Enhancing Autonomous Vehicle Safety through Real-time User Health Monitoring and Various factors Integration

Deekshith Reddy Yeruva, Lavanya Reddy Duddhugunta, Manohar Veeravalli, Sai Charan Raghupatruni, Shubham Sudhakar Indulkar, Sai Likhith Yadav Seera
{*dyeruva, lduddhug, mveerava, sraghup7, sindulka, sseera*}*@asu.edu*
Arizona State University
Tempe, AZ

***Abstract*—This project endeavors to create a safety-driven framework tailored for autonomous vehicles, concentrating on the real-time monitoring of vital passenger health metrics. With a specific focus on intoxication levels, respiratory rate, heart rate, BMI, and sleepiness, the project aims to develop an adaptive advisory control system. The primary challenge lies in crafting a responsive system capable of dynamically adjusting to these varied metrics. By integrating these health parameters into the autonomous vehicle's control mechanisms, the goal is to guarantee a secure and comfortable travel experience, thereby enhancing safety in autonomous transportation.**

I. Introducton

The evolution of autonomous vehicle technology promises enhanced safety, efficiency, and convenience in transportation. Ensuring the safety and well-being of passengers within these vehicles is a paramount concern, and this project aims to create a comprehensive safety-centric solution for autonomous transportation.

The project's core objective is to develop a robust advisory control system that actively monitors and responds to various vital health parameters of passengers. By integrating user-generated inputs such as heart rate, respiratory rate, height, weight, sleep duration and visual information obtained through selfies, the system endeavors to compute essential health metrics like reaction time, Body Mass Index (BMI), sleep quality and intoxication levels. Moreover, the incorporation of external environmental factors such as vehicle speed and proximity to obstacles complements the system's capability to make informed decisions.

II. Architecture

Figure 1 shows architecture of our project. Our project makes use of android studio, machine learning and MATLAB to fully implement the advisory control. The project has 3 level.

1) The mobile app is responsible for taking the user inputs like sleep duration, height, weight, video of finger and the selfie video and send to the Firebase
2) The Convolutional neural networks determine the intoxication level of the user from the selfie video.
3) Advisory control take in these processed values from above and additionally it takes in velocity and distance values to decide switching

If there is a switching, Firebase switch value is updated and mobile application notifies the user.

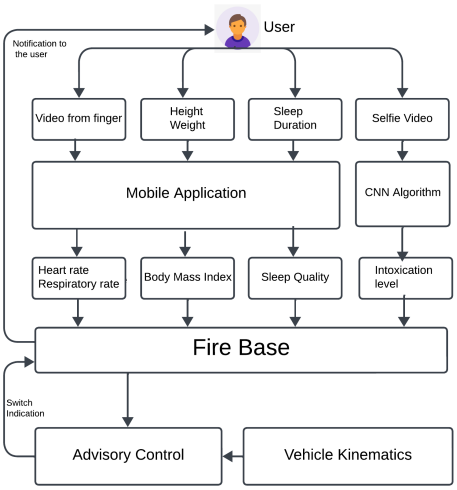

Fig. 1: Architecture of the project

III. The Application Suite

The design flow for Task 1 in our project involves calculating BMI and submitting it, along with sleep rating, and video capture for checking level of intoxicty to Google Firebase through our dedicated app. .

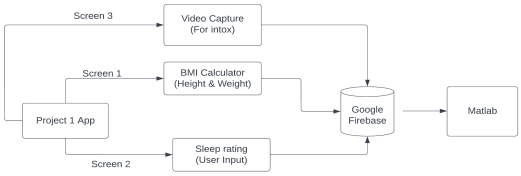

Fig. 2: Design Flow of the Task 1

The flowchart illustrates the sequence for identifying a user's state and level of intoxication using video data captured from the mobile application.

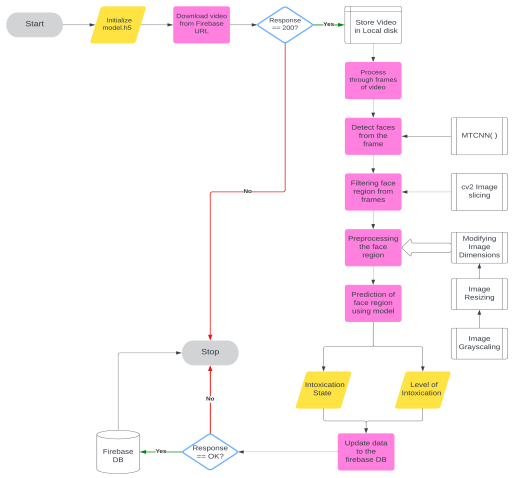

Fig. 3: Process flow of application for intoxication detection .

The process involves downloading a recorded video from Firebase, utilizing MTCNN for face detection, and predicting intoxication levels. It calculates probabilities for each detected face, determines the most frequent state, and averages probabilities. The final step includes updating a Firebase database with the inferred intoxication state and score. The process, comprising frame skipping, face filtering, and model predictions, evaluates intoxication levels in the provided video, storing results for analysis or tracking.

