizes, without requiring

explicit shape matching or copying of data. This makes it easy to perform operations on arrays with different dimensions and sizes, which can greatly simplify code and improve performance.

Integration with other Libraries: NumPy is often used in conjunction with other popular libraries in the Python ecosystem, such as Pandas for data analysis, Matplotlib for data visualization, and Scikit-learn for machine learning. NumPy provides the foundation for efficient numerical computations in these libraries and allows for seamless integration.

Performance: NumPy is written in C and allows for vectorized operations, making it highly optimized for numerical computations. Its efficient array operations and mathematical functions make it much faster than using regular Python lists for numerical computing tasks, making it a preferred choice for performance-critical applications.

To use NumPy in your Python code, you need to install it using a package manager such as pip or conda. Once installed, you can import it into your Python script or Jupyter notebook and start using its functions and arrays for numerical computations. NumPy's official documentation is extensive and provides detailed information on how to use its various functions and features for different use cases.

To install NumPy using pip, follow these steps:

Open a command prompt or terminal window.

Ensure that you have Python and pip (Python's package manager) installed on your system. If not, you can download Python from the official Python website (https://www.python.org/) and pip will be included with it.

Enter the following command to install NumPy using pip:

```
pip install numpy
```

Press Enter to execute the command.

pip will download and install the NumPy package along with its dependencies.

Once the installation is complete, you can import NumPy in your Python script or Jupyter notebook using the following statement:

Python

```
import numpy as np
```

Now you are ready to use NumPy in your Python code for numerical computing tasks! NumPy's official documentation (https://numpy.org/doc/stable/) provides detailed information on how to use its functions and features for various numerical computations.

2)Scipy

Scipy is a powerful scientific computing library for Python that is built on top of NumPy. It provides a wide range of advanced numerical computing capabilities, including optimization, integration, interpolation, linear algebra, signal and image processing, statistical analysis, and more. Scipy is widely used in fields such as physics, engineering, biology, finance, and machine learning.

Some key features of Scipy include:

Optimization: Scipy provides a collection of optimization algorithms for solving various types of optimization problems, such as linear programming, nonlinear programming, unconstrained and constrained optimization, and more. These optimization algorithms can be used to find the optimal values of parameters in mathematical models or machine learning algorithms.

Integration: Scipy offers numerical integration routines for computing definite and indefinite integrals of functions. These integration routines are useful for solving problems that involve finding areas under curves, computing probabilities, and more.

Interpolation: Scipy provides interpolation functions for constructing smooth curves or surfaces from a set of scattered data points. These interpolation functions are useful for estimating values of a function at points that lie between the data points or for creating smooth representations of data for visualization or further analysis.

Linear Algebra: Scipy includes a comprehensive suite of linear algebra functions for performing various operations on matrices and vectors, such as solving linear systems of equations, eigenvalue and eigenvector computations, matrix factorizations, and more.

Signal and Image Processing: Scipy offers a wide range of signal and image processing functions for tasks such as filtering, convolution, image enhancement, feature extraction, and more. These functions are useful in applications such as image processing, audio processing, and other signal processing tasks.

Statistical Analysis: Scipy provides statistical functions for descriptive statistics, hypothesis testing, probability distributions, regression analysis, and more. These statistical functions are useful for analyzing data and making statistical inferences in scientific research, data analysis, and machine learning.

Integration with other Libraries: Scipy is often used in conjunction with other popular libraries in the Python ecosystem, such as NumPy, Matplotlib, and Pandas. Scipy provides advanced numerical computing capabilities that complement the functionalities of these libraries, allowing for seamless integration and powerful data analysis workflows.

To use Scipy in your Python code, you need to install it using a package manager such as pip or conda. Once installed, you can import it into your Python script or Jupyter notebook and start using its functions and features for advanced numerical computing tasks. Scipy's official documentation (https://docs.scipy.org/doc/scipy/reference/) provides detailed information on how to use its functions and features for different use cases.

Why use SciPy?

SciPy contain significant mathematical algorithms that provide easiness to develop sophisticated and dedicated applications. Being an open-source library, it has a large community across the world to the development of its additional module, and it is much beneficial for scientific application and data scientists.

To install Scipy, you can use a package manager such as pip or conda. Here are the steps for installation using pip:

Open a command prompt or terminal window.

Ensure that you have Python and pip (Python's package manager) installed on your system. If not, you can download Python from the official Python website (https://www.python.org/) and pip will be included with it.

Enter the following command to install Scipy using pip:

```
pip install scipy
```

Press Enter to execute the command.

pip will download and install the Scipy package along with its dependencies.

Once the installation is complete, you can import Scipy in your Python script or Jupyter notebook using the following statement:

```
import scipy
```

3) Matplotlib

Matplotlib is a popular and powerful data visualization library in Python that provides a wide range of tools for creating static, animated, and interactive visualizations. It is widely used in the field of data science, machine learning, and scientific computing to create visually appealing and informative plots and charts to analyze data and communicate results effectively.

Matplotlib was originally created by John D. Hunter in 2003 and has since become one of the most widely used libraries for data visualization in Python. It provides a flexible and customizable interface for creating various types of plots, including line plots, scatter plots, bar plots, histograms, 3D plots, and more. Matplotlib allows users to create publication-quality plots with extensive control over plot properties, such as colors, markers, fonts, and labels.

Matplotlib also integrates well with other popular Python libraries such as NumPy, Pandas, and SciPy, making it a powerful tool for data analysis and visualization. It has a large and active community of users and developers, which ensures regular updates, bug fixes, and improvements to the library.

Whether you are a beginner or an experienced data scientist, Matplotlib provides a wide range of functionalities to create visually appealing and meaningful plots to gain insights from data and communicate results effectively. Its versatility, ease of use, and wide adoption make it an essential tool in the Python data visualization ecosystem.

4)Cycler

"Cycler" is a Python library that provides a simple and convenient way to define and manipulate cycling behavior of various parameters in data visualization. It is often used in conjunction with Matplotlib to create customized and dynamic plots.

Cycler allows users to define cycles of values for different plot parameters, such as line styles, colors, markers, and more. These cycles can then be applied to different plot elements, allowing for easy customization of visual properties across multiple data points or plot elements. For example, you can define a cycle of line styles and apply it to multiple lines in a line plot, creating visually distinctive lines with ease.

Cycler provides a flexible and intuitive API for creating and manipulating cycles, including options for linear and nonlinear interpolation, cycling through predefined sequences, and more. It also supports nesting of cycles, allowing for complex and dynamic cycling behavior.

One of the key benefits of using Cycler in combination with Matplotlib is that it provides a consistent and efficient way to customize plot properties across different plot types and styles. It allows users to define reusable and consistent styles for their plots, making it easier to create visually appealing and professional-looking visualizations.

Cycler is a powerful tool for creating dynamic and customized visualizations in Matplotlib, and it is widely used in the data science and visualization communities to create visually engaging and informative plots. It is a useful addition to any Python data visualization workflow that involves Matplotlib and provides a convenient way to create visually appealing and customized plots.

6)Pandas

Pandas is a popular open-source Python library for data manipulation and analysis. It provides data structures such as Series and Data Frame, along with a wide range of functions and methods for data cleaning, preparation, analysis, and visualization. Pandas is widely used in data science, machine learning, and data analysis tasks due to its efficiency, ease of use, and versatility. It allows users to handle and analyze data in a flexible and efficient manner, making it a powerful tool for data manipulation and analysis in Python.

Installation of Pandas: You can install Pandas using pip, which is the package installer for Python. Follow the steps below to install Pandas:

Step 1: Open a command-line interface or terminal on your computer.

Step 2: Ensure that you have Python and pip installed. If you don't have Python or pip installed, you can download and install them from the official Python website (https://www.python.org/).

Step 3: Enter the following command to install Pandas using pip:

```
pip install pandas
```

This will download and install the Pandas library along with its dependencies.

Step 4: Once the installation is complete, you can import Pandas in your Python script or Jupyter notebook using the import statement, like this:

```
import pandas as pd
```

The pd alias is a common convention used for Pandas to make it easier to reference Pandas functions and objects in your code.

That's it! You have now successfully installed Pandas and can start using it for data manipulation and analysis in Python.

7) kiwisolver

Kiwisolver is an open-source, lightweight, and efficient C++ library for solving nonlinear equations and optimization problems. It provides a set of solvers for finding solutions to systems of equations, nonlinear least-squares problems, and unconstrained and constrained optimization problems. Kiwisolver is commonly used in scientific computing, numerical simulations, and optimization tasks.

Kiwisolver is known for its fast and robust performance, and it offers several different solvers with varying capabilities and trade-offs, including the Newton-Raphson method, the Levenberg-Marquardt algorithm, and the Trust Region Reflective algorithm. It also supports dense and sparse matrix representations, and it can handle both smooth and non-smooth problems.

One notable feature of Kiwisolver is its Python interface, which makes it easy to use in Python-based projects. It is often used in combination with other Python libraries for scientific computing and data analysis, such as NumPy, SciPy, and scikit-learn, to solve complex mathematical problems efficiently.

Overall, Kiwisolver is a powerful tool for solving nonlinear equations and optimization problems, and it is widely used in various fields such as physics, engineering, finance, and machine learning. Its efficient implementation and Python interface make it a popular choice for many numerical computing tasks.

8)Pyptz

"pytz" is a Python library that provides time zone support for working with dates and times in different time zones. It allows you to work with date and time objects in Python while taking into account the various time zones around the world, including daylight saving time (DST) changes and historical time zone data.

With "pytz", you can perform operations such as converting datetime objects between different time zones, calculating time differences between time zones, and handling ambiguous or non-existent local times during DST transitions. "pytz" also provides access to a comprehensive database of time zones, including many historical and regional time zones.

