TED UNIVERSITY

CMPE 492

Low-Level Design Report

Project Name: Drive Safe-Off

Team Members

Gökçe BEKAR

Rabia Esra ŞENDUR

Rumeysa OMAY

Oğuzhan Uğur SARISAKALOĞLU

Supervisor

Venera ADANOVA

Jury Members

Aslı GENÇTAV

Orkunt SABUNCU

Table of Contents

Low-Level Design Report

1. Introduction

Driving is a task that requires the driver's full attention. Whether the driver is fully focused on the task, it can reduce the ability to detect and react to risky situations that exist or occur during driving. Drowsiness is one of the unsafe behaviors that drivers should not do while driving.

Williamson et al. conducted a study in a driving simulator and found that drivers who reported higher levels of sleepiness were more likely to have crashes and centerline crossings. In addition, slower reaction times have been associated with drowsy drivers in studies by He et al.

Drive Safe-Off is a system developed to solve this problem. It is a program that detects behaviors that indicate drowsiness by the driver - such behaviors, eye closure for a certain period of time, yawning for a long time - and warns during driving. By doing this, it aims to reduce and prevent the accidents that may occur and the deaths that may occur as a result of these accidents.

1.1 Object design trade-off

Given that drowsiness significantly increases the risk of accidents, we decided to design a system that can instantly detect drowsiness and send a warning to the driver. In order to achieve this, we decided to examine the driver's facial, eye and mouth movements in real-time while driving. Among these movements, features such as the number of blinking, eye aspect ratio, mouth aspect ratio, and yawning frequency are being examined. When any drowsiness is detected, the necessary warning is sent to the driver to prevent any accident that may be caused by the drowsiness.

Considering that the driver will be within certain physical limits and her/his mobility will be limited during driving, we used a suitable dataset for two purposes. Therefore, we used it for the detection & tracking part and the feature of the eye and mouth parts.

First of all, we detected the face and the eyes of the driver by Haar feature-based Cascade Classifiers. In addition, we implemented the tracking process for the face of the driver, taking into account the situations where detection might be insufficient. If the person's face is blocked by an object or the person turns their head in another direction while the face detector is running in a video, the face detector will likely fail since the Haar cascade classifier will not be able to recognize the face. Therefore, a good tracking algorithm will solve the problem when faced with this kind of situation.

Secondly, we used the same datasets to be able to use anomaly detection algorithms. Thanks to the various videos in the dataset, we were able to examine different types of frames. As a result, we found and observed that situations such as the intensity of light, turning the head rapidly or direction of the head cause the anomaly.

1.2 Interface documentation guidelines

Every class has a **name**, **description**, **attributes**, **methods** and **package** or packages that it relates to. The names of the classes are not established on a specific rule. Class names are made up of nouns, or strings of nouns that best express the function of the class. Methods are named with verb phrases, fields, and parameters with noun phrases.

