DRIVER MONITORING SYSTEM

Andrea Bernini
Department of Computer Science
Sapienza University, Rome, Italy
bernini.2021867@studenti.uniroma1.it

Antonio Andrea Gargiulo
Department of Computer Science
Sapienza University, Rome, Italy
gargiulo.1769185@studenti.uniroma1.it

Donato Francesco Pio Stanco
Department of Computer Science
Sapienza University, Rome, Italy
stanco.2027523@studenti.uniroma1.it

PROJECT FOR
MULTIMODAL INTERACTION
COURSE

Academic Year 2022/2023

CONTENTS

1 Introduction

In the modern era, road safety has become a significant concern of society. Human factors, such as **driver distraction**, **drowsiness** or **lack** of attention behind the steering wheel cause many road accidents. More specifically, our goal is to detect such situations through the integration of different input modes, such as **gaze tracking**, **eyelid tracking**, and **driver activity** monitoring.

To achieve this goal, we have adopted **Deep Learning** approaches and **Computer Vision** techniques for processing and analyzing the data from the cameras inside the vehicle. The driver monitoring system developed in our project can detect the driver's gaze, i.e. the direction and point of gaze fixation, to evaluate the driver's attention and concentration on the road. In addition, the system can identify signs of drowsiness, such as prolonged closing of the eyes.

In addition, the system can also detect driver distractions by analyzing their **activities** and positions while driving. This helps identify times when the driver might be distracted by external factors, such as mobile phone use or conversations with passengers. If distraction or sleepiness is detected, the system can provide warnings or take action to reduce the risk of an accident.

The project code is available at the following link: multimodal-interaction-project.

2 System Goals

Our idea for the project was to build a driving monitoring system. This system takes care of monitoring various aspects including driver drowsiness, driving distractions, and the audio volume inside the passenger compartment to avoid dangerous distractions while driving.

The developed system achieves the following **objectives**:

- the user automatically activates the video and audio recording mode as soon as he starts the vehicle;
- the user will be notified via an alert if he is distracted, sleepy while driving, or if the noise inside the passenger compartment is too loud;
- for testing purposes, a GUI or CLI can be used to select a video-only, audio-only, or both recording.

3 Requirements Analysis

This section describes the system requirements, which are identified as follows:

$$\langle\text{type}\rangle \text{ - } \langle\text{number}\rangle$$

where **type** indicates the subdivision of the requirements into:

- **Functional Requirements**: indicate the functionality of the program and are divided into user functional requirements (**UF**) and system functional requirements (**SF**)
- **Non-Functional Requirements**: Indicated with the **NF** type, they represent constraints on the services offered.

3.1 Functional Requirements

In this section, we describe the functional requirements that the system satisfies:

ID	**Name**
UF-01	Driver user
UF-02	Deactivation of monitoring
SF-01	Automatic Video and Audio Recording
SF-02	Continuous Video and Audio Recording
SF-03	Distraction Detection
SF-04	Drowsiness Detection
SF-05	Gaze Detection
SF-06	Loudness Detection
SF-07	Send Alerts

Now let's look at every single functional requirement of the system in more detail:

ID	**UF-01**
Name	Driver User
Type	Functional Requirement of the User
Priority	Obligatory
Description	The user must be able to drive according to the Italian highway code

ID	**UF-02**
Name	Deactivation of monitoring
Type	Functional Requirement of the User
Priority	Obligatory
Description	The user must be able to deactivate the system manually

ID	**SF-01**
Name	Automatic Video and Audio Recording
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system needs to record video and audio automatically from the very beginning of vehicle startup

ID	**SF-02**
Name	Continuous Video and Audio Recording
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system shall record video and audio continuously during vehicle operation.

ID	**SF-03**
Name	Distraction Detection
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system needs to analyze the video to detect user distraction

ID	**SF-04**
Name	Drowsiness Detection
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system needs to analyze the video to detect if the user is drowsy

ID	**SF-05**
Name	Gaze Detection
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system needs to analyze the video to detect looking away from the road

ID	**SF-06**
Name	Loudness Detection
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system needs to analyze the audio to detect internal vehicle noise

ID	**SF-07**
Name	Send Alerts
Type	Functional Requirement of the System
Priority	Obligatory
Description	The system needs to send audio cues to the user if it detects distraction, drowsiness or looking away from the road

3.2 Non-Functional Requirements

In this section, we describe the non-functional requirements of the system:

ID	**Name**
NF-01	Portability
NF-02	Variation of brightness
NF-03	Efficiency
NF-04	Reliability
NF-05	Hardware accessibility

Now let's look at every single non-functional requirement of the system in more detail:

ID	**NF-01**
Name	Portability
Type	Non-Functional Requirement
Priority	Obligatory
Description	The system must be able to adapt to different types of vehicles (cars, trucks, etc.).

ID	**NF-02**
Name	Variation of brightness
Type	Non-Functional Requirement
Priority	Obligatory
Description	The system must be able to operate reliably in variable ambient light conditions

ID	**NF-03**
Name	Reliability
Type	Non-Functional Requirement
Priority	Recommended
Description	The results produced by the system must be reliable to avoid annoying the user

ID	**NF-04**
Name	Evolvability
Type	Non-Functional Requirement
Priority	Recommended
Description	The system must be composed of a modular architecture.

ID	**NF-05**
Name	Hardware accessibility
Type	Non-Functional Requirement
Priority	Obligatory
Description	The system must be able to access the camera and the microphone.

4 Architecture

The system integrates different input and output modes (Figure 1). In particular, **cameras** are used to monitor the driver's behavior, this is analyzed in *real-time* to detect and identify the driver's drowsiness or distraction. In addition, the vehicle's **microphones** are also used to analyze the volume inside the passenger car's compartment.

To avoid increasing driver distraction, the multimodal approach aims to provide useful warning, advice and feedback on the most appropriate output channels based on the driver's "**possibilities**" and **preferences**.