Here's an example of how you might use "pytz" to work with time zones in Python:

import datetime

import pytz

 # Create a datetime object in a specific time zone

dt = datetime.datetime(2023, 4, 16, 12, 0) # April 16, 2023 12:00 PM

```
tz = pytz.timezone('US/Eastern')  # Eastern Time Zone (UTC-5)

dt_eastern = tz.localize(dt)  # Localize the datetime object to Eastern Time Zone

# Convert to another time zone

tz_pacific = pytz.timezone('US/Pacific')  # Pacific Time Zone (UTC-8)

dt_pacific = dt_eastern.astimezone(tz_pacific)  # Convert to Pacific Time Zone

# Format the datetime object with the time zone information

print(dt_eastern.strftime('%Y-%m-%d %H:%M:%S %Z%z'))  # 2023-04-16 12:00:00 EDT-0400

print(dt_pacific.strftime('%Y-%m-%d %H:%M:%S %Z%z'))  # 2023-04-16 09:00:00 PDT-0700
```

Note that "pytz" is a third-party library and needs to be installed separately using a package manager such as pip. It is commonly used in Python applications that deal with dates, times, and time zones, such as web applications, data analysis, and scientific computing.

9)OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library written in C++ and Python. It provides a wide range of functions for image and video processing, including image filtering, feature detection and extraction, object recognition, camera calibration, and image stitching, among others. OpenCV is widely used in various applications, such as robotics, autonomous vehicles, augmented reality, medical imaging, and computer vision research.

OpenCV was initially developed by Intel in 1999 and has since grown into a popular and widely-used open-source library with a large community of developers and users. It is cross-platform and runs on multiple operating systems, including Windows, macOS, Linux, and even mobile platforms such as Android and iOS.

Some of the key features of OpenCV include:

Image and video I/O: OpenCV provides functions to read and write images and videos in various formats, including common image formats such as JPEG, PNG, BMP, and video formats such as AVI, MP4, and MKV.

Image processing: OpenCV includes a rich set of functions for image processing tasks such as image filtering (e.g., Gaussian blur, median blur), image enhancement (e.g., contrast adjustment, histogram equalization), and morphological operations (e.g., erosion, dilation).

Feature detection and extraction: OpenCV provides various algorithms for detecting and extracting features from images, such as corners, edges, and blobs. These features can be used for tasks such as object recognition, image stitching, and image registration.

Machine learning: OpenCV includes machine learning modules that provide support for training and using machine learning models, such as support vector machines (SVMs), k-nearest neighbors (KNN), and neural networks, for tasks such as image classification and object detection.

Camera calibration: OpenCV provides functions for camera calibration, which is an important step in computer vision tasks that involve estimating camera parameters such as intrinsic and extrinsic parameters, distortion coefficients, and camera pose.

Computer vision algorithms: OpenCV includes a wide range of computer vision algorithms, such as optical flow, object tracking, stereo vision, and 3D reconstruction, which are commonly used in applications such as robotics, augmented reality, and autonomous vehicles.

Python bindings: OpenCV has a Python interface that provides easy access to its functions using Python programming language, making it popular among developers who prefer Python for rapid prototyping and development.

OpenCV has an extensive documentation and a large community of users and developers, which makes it a powerful tool for computer vision and image processing tasks in various applications. Whether you are a beginner or an experienced developer, OpenCV can be a valuable resource for your computer vision projects.

10)h5py

"h5py" is a Python library that provides a high-level interface for working with Hierarchical Data Format 5 (HDF5) files. HDF5 is a data model, file format, and library for storing and managing large and complex datasets. It is widely used in scientific computing, data analysis, and machine learning applications due to its ability to store and organize large amounts of data in a hierarchical and flexible manner.

"h5py" allows you to create, read, write, and manipulate HDF5 files using a simple and intuitive Pythonic interface. It provides support for various data types, including numerical data, text data, and complex data structures such as nested arrays and compound data types. Some of the key features of "h5py" include:

Efficient data storage: "h5py" provides efficient storage of large datasets in HDF5 files, including support for compression and chunking, which allows for optimized storage and retrieval of data.

Hierarchical data organization: HDF5 files can store data in a hierarchical structure, similar to a file system. "h5py" provides a Pythonic interface to create and manage groups, datasets, and attributes in an HDF5 file, allowing you to organize and manage your data in a hierarchical manner.

Data indexing and slicing: "h5py" allows you to efficiently index and slice datasets in HDF5 files, similar to how you would index and slice arrays in Python, allowing for selective retrieval of data from large datasets without loading the entire dataset into memory.

Support for large datasets: "h5py" can handle large datasets that do not fit in memory, allowing you to work with datasets that are larger than the available RAM on your system.

Interoperability: "h5py" provides interoperability with other Python libraries and tools for scientific computing and data analysis, such as NumPy, Pandas, and Matplotlib. It also supports compatibility with other HDF5 libraries in other programming languages.

Performance optimizations: "h5py" provides performance optimizations, such as caching and parallel I/O, to efficiently read and write data from and to HDF5 files.

Pythonic interface: "h5py" provides a Pythonic and intuitive interface for working with HDF5 files, making it easy to integrate into your Python-based data processing workflows.

"h5py" is a popular and widely-used library in the scientific computing and data analysis communities, particularly in applications that involve working with large and complex datasets. It is well-documented, actively maintained, and has a large community of users and developers, making it a powerful tool for working with HDF5 files in Python.

11)TensorFlow

TensorFlow is a popular open-source machine learning framework developed by Google, and it has a Python API that allows developers to build, train, and deploy machine learning models using Python. Here's an overview of how you can use TensorFlow in Python:

Installation: You can install TensorFlow in Python using the pip package manager, which is the most common method. You can run the following command in your Python environment to install TensorFlow:

```
pip install tensorflow
```

Importing TensorFlow: After installation, you can import TensorFlow in your Python script using the following import statement:

```
import tensorflow as tf
```

Defining a Computational Graph: TensorFlow uses a computational graph to define and execute computations. You can define a computational graph by creating TensorFlow operations (ops) and tensors. Ops are operations that perform computations, and tensors are multi-dimensional arrays that hold the data.

Running a Session: To execute computations in a TensorFlow graph, you need to create a session. A session is an environment for running computations in a TensorFlow graph. You can create a session using the tf.Session() class, and then use the run() method to execute operations and fetch the results.

Building and Training a Model: TensorFlow provides various APIs for building and training machine learning models. You can use high-level APIs like Keras, which is a user-friendly API for building deep neural networks, or lower-level APIs like the tf.layers and tf.nn modules for building custom models.

Evaluating and Deploying a Model: Once a model is trained, you can evaluate its performance on test data using the **evaluate()** method, and deploy it for inference on new data. TensorFlow provides tools for saving and loading models, and you can use the **predict()** method to make predictions with a trained model.

These are just some basic examples of how you can use TensorFlow in Python to build, train, and deploy machine learning models. TensorFlow provides a wide range of functionality for various machine learning tasks, including image recognition, natural language processing, time series analysis, and more, and it has a large community of developers and users who contribute to its ongoing development

12)Scikit-Learn (Sklearn)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Origin of Scikit-Learn

It was originally called *scikits.learn* and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Let’s have a look at its version history −

May 2019: scikit-learn 0.21.0

March 2019: scikit-learn 0.20.3

December 2018: scikit-learn 0.20.2

November 2018: scikit-learn 0.20.1

September 2018: scikit-learn 0.20.0

July 2018: scikit-learn 0.19.2

July 2017: scikit-learn 0.19.0

September 2016. scikit-learn 0.18.0

November 2015. scikit-learn 0.17.0

March 2015. scikit-learn 0.16.0

July 2014. scikit-learn 0.15.0

August 2013. scikit-learn 0.14

Prerequisites

Before we start using scikit-learn latest release, we require the following −

Python (>=3.5)

NumPy (>= 1.11.0)

Scipy (>= 0.17.0)li

Joblib (>= 0.11)

Matplotlib (>= 1.5.1) is required for Sklearn plotting capabilities.

Pandas (>= 0.18.0) is required for some of the scikit-learn examples using data structure and analysis.

Installation

If you already installed NumPy and Scipy, following are the two easiest ways to install scikit-learn −

Using pip

Following command can be used to install scikit-learn via pip −

pip install -U scikit-learn

Using conda

Following command can be used to install scikit-learn via conda −

conda install scikit-learn

On the other hand, if NumPy and Scipy is not yet installed on your Python workstation then, you can install them by using either pip or conda.

Another option to use scikit-learn is to use Python distributions like *Canopy* and *Anaconda* because they both ship the latest version of scikit-learn.

Features

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models.

Feature extraction − It is used to extract the features from data to define the attributes in image and text data.

Feature selection − It is used to identify useful attributes to create supervised models.

Open Source − It is open source library and also commercially usable under BSD license.

13) PySwarms

PySwarms is a Python library that provides optimization algorithms inspired by swarm intelligence, such as Particle Swarm Optimization (PSO) and Differential Evolution (DE). It is used for solving various optimization problems, including global optimization, parameter estimation, feature selection, and more. Here's an overview of how you can use PySwarms in Python:

Installation: You can install PySwarms in Python using the pip package manager. You can run the following command in your Python environment to install PySwarms:

```
pip install pyswarms
```

Importing PySwarms: After installation, you can import PySwarms in your Python script using the following import statement:

```
import pyswarms as ps
```

Defining the Objective Function: The objective function is the function that you want to optimize using PySwarms. You need to define this function in Python, and it should take a vector of parameters as input and return a scalar value that represents the fitness or quality of the solution.

```
# Example of defining an objective function

import numpy as np

def objective_function(x):

    # Calculate the fitness based on the input parameters x

    fitness = np.sum(x**2)

    return fitness
```

Configuring the Optimization Algorithm: PySwarms provides various optimization algorithms, such as PSO, DE, and their variants. You need to configure the algorithm by setting hyperparameters such as the number of particles, the maximum number of iterations, the bounds of the parameter search space, and others.

```
# Example of configuring the PSO algorithm in PySwarms
```

```
options = {'c1': 0.5, 'c2': 0.3, 'w': 0.9}

optimizer = ps.single.GlobalBestPSO(n_particles=10, dimensions=2, options=options)
```

Optimizing the Objective Function: Once the algorithm is configured, you can use it to optimize the objective function. You can call the **optimize()** method of the optimizer object, passing the objective function as an argument. The optimizer will search for the optimal solution in the defined search space, and return the best solution found.

```
# Example of optimizing the objective function with PSO

best_solution, best_fitness = optimizer.optimize(objective_function, iters=100)

print('Best Solution:', best_solution)

print('Best Fitness:', best_fitness)
```

Analyzing the Results: You can analyze the results of the optimization, such as the best solution found, the fitness value of the best solution, and any other relevant information depending on your specific optimization problem.

```
# Example of analyzing the optimization results

# (Assuming a 2-dimensional search space)

x_optimal = best_solution[0]

y_optimal = best_solution[1]

print('Optimal Solution (x, y):', x_optimal, y_optimal)

print('Optimal Fitness:', best_fitness)
```

Visualizing the Results: PySwarms provides visualization tools to help you visualize the optimization process, such as the evolution of the fitness value over iterations, the movement of particles in the search space, and others.

```
# Example of visualizing the optimization process

import matplotlib.pyplot as plt

# Get the fitness history from the optimizer object

fitness_history = optimizer.cost_history

# Plot the fitness history

plt.plot(fitness_history)

plt.xlabel('Iterations')

plt.ylabel('Fitness')

plt.title('Fitness History')

plt.show()
```

That's a brief overview of how you can use PySwarms in Python for optimization problems. PySwarms also provides additional features, such as multiobjective optimization, constrained optimization, and different variants of the PSO and DE algorithms. You can refer to the official documentation of PySwarms for more information on advanced usage and additional features.