1.3 Engineering standards (e.g., UML and APA)

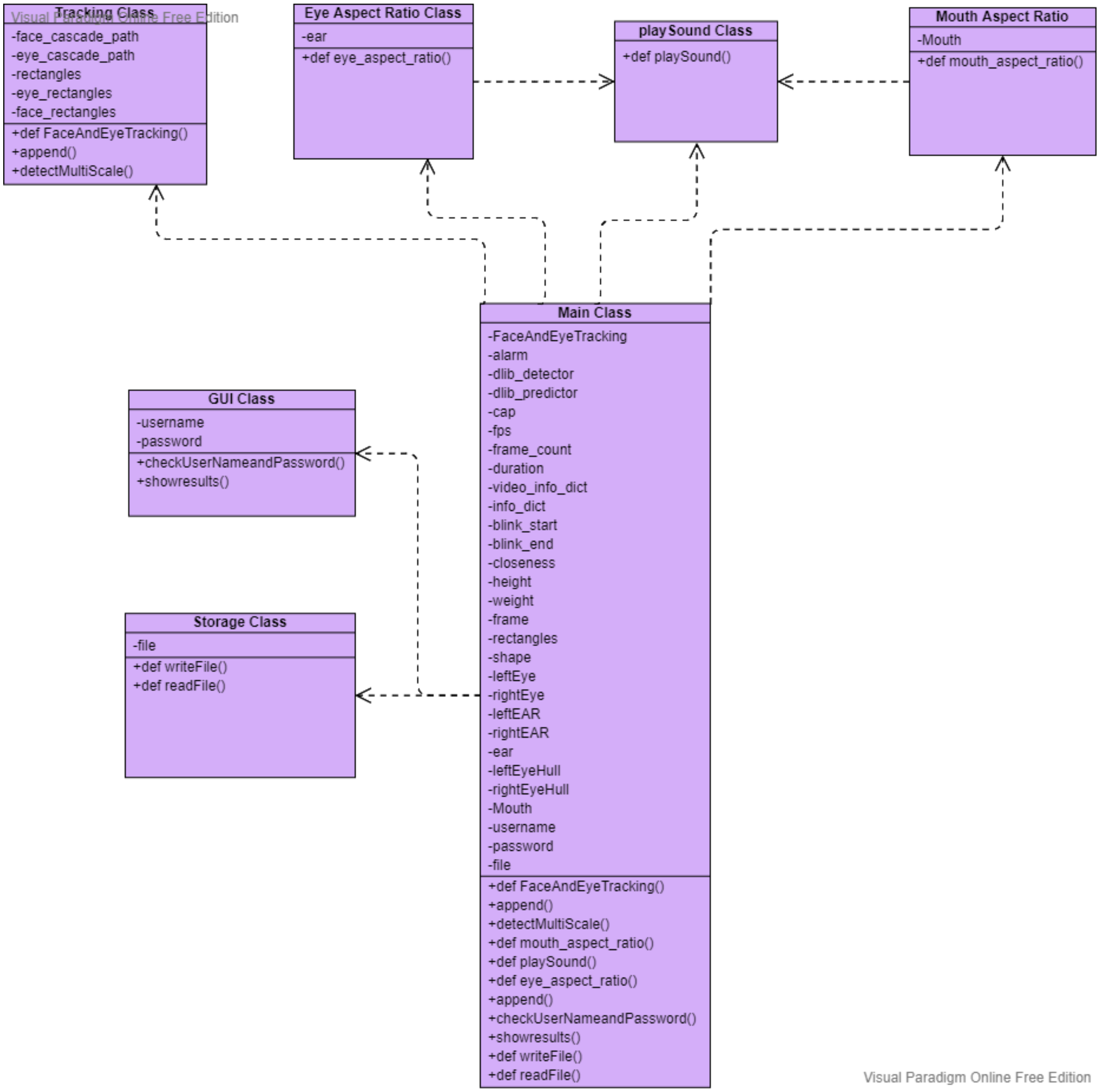

Figure 1: UML Diagram of Drive Safe-Off Project

In order to visualize object-oriented systems in our project, UML Class Diagram is used. It is a type of static structure diagram which represents the system's structure by presenting the system's classes, attributes, methods, and relationships between these objects. As can be seen in Figure 1, the UML diagram of the Drive Safe-Off project is shown. In the next reports, there may be changes such as adding or removing sections on the UML. This should be taken into account. References of reports during technical writing of system are formatted in APA format. This standard provides readers to recognize references easily.

1.4 Definitions, acronyms, and abbreviations

UML: Unified Modeling Language

EAR: eye aspect ratio

MAR: Mouth aspect ratio

FPS: Frames per second

2. Packages

OpenCV : It is an open source library used in real-time computer vision applications.

CV2: It is the name that OpenCV developers chose when they created the binding generators,

NumPy: It is a Python library used for working with arrays.

Dlib: It is a modern C++ toolkit that contains machine learning algorithms.

distance: It computes the euclidean distance between the two vertical eye landmarks (x and y coordinates)

argparse : The argparse module makes it easy to write user-friendly command-line interfaces. It parses the defined arguments that are taken from the system.

FPS: It is used to read a number of frames from the video and see how much time has elapsed to calculate frames per second.

playSound: The playsound module is a cross-platform module that is used to play audio files.

Tracking: The class that is used to track face and eye after the detection.