IV. Implementation

The project comprises three tasks: Task 1 creates a user-friendly interface for BMI, sleep rating, and facial video capture, seamlessly transferring data to Firebase. Task 2 utilizes machine learning to assess intoxication levels from facial patterns, relaying results, along with BMI and sleep data, to Firebase. Task 3 employs MATLAB and fuzzy logic to determine the switch between autonomous and manual driving based on the incoming data.

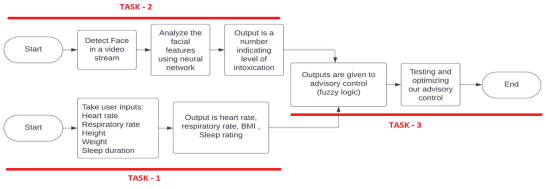

Fig. 4: Design Flow of the Project

A. Generating BMI and Sleep Rating

This task aims to enhance the user interface (UI) within the app, allowing users to input data crucial for determining their sleep quality and body mass index (BMI).

1) UI for BMI Calculation: The focus is on designing an intuitive layout that facilitates user input for height and weight. This UI element includes fields for users to enter their height and weight, and upon input submission, it triggers a calculation for the user's BMI. The resulting BMI value and weight category classification are displayed to the user within the app interface.

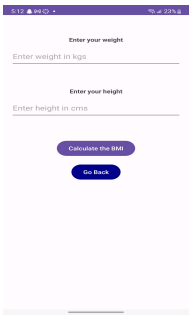

Fig. 5: UI for BMI screen

2) UI for Sleep Rating: In this section, the goal is to create a user-friendly interface that allows users to input the duration of their sleep. A logical framework, developed using Kotlin, processes this information to derive a rating representing the user's sleep quality. This derived sleep quality rating is then uploaded to Google Firebase using HTTP requests, ensuring the secure storage of this critical data.

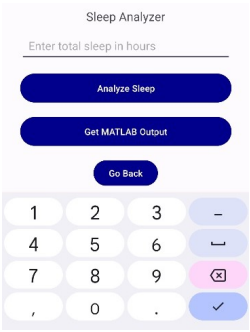

Fig. 6: UI for Sleep Rating screen

3) Camera Feature for Facial Video Recording: This task involves implementing a feature that allows users to record videos via the app. The video capture module ensures secure handling by dynamically managing permissions for each recording. These recorded videos are seamlessly uploaded and stored in Firebase storage. They serve as valuable visual data for Task 2, which involves machine learning algorithms analyzing facial patterns within the videos to assess intoxication levels.

4) Uploading BMI, Sleep Rating, and Videos to Firebase: Utilizing Google Firebase, this section ensures secure storage of BMI, sleep ratings, and recorded videos, streamlining data processing. The cloud architecture facilitates seamless data transfer to MATLAB for in-depth analysis.

B. Determining Intoxication Level

Task-2 involves two core phases: dataset creation and the development of a convolutional neural network (CNN) model.

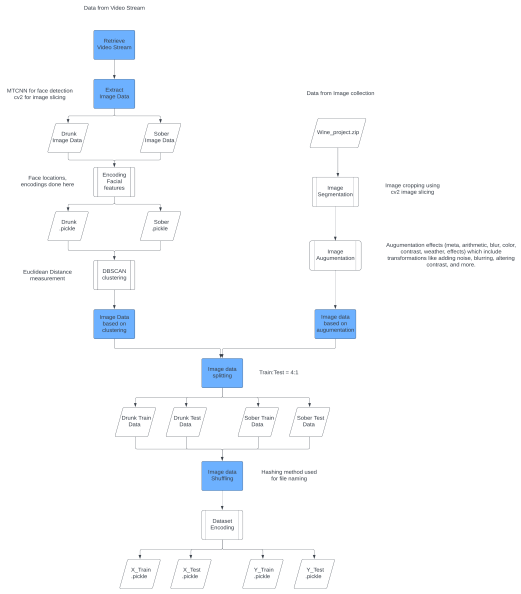

Fig. 7: Methodology for dataset creation from an online source.

The primary focus lies in meticulous data acquisition and preprocessing for intoxication level identification. This begins by downloading YouTube videos featuring sober and drunk scenarios, utilizing PyTube. Data processing employs OpenCV and MTCNN for face detection, followed by encoding facial features with the `face_recognition` library. Clustering using DBSCAN groups encoded features by scenarios, enhancing precision. Image segmentation and augmentation techniques enrich the dataset, enabling detailed facial analysis. The dataset is partitioned into distinct test and train sets for model training.

The CNN model, developed using TensorFlow, initiates with Conv2D layers employing various filter sizes for pattern extraction. MaxPool2D layers reduce complexity and overfitting. Dropout regularization improves model generalization. The Flatten Layer prepares data for densely connected layers, followed by specific Dense layers tailored for classification. Output layers use softmax for classification and linear functions for regression. The model is compiled using Binary Crossentropy and Adam optimizer, trained over 20 epochs to refine predictions.