Here's a summary of the steps to use PySwarms for optimization in Python:

Install PySwarms using pip:

```
pip install pyswarms
```

Import PySwarms in your Python script:

```
import pyswarms as ps
```

Define your objective function to optimize.

Configure the optimization algorithm by setting hyperparameters such as number of particles, maximum iterations, and search space bounds.

Create an optimizer object using the desired algorithm and hyperparameters.

Call the optimize() method of the optimizer object, passing the objective function as an argument, to perform the optimization.

Analyze the results, such as the best solution found and the fitness value.

Visualize the optimization process using the provided visualization tools.

PySwarms is a powerful tool for solving optimization problems in Python and can be applied in various domains, such as machine learning, data science, engineering, and more.

14)Django

Django is a Web Application Framework which is used to develop web applications.

What is Django

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement.

This framework uses a famous tag line :**The web framework for perfectionists with deadlines.**

By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.

History

Django was design and developed by Lawrence journal world in 2003 and publicly released under BSD license in July 2005. Currently, DSF (Django Software Foundation) maintains its development and release cycle.

Django was released on 21, July 2005. Its current stable version is 2.0.3 which was released on 6 March, 2018.

Popularity

Django is widely accepted and used by various well-known sites such as:

- Instagram
- Mozilla
- Disqus
- Pinterest
- Bitbucket
- The Washington Times

Features of Django

- Rapid Development
- Secure
- Scalable
- Fully loaded
- Versatile
- Open Source
- Vast and Supported Community

Rapid Development

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.

Secure

Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.

Scalable

Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

Fully loaded

Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

Versatile

Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.

Open Source

Django is an open source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.

Vast and Supported Community

Django is an one of the most popular web framework. It has widely supportive community and channels to share and connect.

15)Pymysql

pymysql is a Python library that provides a pure Python interface for connecting to and interacting with MySQL databases. It allows developers to perform various database operations such as querying, inserting, updating, and deleting data in MySQL databases using Python code.

Here's an overview of pymysql and its key features:

MySQL Database Connectivity: pymysql provides a simple and convenient way to connect to MySQL databases from Python code. It supports both Python 2 and Python 3 and can be installed using pip, the Python package manager.

Pure Python Implementation: pymysql is a pure Python implementation of the MySQL protocol, which means it doesn't require any additional dependencies or external libraries to connect to MySQL databases. It provides a native Pythonic interface for working with MySQL databases.

CRUD Operations: pymysql allows developers to perform CRUD (Create, Read, Update, Delete) operations on MySQL databases. It provides functions for executing SQL queries, inserting data into tables, updating data, and deleting data from tables.

Prepared Statements: pymysql supports prepared statements, which allows for secure and efficient handling of SQL queries with user-supplied data. Prepared statements help

prevent SQL injection attacks, which are a common security vulnerability in database applications.

Transaction Management: pymysql supports transaction management, which allows developers to execute multiple database operations as a single unit of work. Transactions provide a way to ensure data consistency and integrity in the database, and pymysql provides functions for starting, committing, and rolling back transactions.

Cursor and Fetching Data: pymysql provides cursor objects that can be used to execute SQL queries and fetch data from MySQL databases. Cursors allow for fetching data in different ways, such as fetching all rows at once or fetching one row at a time.

Connection Pooling: pymysql supports connection pooling, which helps in efficiently managing database connections and reusing them across multiple database operations. Connection pooling can improve the performance of database operations in applications with high concurrent database requests.

Error Handling: pymysql provides robust error handling mechanisms, allowing developers to handle various errors that can occur during database operations. Error handling helps in identifying and resolving issues with database operations and ensures the reliability of the application.

Customizable: pymysql provides many options and configurations that can be customized to suit the specific needs of an application. This includes options for setting character encoding, timeout values, and other MySQL-related settings.

Overall, pymysql is a powerful and easy-to-use Python library for interacting with MySQL databases. It provides a pure Pythonic interface for performing various database operations and offers features such as prepared statements, transaction management, connection pooling, and error handling, making it a popular choice for Python developers working with MySQL databases.

16)Imutils

Imutils is a popular Python library that provides a collection of utility functions for image processing tasks. It aims to simplify common image processing operations, such as resizing, rotating, translating, and displaying images, making it easier for developers to work with images in their Python applications. Here's an overview of imutils and its key features:

1. Image Resizing: imutils provides functions for resizing images, including resizing by a specific width or height, maintaining aspect ratio, and resizing to a target size while preserving the aspect ratio.
2. Image Rotation: imutils offers functions for rotating images by a specific angle, both clockwise and counter clockwise, and rotating images around a specific point or the center of the image.
3. Image Translation: imutils includes functions for translating images by a specified number of pixels in the x and y direction, allowing for image shifting or panning.

4. Image Flip and Mirror: imutils provides functions for flipping images horizontally or vertically, as well as functions for mirroring images along the x or y axis.
5. Image Drawing: imutils offers functions for drawing various shapes on images, such as lines, rectangles, circles, and text.
6. Image Channels and Colo spaces: imutils provides functions for splitting and merging image channels, as well as converting images between different colorspaces, such as RGB, BGR, grayscale, and HSV.
7. Image Filtering: imutils includes functions for applying common image filters, such as Gaussian blur, median blur, and bilateral filter, to smooth or enhance images.
8. Image Thresholding: imutils provides functions for applying thresholding techniques, such as simple thresholding, adaptive thresholding, and Otsu's thresholding, to convert grayscale images to binary images.
9. Image Display: imutils offers functions for displaying images using popular GUI frameworks, such as OpenCV's high gui, matplotlib, and PyQt, making it easy to visualize images in Python applications.
10. Video Processing: imutils includes functions for processing video frames, such as reading frames from video files, displaying video frames, and saving video frames to video files.

Overall, imutils is a handy library for performing common image processing tasks in Python. It provides a collection of utility functions that can simplify image processing operations and make it easier for developers to work with images in their Python applications.

1. Image Operations: imutils also provides functions for performing various image operations, such as bitwise operations (AND, OR, XOR), image blending, and image masking. These operations can be useful for advanced image processing tasks, such as image manipulation, object detection, and image composition.
2. Image Feature Detection: imutils offers functions for detecting image features, such as corners, edges, and keypoints, using popular feature detection algorithms, such as Harris corner detection, Canny edge detection, and SIFT, SURF, or ORB feature detection.
3. Image Matching: imutils provides functions for performing image matching tasks, such as template matching and feature-based matching, using popular

algorithms like Normalized Cross-Correlation (NCC), Brute-Force Matching, and FLANN-based Matching. These functions can be used for tasks such as object recognition, image registration, and image stitching.

4. Image Utilities: imutils includes various utility functions, such as converting between different image data types, checking if an image is grayscale or color, handling image file paths, and generating image filenames based on timestamps, making image processing tasks more convenient and efficient.
5. OpenCV Integration: imutils is designed to work seamlessly with OpenCV, a popular computer vision library in Python. It provides helper functions that complement OpenCV's functionality and can be easily integrated into OpenCV-based projects.
6. Cross-Platform Compatibility: imutils is compatible with different operating systems, such as Windows, macOS, and Linux, and can be used with different Python versions (Python 2 and Python 3).
7. Easy-to-Use: imutils is known for its simple and intuitive API, making it easy for developers to quickly start using it in their image processing projects. The library is well-documented, with comprehensive documentation and examples available, making it easy to learn and use.

Overall, imutils is a versatile and user-friendly library for image processing tasks in Python. It provides a wide range of utility functions for common image processing operations, image feature detection, image matching, and other useful functions that can simplify image processing tasks and accelerate development of computer vision applications.

17)Seaborn

Seaborn is a popular Python data visualization library based on Matplotlib that provides a high-level interface for creating attractive and informative statistical graphics. It is built on top of Matplotlib and provides additional functionality for creating visually appealing and informative plots for data analysis and visualization tasks. Seaborn is particularly well-suited for visualizing complex datasets and making them more accessible to data analysts and scientists. Here are some key features of Seaborn:

Statistical Data Visualization: Seaborn provides a wide range of statistical visualizations, such as scatter plots, line plots, bar plots, histogram, violin plots, box plots, heatmap, pair plots, and many more. These visualizations are designed to provide insights into the underlying data distribution, relationships, and patterns.

Attractive Plot Aesthetics: Seaborn comes with default plot styles that are visually appealing and can be easily customized. It offers a wide range of color palettes, plot themes, and plot styles to create visually appealing plots with minimal effort.

Built-in Statistical Functionality: Seaborn integrates statistical functions from the SciPy library to automatically compute and visualize statistical measures, such as regression lines, confidence intervals, error bars, and probability distributions, within the plot itself. This makes it easy to analyze and interpret data in a visual form.

Categorical and Numerical Data Visualization: Seaborn provides specialized functions for visualizing categorical and numerical data, including functions for creating

categorical plots, point plots, bar plots, and box plots that are specifically designed for handling different types of data.

Multi-plot Grids: Seaborn allows for creating multi-plot grids, such as faceted plots, pair plots, and scatterplot matrices, which enable visualization of multiple variables and their relationships in a single plot.