4.1 System Architecture Implementation

The class diagram of our system is shown in Figure 2, where one of the main components is the `Detector` class, which assumes the role of **controller**: it manages the results of the underlying classes (`Distraction`, `Drowsiness`, `LookingAway`, and `Loudness`), each one having a different driver monitoring task.

The computation of the three main types of detection is carried out **sequentially**, as well as the propagation of the alarm, in fact, each class of detection, returns an *alarm status*, and these are controlled by the `Detector` sequentially and exclusively for drowsiness and gaze, i.e. the first is the state of the `Distraction`, then the `Drowsiness` and `LookingAway` are controlled exclusively. Then if there is an alarm, it is propagated for the positive state/s to the controller.

Instead, to manage the *simultaneous recording of video and audio*, we used the `OpenCV` and `PyAudio` libraries, following the work done by Flores, defining a main `Recorder` class, which manages the execution of the `VideoRecorder` and `AudioRecorder` classes, starting the recordings through the use of *Threads*.

Finally, for emulative and testing purposes, we have also decided to develop a graphical interface which we will discuss in Section 4.2.

4.1.1 Deployment Diagram

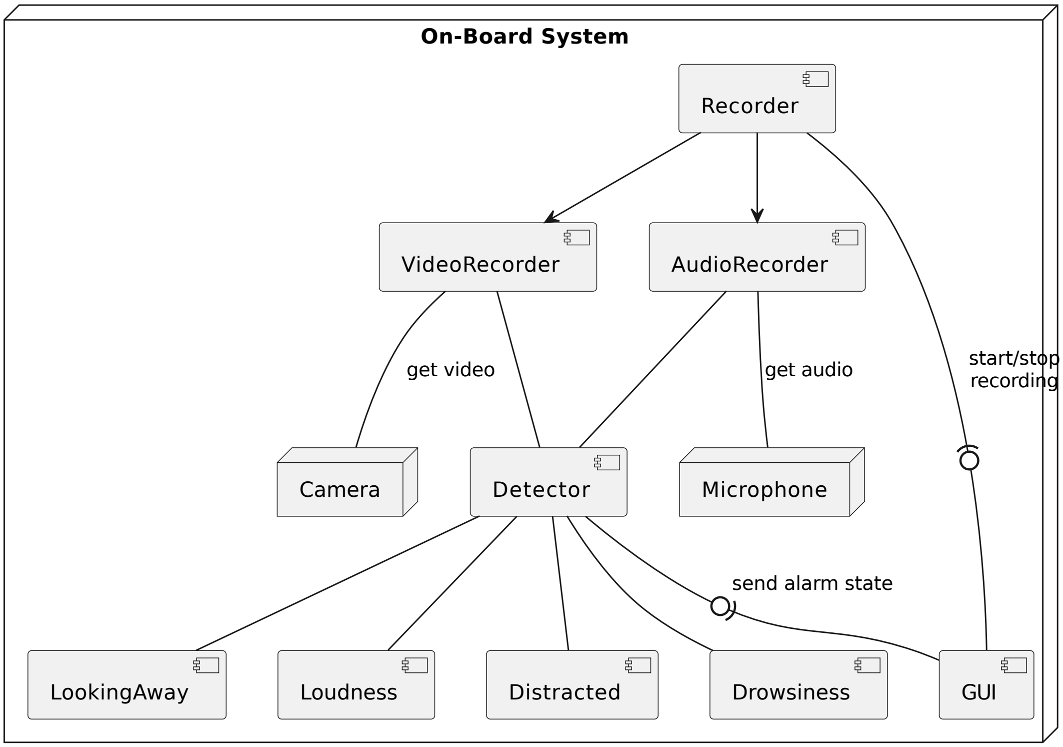


Figure 1: Deployment Diagram.