3. Class Interfaces

When creating a class, each class consists of descriptions, attributes, and methods. Since these classes are still under development, changes can occur. The currently assumed order is as follows:

3.1 Tracking Class

Class Name: Tracking Class

Description: This class performs the necessary operations to follow after detecting the units that need to be followed (eye, mouth) on the face.

Attributes:

- String face_cascade_path
- String eye_cascade_path
- double[] rectangles
- double[] eye_rectangles
- double[] face_rectangles

Methods:

- *double[] FaceAndEyeTracking():* This method tracks the face and the eye.
- *Void append() :* It is used to add the width, height and coordinates of the face and eye to the array.
- *void detectMultiScale() :* It is used to detect the face and eye of the driver in our dataset. Then, the detected face and eye are returned in the type of rectangles.

3.2 Eye Aspect Ratio Class

Class Name: Eye Aspect Ratio Class

Description: This class calculates the EAR value of the eye. It is a fixed value when the eye is open, but approaches 0 as the eye is closed.

Attributes:

- double[] ear

Methods:

- *int eye_aspect_ratio():* This method calculates EAR value of the eye.

3.3 playSound Class

Class Name: playSound Class

Description: This class warns you when a behavior that should not have happened is detected. These behaviors are long-term eye closure and long-term yawning.

Attributes:

Methods:

- *void playSound():* This method play sound for warning.

3.4 Storage Class

Class Name: Storage Class

Description: This class responsible for writing and reading file. These files keep data shared while tracking, username and password.

Attributes:

- File file

Methods:

- *void writeFile(File file) :* This methods writes the data to the file.
- *void readFile(String pathToFile):* This method reads the data in the file.

3.5 MouthAspectRatio Class

Class Name: MouthAspectRatio Class

Description: This class calculates the MAR value of the mouth. Mouth aspect ratio is almost zero when the mouth is closed, and the aspect ratio of the mouth increases slightly when the mouth is open.

Attributes:

- double[] Mouth

Methods:

- *double mouth_aspect_ratio() :* This method calculates MAR value of the mouth.

3.6 GUI Class

Class Name: GUI Class

Description: This class handles everything related to the GUI.

Attributes:

- String username
- String password
- File file
- Storage storage

Methods:

- *boolean checkUserNameandPassword(username,password*): This method checks if username and password match.
- *void showresults(File file*): This methods shows the results in the saved files.

3.7 Main Class

Class Name: Main Class

Description: The carry method of the system handles entire intersection of classes and method provides an singular execution from the beginning to the end of the driving session.

Attributes:

- FaceAndEyeTracking
- alarm
- fhog_object_detector dlib_detector
- shape_predictor dlib_predictor
- VideoCapture cap
- float fps
- int frame_count
- float duration
- dict video_info_dict
- dict info_dict
- int blink_start
- int blink_end
- int closeness
- int height
- int weight
- ndarray frame
- list rectangles
- full_object_detection shape
- ndarray leftEye
- ndarray rightEye
- float64 leftEAR
- float64 rightEAR
- int ear
- int Mouth
- ndarray leftEyeHull
- ndarray rightEyeHull

Methods:

- *def FaceAndEyeTracking()*
- *append()*
- *detectMultiScale()*
- *def mouth_aspect_ratio()*
- *def playSound()*
- *def eye_aspect_ratio()*
- *append()*

4. Glossary

Session: The time interval between driving starts and ends.

User: It is the person who will drive the vehicle in a certain time period.

Supervisor: The person has the administrative authority of the system environment.

Detection: It is a computer technology being used in a variety of applications that identifies human faces in digital images.

Tracking: It is the faster and more accurate determination of the location of the object with the detection made in the previous frame and the information obtained.

Anomaly Detection: It is the identification of uncommon situations that raise suspicions by differing significantly from the major part of the data.

5. Reference

Williamson, A.; Friswell, R.; Olivier, J.; Grzebieta, R. Are drivers aware of sleepiness and increasing crashrisk while driving? Accid. Anal. Prev.2014,70, 225–234.

He, J.; Choi, W.; Yang, Y.; Lu, J.; Wu, X.; Peng, K. Detection of driver drowsiness using wearable devices: A feasibility study of the proximity sensor. Appl. Ergon.2017,65, 473–480