Support for Complex Datasets: Seaborn is capable of handling complex datasets with multiple variables, and provides advanced visualization techniques such as faceting, hue mapping, and color palettes to create meaningful visualizations for exploring and analyzing such data.

Integration with Pandas: Seaborn can easily be integrated with Pandas, a popular data manipulation library in Python, allowing for seamless data visualization workflows with data stored in Pandas Data Frames.

Customization: Seaborn provides extensive customization options for fine-tuning plots, including setting plot styles, colors, fonts, and other visual elements to create customized and professional-looking visualizations.

Comprehensive Documentation and Community Support: Seaborn has a large community of users and developers, and provides comprehensive documentation, tutorials, and examples, making it easy to learn and use. Additionally, Seaborn is well-maintained and regularly updated with new features and bug fixes.

Overall, Seaborn is a powerful data visualization library in Python that provides an easy-to-use interface for creating attractive and informative statistical graphics. It is well-suited for visualizing complex datasets, exploring data relationships, and gaining insights from data in a visually appealing and meaningful way.

18)Dlib

dlib is a popular open-source C++ library that provides tools and algorithms for various computer vision tasks, including face detection, face recognition, facial landmark detection, object detection, and image processing. dlib is widely used in the field of computer vision and machine learning due to its efficiency, accuracy, and ease of use. In addition to its C++ API, dlib also provides Python bindings, making it accessible for Python developers as well.

Here are some key features of dlib:

Face Detection: dlib provides highly accurate face detection capabilities using its deep learning-based face detection model. It can detect faces in images and videos, even under varying lighting conditions, orientations, and occlusions.

Facial Landmark Detection: dlib can accurately detect facial landmarks, such as the eyes, nose, mouth, and jawline, in facial images. These facial landmarks can be used for tasks such as facial expression analysis, face alignment, and facial feature extraction.

Face Recognition: dlib includes a state-of-the-art deep learning-based face recognition model that can be used to recognize faces in images and videos. It can identify

individuals based on their facial features, making it useful for applications such as face authentication, face verification, and face clustering.

Object Detection: dlib supports object detection using its object detection model, which can detect various objects in images and videos, including cars, pedestrians, and other objects of interest.

Image Processing: dlib provides a variety of image processing functions, such as image resizing, image filtering, color manipulation, and geometric transformations, which can be used for tasks such as image pre-processing, augmentation, and manipulation.

Machine Learning Tools: dlib includes a wide range of machine learning tools, such as support vector machines (SVMs), k-nearest neighbors (KNN), and deep neural networks, which can be used for tasks such as classification, regression, and clustering.

Cross-Platform Compatibility: dlib is cross-platform compatible and can be used on various operating systems, including Windows, macOS, Linux, and embedded systems.

Integration with OpenCV: dlib can be easily integrated with OpenCV, a popular computer vision library, to perform more advanced image processing and computer vision tasks.

Extensive Documentation and Community Support: dlib has extensive documentation, tutorials, and examples available, making it easy to learn and use. It also has a large community of users and developers who provide support and contribute to its development.

Dlib is a powerful and versatile library that provides a wide range of computer vision tools and algorithms for various tasks. Its high accuracy, efficiency, and ease of use make it a popular choice among researchers, developers, and practitioners in the field of computer vision and machine learning.

19)Playsound

playsound is a Python library that provides a simple way to play sound files in various formats, such as WAV, MP3, and OGG. It allows you to play sound files directly from your Python scripts or Jupyter notebooks without the need for external media players.

Here are some key features of **playsound**:

1. Easy to Use: **playsound** provides a simple and easy-to-use interface for playing sound files in Python. It does not require any complicated setup or configuration, making it accessible for beginners and experienced Python developers alike.
2. Cross-Platform Compatibility: **playsound** is cross-platform compatible and can be used on various operating systems, including Windows, macOS, and Linux.
3. Support for Multiple Audio Formats: **playsound** supports popular audio formats such as WAV, MP3, and OGG, allowing you to play sound files in different formats depending on your requirements.
4. Blocking and Non-Blocking Playback: **playsound** allows you to play sound files in blocking or non-blocking mode. In blocking mode, the playback is synchronous, meaning that the execution of your script is paused until the sound

file finishes playing. In non-blocking mode, the playback is asynchronous, allowing your script to continue executing while the sound file is playing.

5. Simple Syntax: **playsound** provides a simple syntax for playing sound files, with just one function call to play a sound file, making it easy to incorporate sound effects, background music, or audio feedback in your Python applications.
6. Error Handling: **playsound** provides basic error handling for handling cases when a sound file is not found or cannot be played due to audio device issues. This helps you handle exceptions gracefully and provide a better user experience in your applications.
7. Lightweight: **playsound** is a lightweight library that does not require any external dependencies, making it easy to install and use in your Python environment.
8. Extensive Documentation: **playsound** has extensive documentation, including usage examples and troubleshooting guides, to help you get started quickly and troubleshoot any issues that may arise during usage.

Overall, **playsound** is a simple and handy library for playing sound files in Python, providing an easy-to-use interface, cross-platform compatibility, and support for multiple audio formats. It is particularly useful for applications that require playing sound effects, background music, or audio feedback, such as multimedia applications, games, and interactive user interfaces.

To install the **playsound** library in Python, you can use the following steps:

Open a terminal or command prompt in your operating system.

If you have Python and pip (Python package manager) installed, you can directly install **playsound** using the following pip command:

```
pip install playsound
```

Note: If you're using Python 3, you may need to use pip3 instead of pip.

Wait for the installation to complete. Pip will download and install the playsound library along with any dependencies.

Once the installation is complete, you can import playsound in your Python script or Jupyter notebook using the import statement:

```
import playsound
```

That's it! You have successfully installed playsound in your Python environment and can now use it to play sound files in your applications.

Note: Depending on your operating system and audio settings, you may need to configure additional audio settings or have appropriate audio drivers installed to ensure proper playback of sound files using playsound. Please refer to the playsound documentation and your operating system's audio settings for further information.

CHAPTER – 3

LITERATURE SURVEY

The chapter gives brief information about literature survey in the project

3. LITERATURE SURVEY

3. 1. Literature Survey

A. Safe Driving By Detecting Lane Discipline and Driver Drowsiness-[Yashika Katyall,Suhas Alur,Shipra Dwivedi,(2014)]

This paper presents a real time lane detection and driver fatigue or driver drowsiness detection system Road accidents have become all too prevalent in today's environment. They not only inflict property damage, but they also put people's lives in danger while travelling. Given its extent and the resulting negative effects on the economy, public health, safety, and the general welfare of the people, road safety is a national priority. Rough driving, drunk driving, inexperience, jumping signals, and disregarding signboards are all possible causes of road accidents. The paper is divided into two sections. To begin, the Hough Transform is used to detect lanes. Second, driver eye detection for sleepiness detection. As a result, the attention is mostly on the driver's weariness and adherence to lane discipline (Katyall, Alur, & Dwivedi, 2014).

B. *Monitering Driver's Drowsiness Status at Night Based on Computer Vision-[Vidhu Valsan A,Paul P Mathai,Lerin Babu(2021)]*

This research describes a real-time tiredness driving detection system that operates at night. The location of facial landmarks on the driver's face is determined by employing one shape predictor and then computing eye aspect ratio, mouth opening ratio, and yawning frequency.

The values of these parameters are used to identify drowsiness.

The thresholds are established using an adaptive thresholding approach. Offline implementation of machine learning techniques was also done. The proposed approach was tested in both real-time and on the Face Dataset. The system's accuracy and robustness are demonstrated by the experimental findings (Valsan, Paul, & Babu, 2021).

C. *Intelligent Driver Drowsiness Detection through Fusion of Yawning and Eye Closure [M. Omidyeganeh, A. Javadtalab, S. Shirmohammadi,(2011)]*

Drowsy driving is a key component in the majority of car accidents. We describe a robust and clever approach for detecting driver tiredness in this study, which combines eye closure and yawning detection methods. A camera fitted in the automobile captures the driver's face look in this method. The eye and mouth portions of the face are then removed and examined for symptoms of driver tiredness. Finally, during the fusion phase, the driver's condition is assessed, and if sleepiness is identified, a warning message is issued to the driver. Our tests show that the proposed approach is quite effective (Mandal, Li, Wang, & Jie, 2016).

D. *Portable Prevention and Monitoring of Driver's Drowsiness Focuses to Eyelid Movement Using Internet of Things [Menchie Miranda, Alonica Villanueva, Mark Jomar Buo, Reynald Merabite, Sergio Paulo Perez, John Michael Rodriguez,(2018)]*

Since the number of vehicular accidents in the Philippines has been increasing year after year, this paper offers a sleepiness prevention device. Current safety measures are used

to boost driver awareness, such as the construction of standard rumble strips on highways, GPS, speed limiters, sensors, and other research that employ signal processing incorporated in a costly car. The system makes use of the internet of things to allow the car owner to keep track of the driver's tiredness at all times throughout working hours. The current study focuses on eyelid movement, which was not covered in the prior study. This suggested system continually detects the driver's eyelid movements, and if sleepiness is identified, the gadget notifies him with a random-typed sound. It sends the report to the car owner automatically via internet access from the web application (Miranda, Villanueva, Buo, & Merabite, 2018).

3. 2. Summary Of Literature Survey

Driver drowsiness monitoring systems have gained significant attention in recent years as a promising approach to prevent accidents caused by drowsy driving. Researchers and engineers have proposed various methods and techniques to accurately and reliably detect driver drowsiness in real-time. The following is a literature survey highlighting some of the key research and advancements in the field of driver drowsiness monitoring systems:

Physiological-based approaches: Many studies have explored the use of physiological signals, such as heart rate, electroencephalogram (EEG), skin conductance, and eye activity, to monitor driver drowsiness. For example, heart rate variability (HRV) has been widely used as a reliable indicator of drowsiness, with changes in HRV patterns correlated with drowsy states. EEG-based methods, such as spectral analysis and event-related potentials (ERPs), have also shown promise in detecting drowsiness by analyzing brainwave patterns. Skin conductance, which reflects changes in skin sweat gland activity, has been used to measure physiological arousal levels related to drowsiness. Eye activity, such as blink rate, eyelid closure duration, and pupillometry, has also been investigated as potential indicators of drowsiness.