4.1.2 Class Diagram

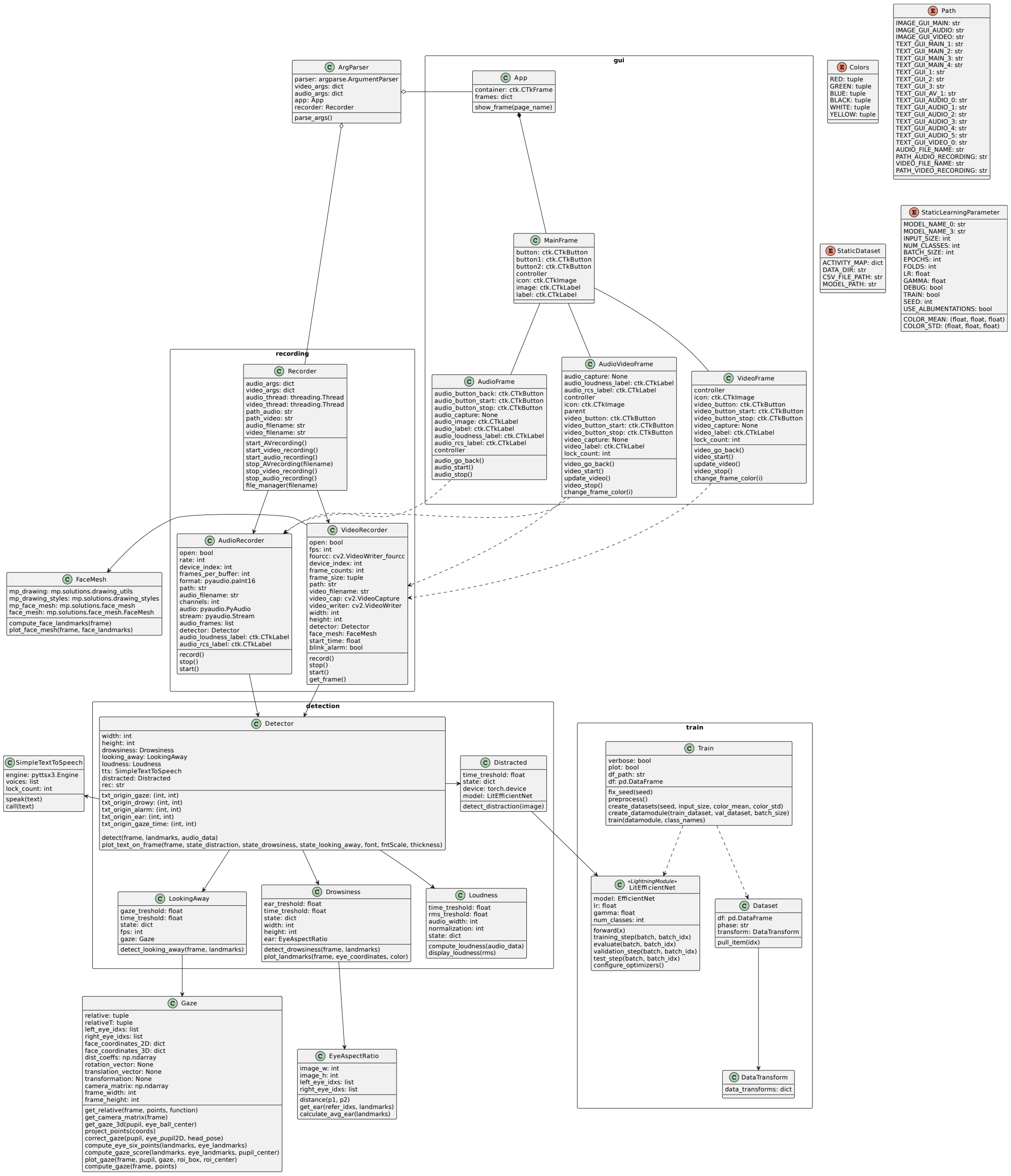
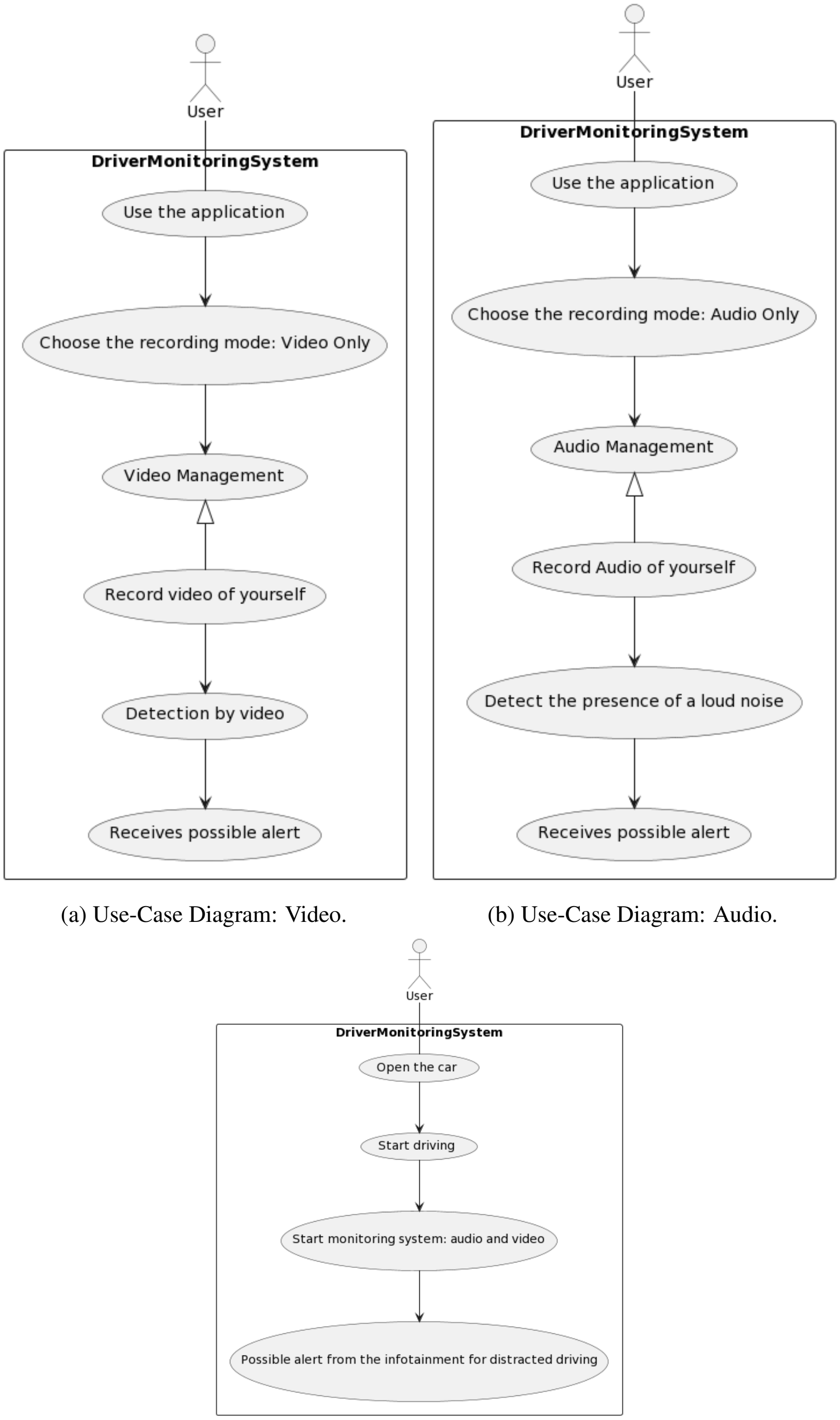


Figure 2: Class Diagram.

4.1.3 Use Cases Diagrams



(a) Use-Case Diagram: Video.

(b) Use-Case Diagram: Audio.

(c) Use-Case Diagram: General Case.