Vision-based approaches: Vision-based methods have been extensively studied for driver drowsiness detection. Computer vision techniques, such as facial feature tracking, eye tracking, and head pose estimation, have been used to monitor driver behavior and detect signs of drowsiness, such as drooping eyelids, yawning, and changes in facial expressions. Machine learning algorithms, such as support vector machines (SVM), deep neural networks (DNN), and convolutional neural networks (CNN), have been used to analyze visual cues and classify drowsiness levels based on facial and eye movements.

Sensor fusion approaches: Several studies have proposed sensor fusion approaches that combine multiple modalities, such as physiological signals, vision-based cues, and vehicle-related information, to improve the accuracy and robustness of driver drowsiness detection. For example, integrating physiological signals with vision-based cues, such as eye movements and facial expressions, can provide a more comprehensive and reliable assessment of drowsiness. In addition, incorporating vehicle-related information, such as steering wheel movements, lane deviation, and vehicle speed, can provide additional context and improve the accuracy of drowsiness detection.

Real-time monitoring and alerting: Real-time monitoring and alerting are crucial for effective driver drowsiness detection systems. Many studies have proposed methods to provide timely alerts to drivers when their drowsiness level reaches a dangerous

threshold. Visual alerts, such as blinking lights on the dashboard or head-up displays, and auditory alerts, such as alarms or voice prompts, have been used to alert drivers and prompt them to take corrective actions, such as taking a break or changing their driving behavior.

Challenges and limitations: Despite significant advancements in driver drowsiness monitoring systems, there are several challenges and limitations that need to be addressed. Variability in drowsiness patterns among individuals, the dynamic nature of drowsiness levels, and the influence of external factors, such as road conditions and weather, pose challenges to accurate and reliable drowsiness detection. Ensuring user acceptance, privacy concerns related to physiological signal collection, system integration with different types of vehicles, and cost-effective implementation of the system are also important considerations.

In conclusion, driver drowsiness monitoring systems have been extensively researched, and various approaches, including physiological-based, vision-based, and sensor fusion methods, have been proposed to detect driver drowsiness in real-time. These systems have the potential to significantly reduce accidents caused by drowsy driving and improve road safety. However, further research is needed to address the challenges and limitations of these systems and ensure their effectiveness in real-world driving scenarios. Future research directions may include exploring advanced machine learning algorithms, incorporating additional physiological signals or vehicle-related information, developing standardized evaluation metrics, and conducting large-scale field studies to validate the performance and usability of driver drowsiness monitoring systems. Additionally, addressing user acceptance, privacy concerns, and cost-effective implementation strategies will be crucial for the successful adoption of these systems in commercial vehicles and everyday driving scenarios. Overall, the literature survey highlights the significant progress made in the field of driver drowsiness monitoring systems while identifying the challenges and opportunities for future research and development in this area.

CHAPTER - 4

SOFTWARE REQUIREMENT SPECIFICATION

Gives the details of platform specifications, Hardware,and Software specifications.

CHAPTER 4

4. REQUIREMENT ANALYSIS

This chapter provides the details of the project's need based survey, system requirements, Hardware Requirements, Software Requirements, and System Requirements.

Project Scope :-

Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio, mouth opening ratio and nose length ratio are computed and depending on their values, drowsiness is detected based on developed adaptive thresholding. Machine learning algorithms have been implemented as well in an offline manner. A sensitivity of 95.58% and specificity of 100% has been achieved in Support Vector Machine based classification.

4.1 Existing System :

Now a days maximum members are using vechile(car, lorry,bus).according to survey 10 to 15% are accidents are accruing because of the driver was in sleepy mode.No software is having to give alert to the driver

Disadvantages :-

- More Accidents are accruing
- Unable to give alert while driver was sleepy.

4.2 Proposed System :

A block diagram of the proposed driver drowsiness monitoring system has been depicted in Fig 1. At first, the video is recorded using a webcam. The camera will be positioned in front of the driver to capture the front face image. From the video, the frames are extracted to obtain 2-D images. Face is detected in the frames using histogram of oriented gradients (HOG) and linear support vector machine (SVM) for object detection [10]. After detecting the face, facial landmarks like positions of eye, nose, and mouth are marked on the images. From the facial landmarks, eye aspect ratio, mouth opening ratio and position of the head are quantified and using these features and machine learning approach, a decision is obtained about the drowsiness of the driver. If drowsiness is detected, an alarm will be sent to the driver to alert him/her. The details of each block are discussed below

Advantages :

Provide alert to the driver.

Decrease the accidents.

CHAPTER - 5

SYSTEM REQUIREMENTS

Gives the details of platform specifications, Hardware,and Software specifications.

5. SYSTEM REQUIREMENTS

5.1 Hardware Requirements :-

1) Operating System supported by
    a. Windows 7
    b. Windows XP
    c. Windows 8
2) Processor – Pentium IV or higher
3) RAM -- 256 MB
4) Space on Hard Disk -- Minimum 512 MB

5.2 Software Requirements :-

1. For developing the Application
    a. Python
    b. Django
    c. Mysql
    d. Mysqlclient
    e. WampServer 2.4
1. Technologies and Languages used to Develop
    a. Python

1. HOME PAGE:-

- XML

- JAVA

2. REGISTRATIONPAGE:-

- XML

- JAVA

3. LOGIN PAGE:-

- XML

- JAVA

4. BOOKINGS PAGE

- XML
- JAVA

5. REFERRALS PAGE

- XML

- JAVA

6. PROFILE PAGE

- XML

7. PREFERENCE PAGE

- XML

- JAVA

8. SP LOGIN PAGE

- XML

- JAVA

9. SP SIGN UP PAGE

- XML

- JAVA

10. SP BOOKING PAGE

- XML

- JAVA

Functional requirements :-

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.

Non –Functional Requirements :-

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

CHAPTER – 6

ANALYSIS & DESIGN

This chapter gives the details of the system and data design.

CHAPTER-6

6. DESIGN PHASE

INTRODUCTION

This chapter provides the design phase of the Application. To design the project, we use the UML diagrams. The Unified Modelling Language (UML) is a general- purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

6.1 USE CASE DIAGRAM

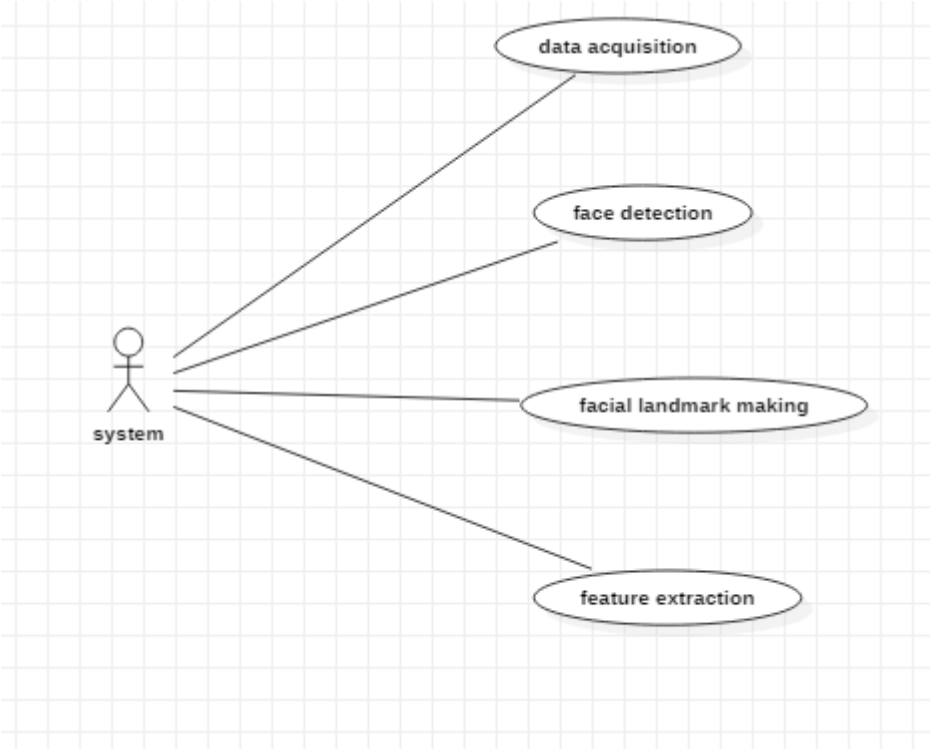


Fig 6.1 Use case Diagram

The use case diagram is used to represent all the functional use cases that are involved in the project.

The above diagram represents the main two **actors** in the project, they are

- User

6.2 CLASS DIAGRAM

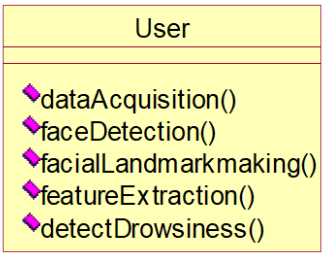


Fig 6.2 class diagram

The above mentioned class diagram represents the Chatbot system workflow model. This diagram has class models with class names as

1. User
2. Home screen

6.3 SEQUENCE DIAGRAM

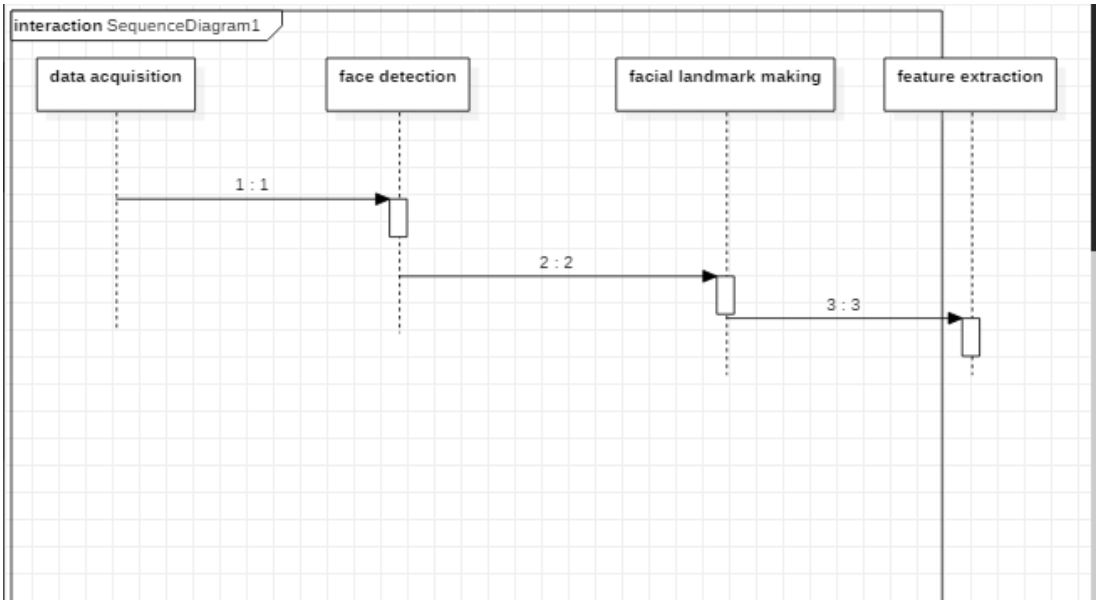
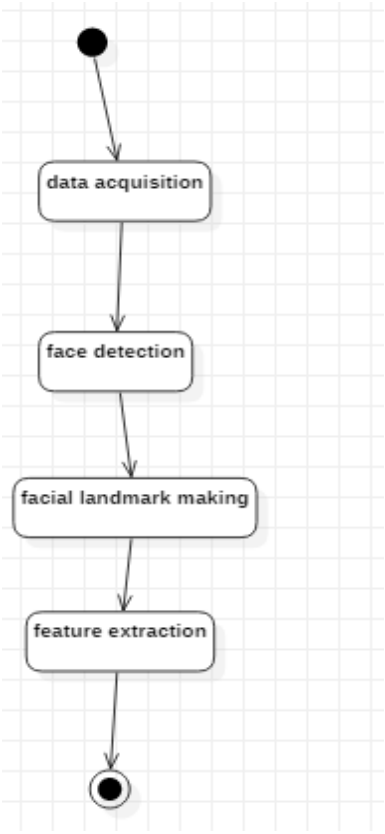


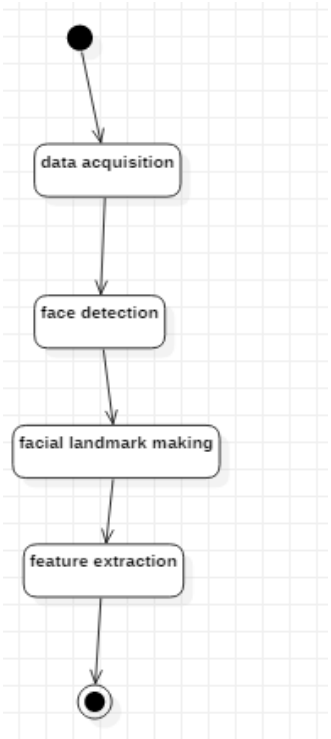
Fig 6.3 sequence diagram

The above diagram represents the sequence of flow of actions in the system.

6.4 ACTIVITY DIAGRAM



6.5 STATE CHART DIAGRAM:-



6.6 DATA DESIGN

Name	Description
Users	Contains all the registered user details.
View Face detection	All the registered service provider details.
Services	Contains all the types of services available.

Table 6.1 List of Database Tables

6.7 CONCLUSION

In this project, a low cost, real time driver drowsiness monitoring system has been proposed based on visual behavior and machine learning. Here, visual behavior features like eye aspect ratio, mouth opening ratio and nose length ratio are computed from the streaming video, captured by a webcam. An adaptive thresholding technique has been developed to detect driver drowsiness in real time. The developed system works accurately with the generated synthetic data. Subsequently, the feature values are stored and machine learning algorithms have been used for classification. Bayesian classifier, FLDA and SVM have been explored here. It has been observed that FLDA and SVM outperform Bayesian classifier. The sensitivity of FLDA and SVM is 0.896 and 0.956 respectively whereas the specificity is 1 for both. As FLDA and SVM give better accuracy, work will be carried out to implement them in the developed system to do the classification (i.e., drowsiness detection) online. Also, the system will be implemented in hardware to make it portable for car system and pilot study on drivers will be carried out to validate the developed system.

CHAPTER - 7
SYSTEM LOWLEVEL DESIGN

This chapter gives an overview of all modules in the project.

CHAPTER 7

SYSTEM LOWLEVEL DESIGN

This chapter mainly provides the overview on modules of the application, objectives of the project and a detailed project overview.

7.1 Modules of the Application:

Data Acquisition :-

The video is recorded using webcam (Sony CMU-BR300) and the frames are extracted and processed in a laptop. After extracting the frames, image processing techniques are applied on these 2D images. Presently, synthetic driver data has been generated. The volunteers are asked to look at the webcam with intermittent eye blinking, eye closing, yawning and head bending. The video is captured for 30 minutes duration.

Face Detection :-

After extracting the frames, first the human faces are detected. Numerous online face detection algorithms are there. In this study, histogram of oriented gradients (HOG) and linear SVM method is used. In this method, positive samples of descriptors are computed on them. Subsequently, negative samples (samples that do not contain the required object to be detected i.e., human face here) of same size are taken and HOG descriptors are calculated. Usually the number of negative samples is very greater than number of positive samples. After obtaining the features for both the classes, a linear SVM is trained for the classification task. To improve the accuracy of VM, hard negative mining is used. In this method, after training, the classifier is tested on the labeled data and the false positive sample feature values are used again for training purpose. For the test image, the fixed size window is translated over the image and the classifier computes the output for each window location. Finally, the maximum value output is considered as the detected face and a bounding box is drawn around the face. This non-maximum suppression step removes the redundant and overlapping bounding boxes.

Facial Landmark marking :-

After detecting the face, the next task is to find the locations of different facial features like the corners of the eyes and mouth, the tip of the nose and so on. Prior to that, the face images should be normalized in order to reduce the effect of distance from the camera, non-uniform illumination and varying image resolution. Therefore, the face image is resized to a width of 500 pixels and converted to grayscale image. After image normalization, ensemble of regression trees is used to estimate the landmark positions on face from a sparse subset of pixel intensities. In this method, the sum of square error loss is optimized using gradient boosting learning. Different priors are used to find different structures. Using this method, the boundary points of eyes, mouth and the central line of the nose are marked and the number of points for eye, mouth and nose are given in Table I. The facial landmarks are shown in Fig 2. The red points are the detected landmarks for further processing.

D. Feature Extraction

After detecting the facial landmarks, the features are computed as described below. Eye aspect ratio (EAR): From the eye corner points, the eye aspect ratio is calculated as the ratio of height and width of the eye as given by

Classification :-

After computing all the three features, the next task is to detect drowsiness in the extracted frames. In the beginning, adaptive thresholding is considered for classification. Later, machine learning algorithms are used to classify the data. For computing the threshold values for each feature, it is assumed that initially the driver is in complete awake state. This is called setup phase. In the setup phase, the EAR values for first three hundred (for 10s at 30 fps) frames are recorded. Out 4 of these three hundred initial frames containing face, average of 150 maximum values is considered as the hard threshold for EAR. The higher values are considered so that no eye closing instances will be present. If the test value is less than this threshold, then eye closing (i.e., drowsiness) is detected. As the size of eye can vary from person to person, this initial setup for each person will reduce this effect. Similarly, for calculating threshold of MOR, since the mouth may not be open to its maximum in initial frames (setup phase) so the threshold is taken experimentally from the observations. If the test value is greater than this threshold then yawn (i.e., drowsiness) is detected. Head bending feature is used to find the angle made by head with respect to vertical axis in terms of ratio of projected nose lengths. Normally, NLR has values rom 0.9 to 1.1 for normal upright position of head and it increases or decreases when head bends down or up in the state of drowsiness. The average nose length is computed as the average of the nose lengths in the setup phase assuming that no head bending is there. After computing the threshold values, the system is used for testing. The system detects the drowsiness if in a test frame drowsiness is detected for at least one feature. To make this thresholding more realistic, the decision for each frame depends on the last 75 frames. If at least 70 frames (out of those 75) satisfy drowsiness conditions for at least one feature, then the system gives drowsiness detection indication and the alarm.

7.2 OBJECTIVES OF THE PROJECT

In this project by monitoring Visual Behaviour of a driver with webcam and machine learning SVM (support vector machine) algorithm we are detecting Drowsiness in a driver. This application will use inbuilt webcam to read pictures of a driver and then using OPENCV SVM algorithm extract facial features from the picture and then check whether driver in picture is blinking his eyes for consecutive 20 frames or yawning mouth then application will alert driver with Drowsiness messages. We are using SVM pre-trained drowsiness model and then using Euclidean distance function we are continuously checking or predicting EYES and MOUTH distance closer to drowsiness, if distance is closer to drowsiness then application will alert driver.

CHAPTER - 8

IMPLEMENTATION

The chapter gives the details of the implementation.

CHAPTER 8
IMPLEMENTATION

This chapter mainly provides the implementation of the project.