Figure 3: Use-Case Diagrams

4.1.4 Activity Diagram

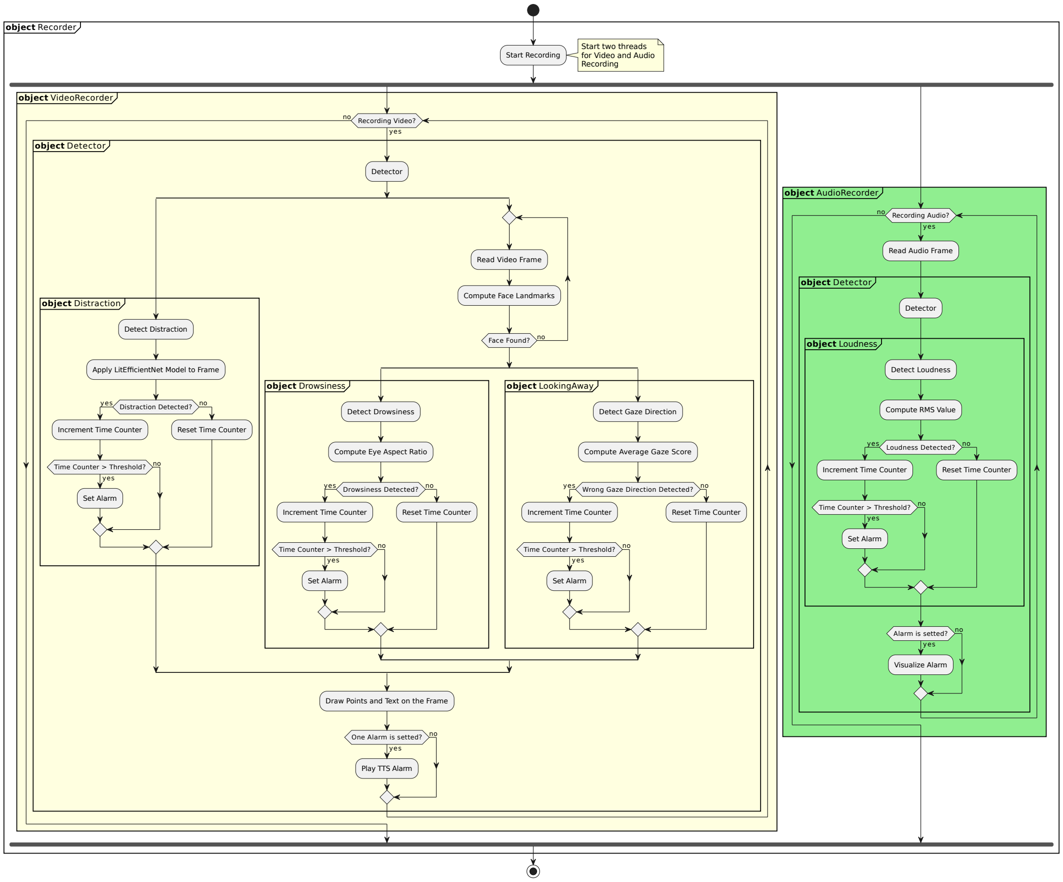


Figure 4: Activity Diagram.

4.1.5 Sequence Diagram

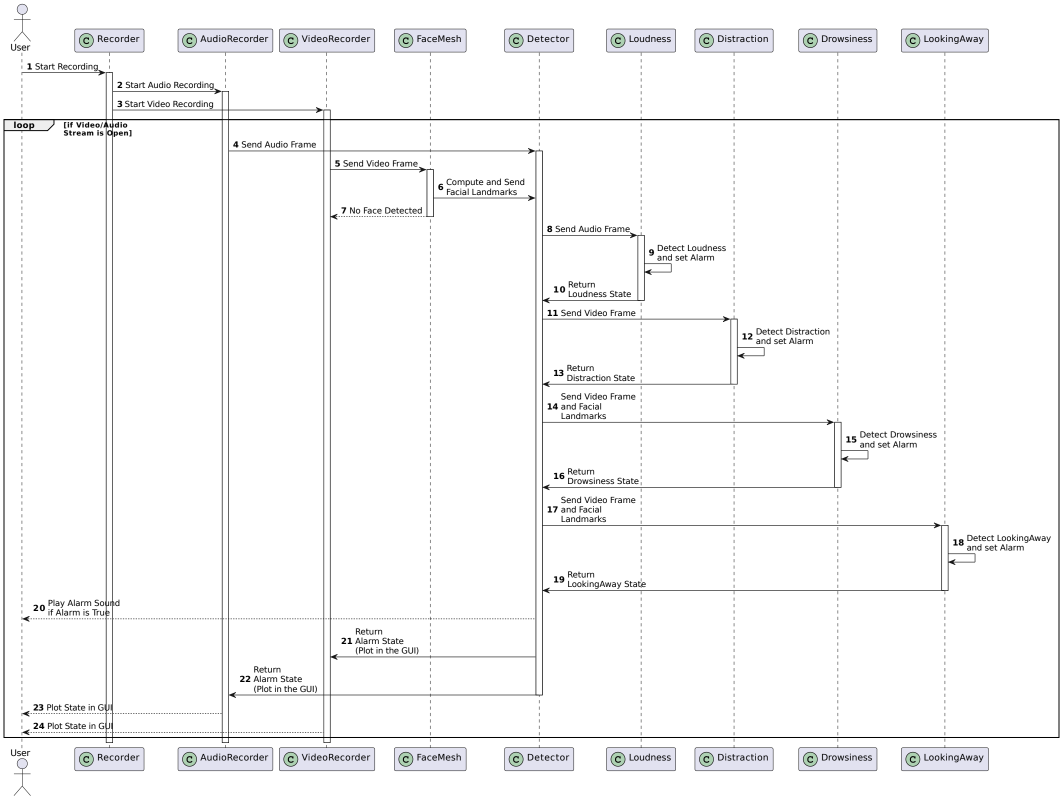
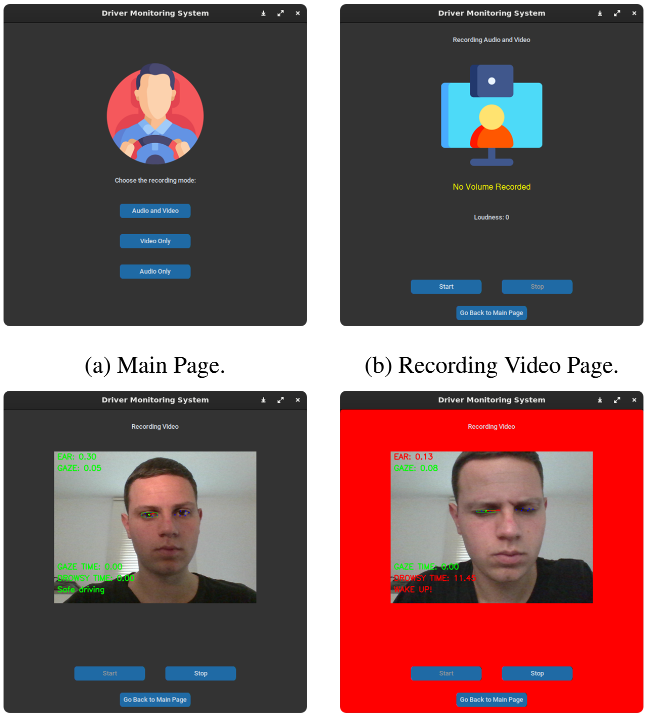