8.2 Screen Captures

8.2.1 User Login Screen:

To run this project double click on ‘run.bat’ file to get below screen

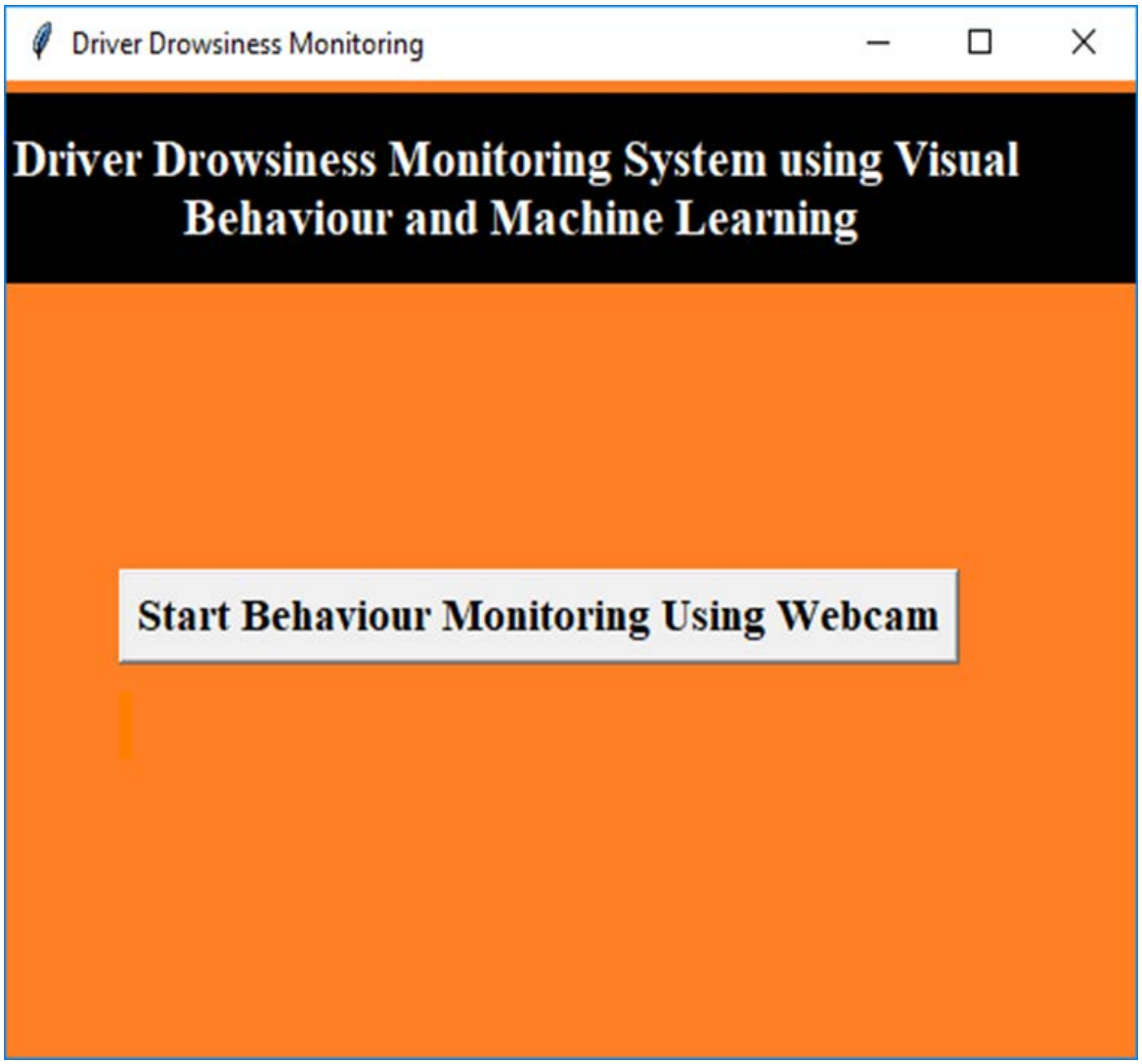


Fig 8.1 To Start Behvaiour Monitoring Using Webcam

In above screen click on ‘Start Behaviour Monitoring Using Webcam’ button to connect application with webcam, after clicking button will get below screen with webcam streaming

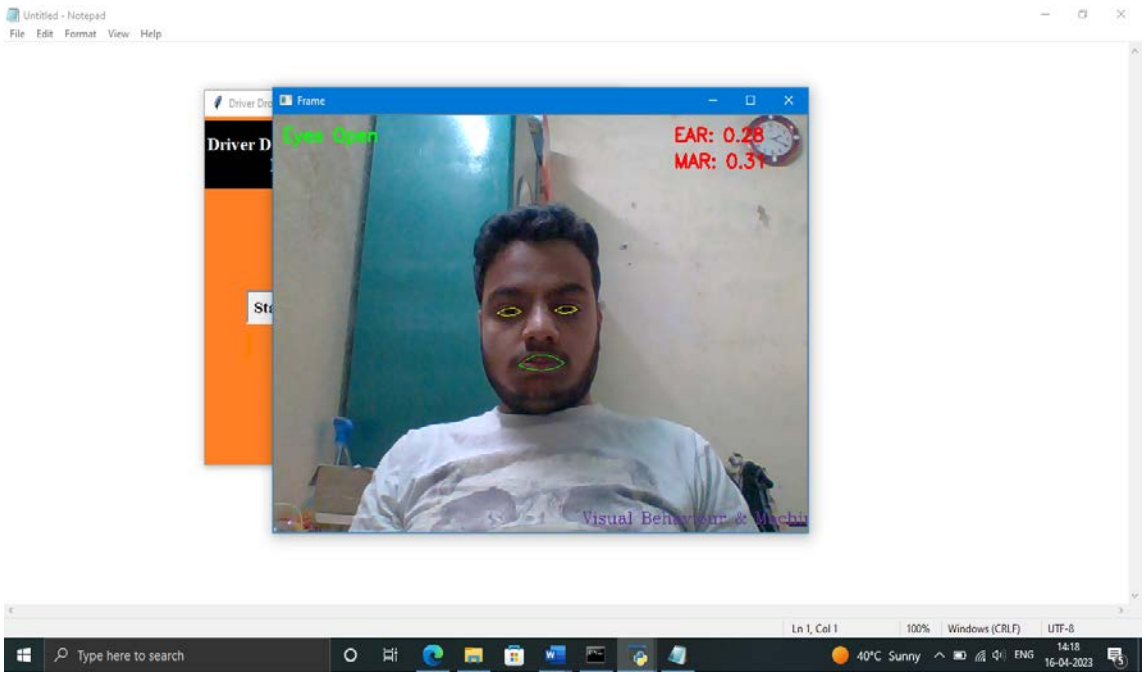


Fig 8.2 To Check Eyes are Open or Closed

In above screen we can see web cam stream then application monitor all frames to see person eyes are open or not, if closed then will get below message