User
Recorder
AudioRecorder
VideoRecorder
FaceMesh
Detector
Loudness
Distraction
Drowsiness
LookingAway
1 Start Recording
2 Start Audio Recording
3 Start Video Recording
loop
[if Video/Audio Stream is Open]
4 Send Audio Frame
5 Send Video Frame
6 Compute and Send Facial Landmarks
7 No Face Detected
8 Send Audio Frame
9 Detect Loudness and set Alarm
10 Return Loudness State
11 Send Video Frame
12 Detect Distraction and set Alarm
13 Return Distraction State
14 Send Video Frame and Facial Landmarks
15 Detect Drowsiness and set Alarm
16 Return Drowsiness State
17 Send Video Frame and Facial Landmarks
18 Detect LookingAway and set Alarm
19 Return LookingAway State
20 Play Alarm Sound if Alarm is True
21 Return Alarm State (Plot in the GUI)
22 Return Alarm State (Plot in the GUI)
23 Plot State in GUI
24 Plot State in GUI
User
Recorder
AudioRecorder
VideoRecorder
FaceMesh
Detector
Loudness
Distraction
Drowsiness
LookingAway

Figure 5: Sequence Diagram.

4.2 Graphic Interface

To facilitate the tests and the project's presentation, we have developed a *simple graphical interface* that allows to start an audio and/or video recording (Figure 6a), display the detection results of our system, showing the frames captured by the laptop's webcam, and the processed drawing above them: the points of the eyes, used for detecting drowsiness, and the line representing the direction of the gaze (Figure 6c).

This **GUI** is not foreseen in the real use of the system, as the latter must be integrated directly into the *vehicle's infotainment*, started and stopped **automatically** with the vehicle's movement. However, we have considered the graphical interface as the car's display, showing the visual alarms through it (red flashing of the interface) (Figure 6d).



(a) Main Page.

(b) Recording Video Page.

(c) Recording Webcam.

(d) Drowsiness Alert

Figure 6: GUI for testing.

4.2.1 Software Libraries

To develop the driver monitoring system, we used the Python programming language, due to its flexibility and the large number of libraries, which are essential for achieving the set objectives.

The main libraries used are:

- **Mediapipe**, an open-source library developed by Google for media processing. It was used in the project to perform real-time analysis and recognition of facial landmarks within the video, generating a face mesh from which individual points can be extracted.

- **OpenCV** (Open Source Computer Vision Library), a popular open-source library for computer vision and image processing. It was used in the project to capture, process and analyze video frames.
- **Pyaudio**, a Python library that provides an interface for real-time audio management. It was used in the project for audio recording and playback.

5 Multimodal interaction

The channels used in our implementation are mainly **two**, namely **Video** and **Audio**, however, our initial idea also envisaged the use of the **tactile sensory channel**, used for example to transmit alarm signals through the vibration of the steering wheel or seat, which however we were not able to implement it due to lack of necessary hardware.

5.1 Video

The video channel is used through:

- input, for **recording** through the vehicle's camera, positioned on the steering axis behind the steering wheel, which records the driver's face. This video stream is analyzed in real-time by the `Detector`, which takes care of starting the three detection phases: Distraction, Drowsiness, and Gaze direction. It controls the status of the three respective alarms and, if necessary, reproduces them via video (explained in the next point) or via audio (Section 5.2).
- output, for **alarm transmission**, which is done via the vehicle's infotainment screen, assuming it is equipped with one, and in our tests we considered the GUI as infotainment as explained in Section 4.2.

5.1.1 Drowsiness

`Drowsiness` is the class that implements the management of the sleepiness detection alarm, implemented in the `EyeAspectRatio` class, taking inspiration from the work done by Singh (2022), where for each frame, through the **Mediapipe** framework, the driver's face mesh, as we can see from Figure 7, is calculated using the `FaceMesh` class and 6 specific points are extracted for each eye, as shown in Figure 8, for the calculation of the **EAR (Eye Aspect Ratio)** formula:

$$EAR = \frac{||P_2 - P_6|| + ||P_3 - P_5||}{2||P_1 - P_4||} \tag{1}$$

The EAR value is a **scalar** representing the level of eye-opening, where 0 equals a completely closed eye.

If in a frame it is detected that the eye is closed: below a certain **EAR** value. A variable that keeps track of the time spent with the eyes closed is increased; if this variable **exceeds** a certain threshold, the **state** of the EAR alarm is returned to the `Detector` class that manage the warning or the action to adopt.