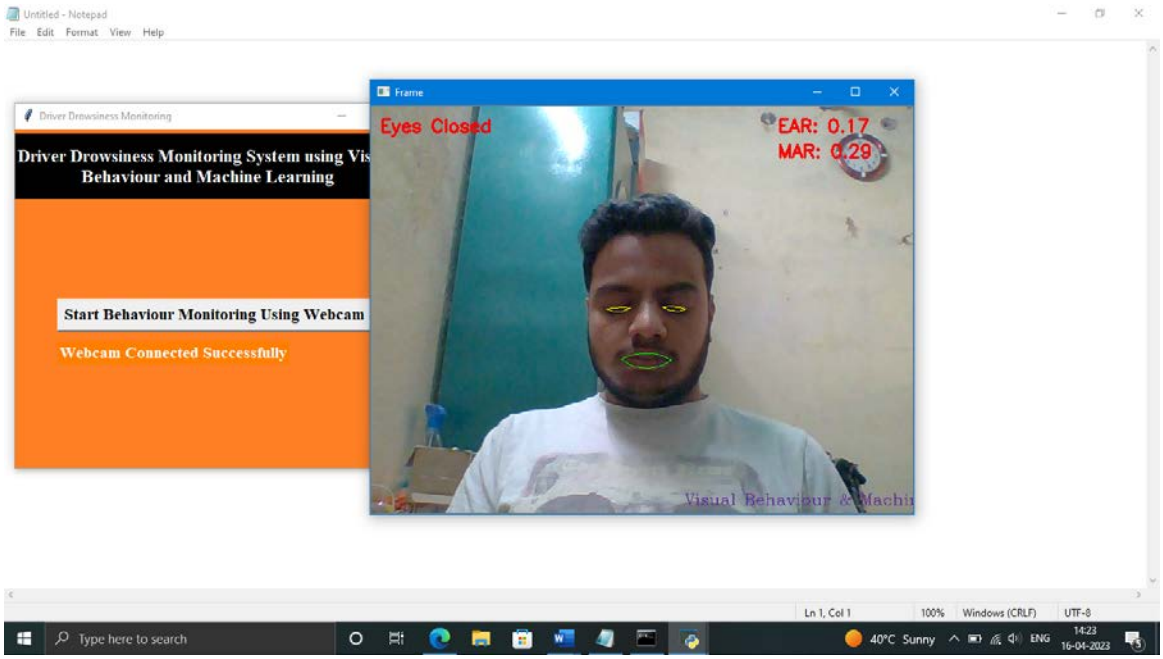


Fig 8.3 To Check Drowsiness Alert

Similarly if mouth starts yawn then also will get alert message

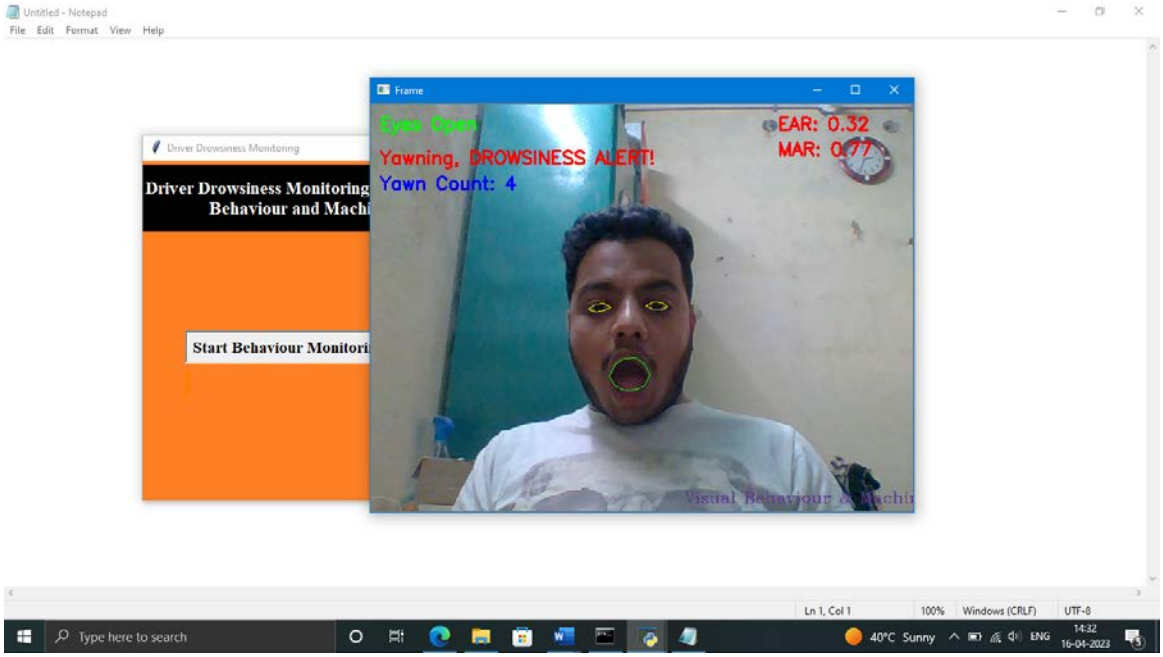


Fig 8.4 To Check Yawning Count

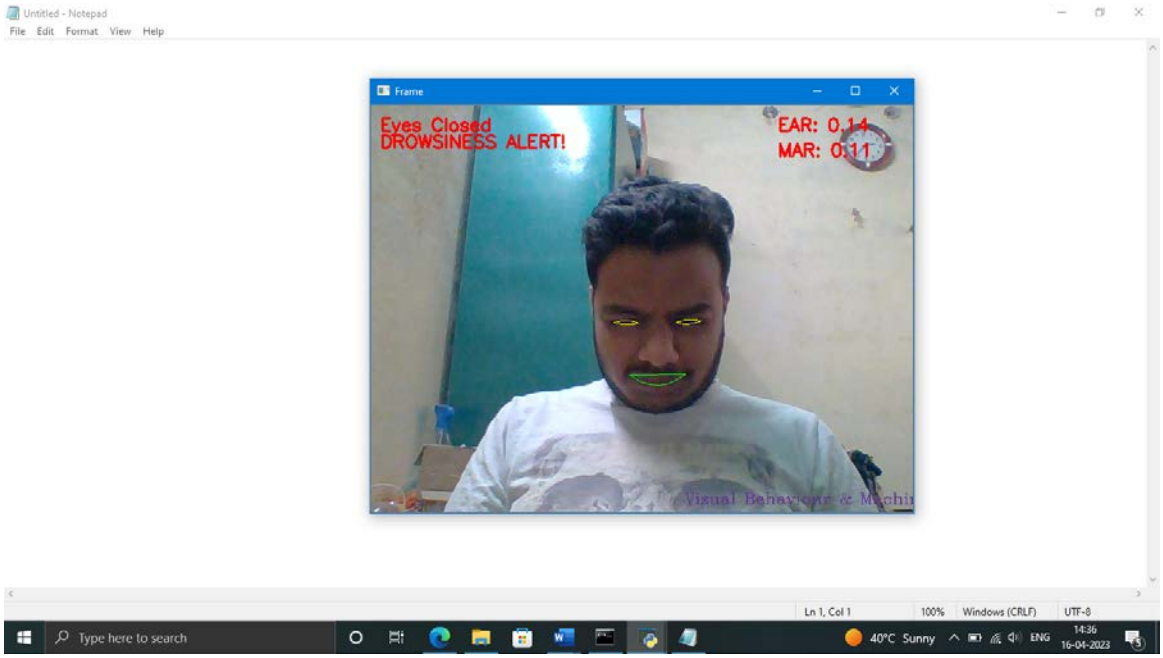


Fig 8.5 To Check Drowsiness Alert or Not

CHAPTER - 9

TESTING

The chapter shows the various test cases.

CHAPTER 9

9.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among these performance testing and load testing are the most important one for an android application and next sections deal with some of these types.

9.2 Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing ,and specification-based testing.

9.3White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

9.4Performance Testing

Performance testing is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

9.5 Load Testing

Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

9.6Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

Test Case 1	
Test Case Name	Empty login fields testing
Description	In the login screen if the username and password fields are empty
Output	Login fails showing an alert box asking to enter username and password.

Table 9.1 Test Case for Empty Login Fields

Test Case 2	
Test Case Name	Wrong login fields testing
Description	A unique username and password are set by administrator. On entering wrong username or password gives.
Output	Login fails showing an alert box username or password incorrect.

Table 9.2 Test Case for Wrong Login Fields

Test Case 3	
Test Case Name	User Signup Fails.
Description	User signup need to provide all data.
Output	Signup Fails and an alert message appears asking to enter valid email and name.

Table 9.3 Test Case for Signup fail

CHAPTER - 10
RESULTS &CHALLENGES

The chapter describes the results and challenges faced in the project.

CHAPTER 10

RESULTS AND CHALLENGES

10.1 Results

The current android application is developed using Xml, Java, SQL with Firebase connectivity. It can be used by every individual who are in a need of fulfilling their household services.

At the time of submission of my application was capable of doing the following:

1. Displaying the home screen with different fragments.
2. Authentication of user by using login screen using Firebase.
3. Home screen to display based on user or service provider.
4. After successful login of user, they can choose the service and book a slot of their particular service provider from the displayed list.
5. Add, update, view, delete the user details.
6. After successful login of service provider, they can view all the bookings that are booked by the users and can attend them one by one.
7. Service provider can also set his preferences to not available, if he's too busy or many users had already booked him.
8. Service provider has the ability to change their particular radius of location for servicing.
9. He can set up to 10 km radius.
10. Logout and end the session.

10.2 Challenges

1. Understanding the connections of SQLite Database is a tricky part and confusing when dealing with multiple tables within a database.
2. Making exact orientation API design levels was a difficult task as there are many types of devices like desktop, tablet, mobile with varying screen size and resolutions.
3. Implementing synchronization with Firebase was a challenging task.
4. Learning different technologies and frameworks with little guidance.

CHAPTER - 11

CONCLUSIONS & FUTURE WORK

The chapter gives brief conclusion about the project.

CHAPTER 11

CONCLUSION

11.1 Conclusion

In this project, a low cost, real time driver drowsiness monitoring system has been proposed based on visual behavior and machine learning. Here, visual behavior features like eye aspect ratio, mouth opening ratio and nose length ratio are computed from the streaming video, captured by a webcam. An adaptive thresholding technique has been developed to detect driver drowsiness in real time. The developed system works accurately with the generated synthetic data. Subsequently, the feature values are stored and machine learning algorithms have been used for classification. Bayesian classifier, FLDA and SVM have been explored here. It has been observed that FLDA and SVM outperform Bayesian classifier. The sensitivity of FLDA and SVM is 0.896 and 0.956 respectively whereas the specificity is 1 for both. As FLDA and SVM give better accuracy, work will be carried out to implement them in the developed system to do the classification (i.e., drowsiness detection) online. Also, the system will be implemented in hardware to make it portable for car system and pilot study on drivers will be carried out to validate the developed system.

11.2 Scope for future work

A system in which density of traffic is measured by comparing captured image with real time traffic information against the image of the empty road as reference image is proposed. Each lane will have a minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled

11.3 Limitations

Intelligent Video-Based Drowsy Driver Detection System under Various Illuminations and Embedded Software Implementation

An intelligent video-based drowsy driver detection system, which is unaffected by various illuminations, is developed in this study. Even if a driver wears glasses, the proposed system detects the drowsy conditions effectively. By a near-infrared-ray (NIR) camera, the proposed system is divided into two cascaded computational procedures: the driver eyes detection and the drowsy driver detection. The average open/closed eyes detection rates without/with glasses are 94% and 78%, respectively, and the accuracy of the drowsy status detection is up to 91%. By implementing on the FPGA-based embedded platform, the processing speed with the 640×480 format video is up to 16 frames per second (fps) after software optimizations

"**Driver Fatigue Detection based on Eye Tracking and Dynamic Template Matching"**

A vision-based real-time driver fatigue detection system is proposed for driving safely. The driver's face is located, from color images captured in a car, by using the characteristic of skin colors. Then, edge detection is used to locate the regions of eyes. In addition to being used as the dynamic templates for eye tracking in the next frame, the obtained eyes' images are also used for fatigue detection in order to generate some warning alarms for driving

safety. The system is tested on a Pentium III 550 CPU with 128 MB RAM. The experiment results seem quite encouraging and promising. The system can reach 20 frames per second for eye tracking, and the average correct rate for eye location and tracking can achieve 99.1% on four test videos. The correct rate for fatigue detection is l00%, but the average precision rate is 88.9% on the test videos.

"Monitoring Driver Fatigue using Facial Analysis Techniques"

In this paper, we describe a non-intrusive vision-based system for the detection of driver fatigue. The system uses a color video camera that points directly rewards the driver's face and monitors the driver's eyes in order to detect micro-sleeps (short periods of sleep). The system deals with skin-color information in order to search for the face in the input space. After segmenting the pixels with skin like color, we perform blob processing in order to determine the exact position of the face. We reduce the search space by analyzing the horizontal gradient map of the face, taking into account the knowledge that eye regions in the face present a great change in the horizontal intensity gradient. In order to find and track the location of the pupil, we use gray scale model matching. We also use the same pattern recognition technique to determine whether the eye is open or closed. If the eyes remain closed for an abnormal period of time (5-6 sec), the system draws the conclusion that the person is falling asleep and issues a warning signal.

"The Steps of Proposed Drowsiness Detection System Design based on Image Processing in Simulator Driving "

Drowsiness detection has many implications including reducing roads traffic accidents importance. Using image processing techniques is amongst the new and reliable methods in sleepy face. The present pilot study was done to investigate sleepiness and providing images of drivers' face, employing virtual-reality driving simulator. In order to detecting level of sleepiness according to the signal, information related to 25 drivers was recorded with imaging rate of 10 fps. Moreover, on average 3000 frames was analysed for each driver. The frames were investigated by transforming in grey scale space and based on the Cascade and Viola & Jones techniques and the images characteristics were extracted using Binary and Histogram methods. The MPL neural network was applied for analysing data.70% of information related to each driver were inserted to the network of which 15% for test and 15% for validation. In the last stage the accuracy of 93% of the outputs were evaluated. The intelligent detection and usage of various criteria in long-term time frame are of the advantages of the present study, comparing to other researches. This is helpful in early detection of sleepiness and prevents the irrecoverable losses by alarming

BIBLIOGRAPHY

Code snippets for any errors http://stackoverflow.com/

Android Development Guide https://www.udemy.com/android

Xml and Layout Guide https://www.androidhive.com/

Connecting to Firebase Docs https://firebase.google.com

Software Testing http://en.wikipedia.org/wiki/Software_testing

Manual Testing http://en.wikipedia.org/wiki/Manual_testing

Performance Testing http://en.wikipedia.org/wiki/Software_performance_testing