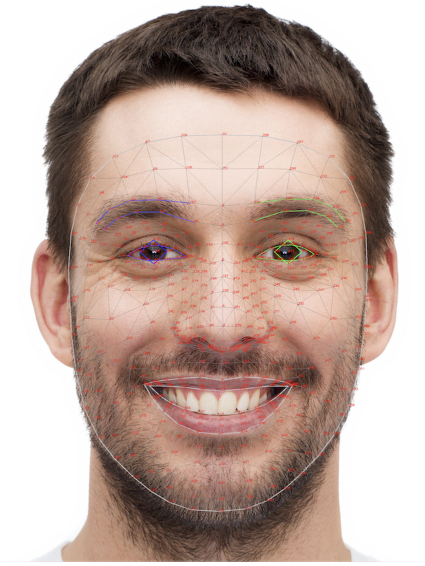


Figure 7: Face Mesh from Mediapipe (Med (2023))

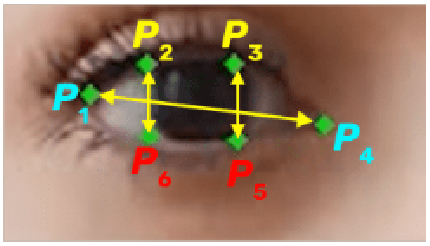


Figure 8: Points for Eye Aspect Ratio (Singh)

5.1.2 Gaze

The `LookingAway` class has a similar role to `Drowsiness`. It manages the propagation of the alarm through a variable that considers how much time a user spends not looking at the road (in front of him).

The `Gaze` class deals with the calculation of the gaze direction, returning the average value of the gaze between the left and right eye, calculated through the **Euclidean distance** between the **center** of the eye and the **pupil** position.

The eye's **center** is calculated as the center of the minimum rectangle (bounding box), which encloses the 6 points used previously in the calculation of the EAR, called **ROI** (Region of Interest). In contrast, the position of the pupil is calculated by extracting from the Mediapipe's **mesh** the specific face landmarks (Figure 9).

Regarding the gaze plot in the testing GUI, we used **OpenCV** to calculate the transformation from the 3D world to the 2D one and then drew a line starting from the pupil and pointing in the direction of the gaze, taking inspiration from the work done by Aflalo (2022). More precisely we have defined a **3D coordinate system**, which has the tip of the nose as its origin, and concerning this other 5 points are defined which correspond to the chin, the corner of the left eye, and the right eye, and the corners left and right of the mouth. We then used the Pinhole camera model (from Fgnievinski) to know the

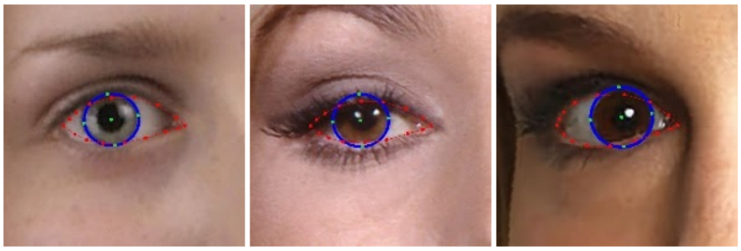


Figure 9: Iris Landmarks Detection with Mediapipe (from the work of Fernandez).

change of the 2D points found by Mediapipe, to the 3D ones, obtaining the translation and rotation vectors, then used to calculate the **projection** of the 3D point to the 2D point of the gaze, a final result can be seen in Figure 10.

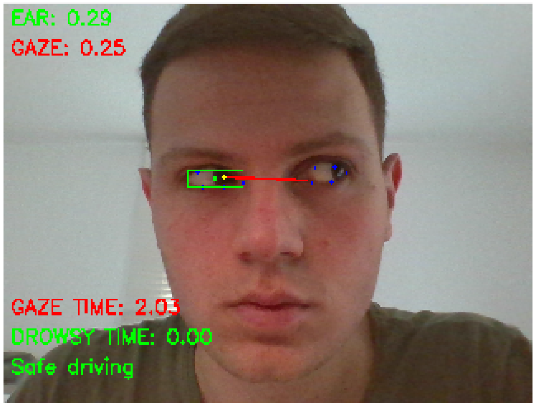


Figure 10: Gaze Plot from our system.

Since the gaze is obtained, as said before, through the Euclidean distance on the 2D coordinate space, the positioning of the camera requires to be necessarily in front of the user, so in a real case scenario this is almost impossible to obtain without occluding the driver's vision and then the system must be adjusted to take in account the different orientation of the camera inside the car dashboard.

5.1.3 Distraction

The detection of user distraction is managed by the `Distracted` class, which uses the Deep Learning model `LitEfficientNet`, based on the **EfficientNet** architecture and pre-trained on **ImageNet-1k** and then finetuned on the *State Farm Distracted Driver Detection* (Montoya et al.), to analyze the video frames checking for distracted behaviors of the driver. If the same type of distraction is detected for a certain period, then the alarm is **propagated** to the `Detector` class.

The mentioned dataset contains images of drivers, each taken in a car with a driver performing some activities inside it (Figure 11) (texting, eating, talking on the phone, applying makeup, stretching behind, etc.). In addition to the images, the dataset provides a CSV containing 3 features, i.e. the driver's ID, the name of the image, and the associated class, for a total of about 22,000 rows and 26 different drivers.

There are a total of 10 classes to be predicted, namely:

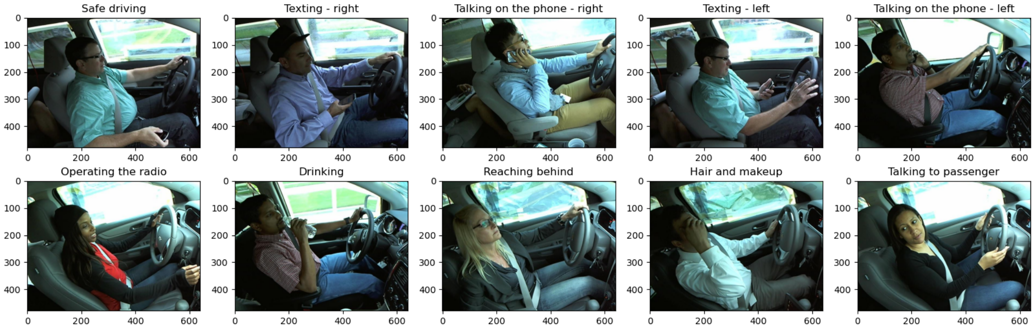


Figure 11: Dataset Classes.

- **c0**: Safe Driving,
- **c1**: Texting - right,
- **c2**: Talking on the phone - right,
- **c3**: Texting - left,
- **c4**: Talking on the telephone - left,
- **c5**: Operating the radio,
- **c6**: Drinking,
- **c7**: Reaching behind,
- **c8**: Hair and makeup,
- **c9**: Talking to the passenger.

The amount of images for each class is very balanced as can be seen in Figure 12. While the same cannot be said of the distribution of the number of images for each driver (Figure 13), we can see that among the user with the most pictures and the one with fewer there is a difference of about 800 units.

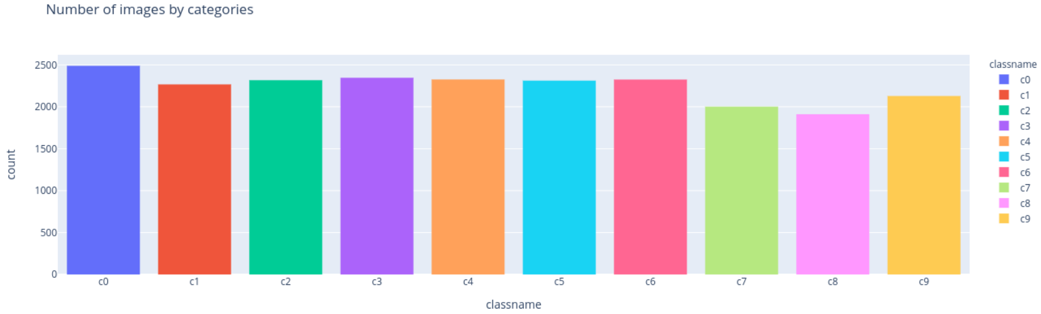


Figure 12: Number of images for each class.

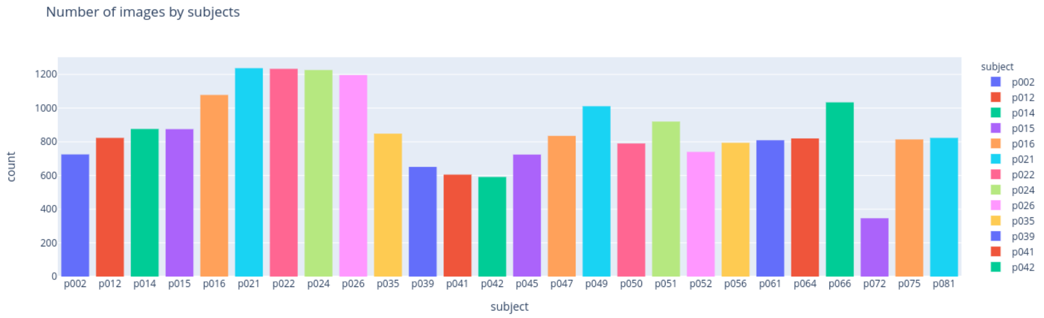


Figure 13: Number of images for each driver.

We finetuned a pre-trained EfficientNet-b0 model using the domain-specific dataset augmented with transformations such as rotation and brightness variation, resulting in significant improvements in the final results. The model, as we can see from Figure 14, achieved through the training a validation **F1 score** of 99.7% and a validation **accuracy** of 99.8%. These resulting metrics demonstrate the model's ability to accurately classify the possible driver's distractions **within our domain** and from the specific viewpoint chosen. The high performance in the F1 score indicates the model's robustness in achieving a balance between precision and recall, making it also reliable for tests in real-world scenarios.

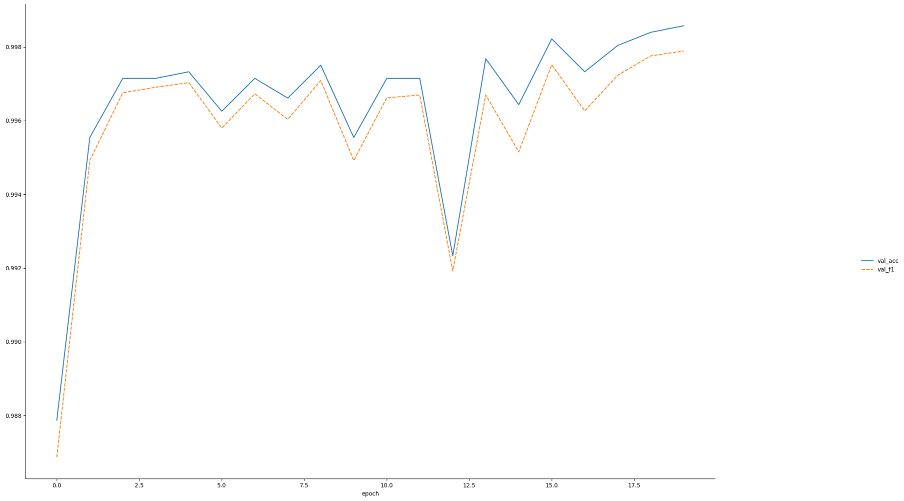


Figure 14: Diagram of F1 score and Accuracy on the validation set by epoch

5.2 Audio

The audio channel is used:

- in input, for recording with the vehicle's microphones, whose data is analyzed and passed in real-time to the `Detector`, which starts detecting the noise level inside the vehicle, instantiating the `LoookingAway` class.

- in output, for *alarm transmission*, which is done via the vehicle's speakers, using a **TTS** system that allows speech to be generated starting from strings of text, which notifies the driver of his sleepiness or distraction.

5.2.1 Loudness

`Loudness`, is the class that implements the management of the alarm relating to the detection of high noise, where for each chunk of an audio frame, through the `audioop` framework, the **RMS** (Root Mean Square) value is calculated, a measure of signal strength audio:

$$RMS = \sqrt{\frac{\sum_{i \in N}(S_i^2)}{N}}. \tag{2}$$

where S represents the audio frame and N its length.

Once the signal strength has been calculated, we can obtain the value in **decibel (dB)**:

$$dB = 20 * log_{10}(RMS/20) + 130 \tag{3}$$

However this methodology brings some problems because we are evaluating a recorded sound, we are measuring it only in the specific position where the microphone is located, polarized with respect to the direction in which the microphone is pointing, and filtered by the frequency response of your hardware.

Furthermore, without being able to calibrate the hardware, the decibel value cannot be calculated exactly, in fact we are only calculating the peaks in the audio file and are compared with other peaks in the same audio file, also called **dB FS** (Full Scale).

In fact, to obtain reliable results, a specialized microphone must be used for calculating the **dB SPL** (Sound Pressure Level).

We decided to develop this monitor because article 350 of the Italian highway code shows that music in cars cannot exceed **60 LAeq dB** measured 10 cm from the driver's ear with the microphone facing the source and with the vehicle at the door and windows closed. This reference value expresses, on the one hand, the equivalent continuous level of sound pressure (LAeq) and the volume in decibels (db).

As with the other detection systems, the `Loudness` class also propagates the alarm only if the dB threshold is exceeded for a given period.

5.2.2 TTS

To warn the user and spread the alarm we decided to use `eSpeak` (esp (2023)), an open-source TTS (Text To Speach) system, for Linux and Windows platforms, which uses a **formant synthesis method**, providing many languages in one file relatively small in size. Instead, for MacOS, we used `say`, which functioning is the same as `eSpeak`. Formant synthesis is an artificial speech generation method that does not use human speech samples during performance, in fact, the signal is created from a simple human speech generation model that includes a set of filters and an initial sound source. We decided to opt for this solution because it is the only one that allows real-time speech generation in `Python`, without the need to first generate an audio file, save it, and then play it when necessary.

To avoid annoying and repetitive alarm messages the system is equipped with a blocking semaphore, whose main function is to discard duplicated warnings, overlapping alarms, and continuous feedback. So, the driver can receive only relevant and useful information without the over usage of the audio channel.

5.3 TACTILE

A tactile feedback implementation within a driver monitoring system holds great potential, particularly in addressing the needs of **deaf individuals**. Incorporating tactile cues into the system, such as vibrations or gentle pulses, can provide essential information and alerts to drivers who rely heavily on visual or tactile feedback. For instance, when the system detects drowsiness or inattentiveness, it can trigger specific types of vibrations on the steering wheel or the seat to draw the driver's attention. This tactile feedback can effectively bridge the communication gap for deaf drivers who may not rely on auditory cues. Moreover, the tactile feedback system can enhance the overall driving experience for all users by providing an additional layer of sensory information, promoting safer and more attentive driving practices. We have not explored this type of implementation since it requires specialized and expensive hardware, but we are confident that the integration of this type of feedback inside our application can be easily done thanks to the modularity of the architecture components.

6 FUTURE WORKS

One of the potential improvements involves using a different dataset than the one currently in use. This new dataset should contain images of drivers taken **head-on** rather than side-on. The goal of this change is to improve the **accuracy** of the detection since the frontal images provide a more direct view of the driver's face. It may be possible to detect signs of distraction, drowsiness, or inattentive gaze with greater accuracy.

Another possible improvement consists in the implementation of a **multi-process** system for the different types of detection: distraction, drowsiness, and inattentive gaze. Currently, the system performs these detections sequentially, but a multi-process architecture would allow these operations to be performed in parallel. This could lead to increasing the overall system performance, allowing for faster and more accurate detection of different driver monitoring conditions.

A further improvement could be to use an **infrared camera** so that it can work even in the dark since the system has currently been tested and implemented using simple RGB webcams. To improve eye recognition, and above all the vertical movement of the pupil, the ideal would be to have a large number of infrared emitters and receivers dedicated exclusively to registering the driver's eye.

Another hardware-based improvement could be to use a **specialized microphone** for measuring SPL decibels, since, as explained in Section 5.2.1, the ones we calculated are the Full-Scale decibels, which have no correlation with the former, and using dedicated hardware it is not possible to measure them.

To conclude, other types of improvements that go over the environment of this system, but can be useful in the view of a **comprehensive multimodal assistant** system can be a voice-controlled interface for the control of the vehicle's infotainment system.

All of these improvements offer interesting perspectives for the future development of the driver monitoring system, allowing further optimization of the detection functions and a general improvement of the overall system performance.

7 Conclusion

In conclusion, the development of a multimodal driver monitoring system that incorporates audio and video modalities of interaction represents a significant advancement in ensuring driver safety and enhancing overall driving experiences. By leveraging both audio and video inputs, this system offers a comprehensive approach to monitoring driver behavior, detecting potential distractions, and providing timely feedback or alerts. The combination of modalities enables a more nuanced understanding of driver actions and facilitates effective communication between the system and the driver without cognitive overload. This project holds great potential for reducing the risk of accidents, improving road safety, and ultimately creating a more secure and comfortable driving environment for all road users. As technology continues to evolve, multimodal interaction systems like the one in this project will play a crucial role in shaping the future of transportation and contributing to a safer driving experience.

References

Face landmark detection guide, 2023. URL https://developers.google.com/mediapipe/solutions/vision/face_landmarker.

eSpeak: speech synthesizer, 2023. URL https://espeak.sourceforge.net/.

Amit Aflalo. *Eye gaze estimation using a webcam.*, Jan 2022. URL https://medium.com/mlearning-ai/eye-gaze-estimation-using-a-webcam-in-100-lines-of-code-570d4683fe23.

Joe Fernandez. *Iris Detection*, 2023. URL https://github.com/google/mediapipe/blob/master/docs/solutions/iris.md.

Fgnievinski. *Pinhole camera model*, Sep 2022. URL https://en.wikipedia.org/wiki/Pinhole_camera_model.

J. Rodrigo Flores. *Audio and video recording using Python*, May 2016. URL https://github.com/JRodrigoF/AVrecordeR.

Anna Montoya, Dan Holman, Taylor Smith, and Wendy Kan. *State Farm Distracted Driver Detection*, 2016. URL https://kaggle.com/competitions/state-farm-distracted-driver-detection.

Vaibhav Singh. *Driver drowsiness detection using mediapipe*, Sep 2022. URL https://learnopencv.com/driver-drowsiness-detection-using-mediapipe-in-python.