C. Design of Advisory Control

Taking the BMI, Sleep duration and average speed metrics from the Firebase, and deciding whether the obstacle is static or dynamic, the advisory control function is called accordingly. For our design, we implemented the advisory control in MATLAB software that includes an add-on called Simulink. Simulink is a simulation tool that is useful in testing the

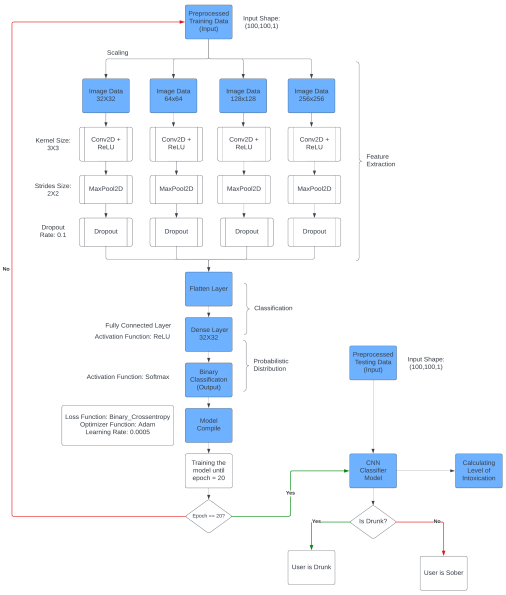

Fig. 8: CNN model architecture for intoxication detection.

advisory control that we designed. The designed advisory control is primarily based on fuzzy logic.

1) Fuzzy Logic: Fuzzy logic manages uncertainty through graded truth values in fuzzy sets. Utilizing membership functions, it assigns degrees of membership. Employing linguistic rules, it connects inputs to outputs using "if-then" statements and connectives. Inference combines rules based on fulfillment degrees, and defuzzification converts fuzzy outputs into precise results or control signals.

2) Factors affecting reaction time of the driver: The baseline reaction time of the user is given as follows:

$$T_{RB} = 0.01 * \frac{Heart\ rate}{Respiratory\ rate} \tag{1}$$

A few of the external factors affecting reaction time that we are focusing on are intoxication, drowsiness, and BMI. We have created a fuzzy logic model that gives an additional reaction time parameter (say T_{RA}).

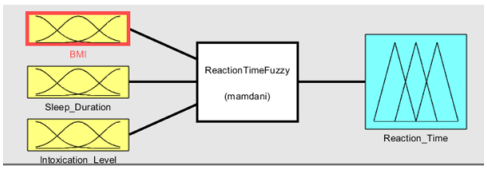

Fig. 9: Reaction time fuzzy model

Now total reaction time is $T_R = T_{RB} + T_{RA}$. From now on, we use this reaction time when needed.

3) Static Advisory Control: The static advisory control employs two fuzzy logics and a simulation of the possible collision with the static obstacle. A fuzzy logic for the reaction time and fuzzy logic to determine the deceleration limit and gain of the controller are performed. Simulink simulates a

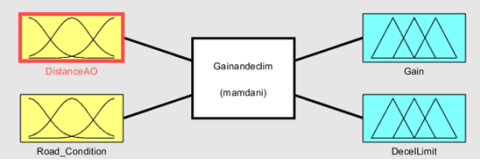

Fig. 10: Fuzzy Logic for Gain and deceleration limit of the controller

potential collision using available parameters. If autonomous mode prevents collision, no switch is needed. Otherwise, it compares user and autonomous collision times to make the appropriate switch.

4) Dynamic Advisory Control: Dynamic advisory control is also very similar to static advisory control, but varies in minor ways. A fuzzy to determine the deceleration limit based on average speed, a fuzzy logic that takes in distance between vehicle and dynamic obstacle, road condition and deceleration of the car ahead to produce deceleration of the vehicle and a fuzzy logic to determine switching based on deceleration and reaction time are performed. No model simulation is needed

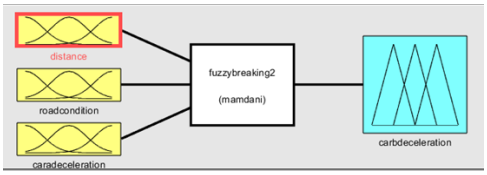

Fig. 11: Fuzzy Logic for deceleration

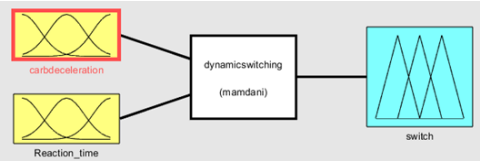

Fig. 12: Fuzzy Logic switching decision

for this advisory control.

5) Firebase Update: The advisory control decides whether there is a switch from autonomous mode to manual mode. This switching is notified to the user via updating the Firebase and the mobile application reading from there.

6) Testing and Validating: We evaluated our designed advisory controls by simulating them in Simulink with randomly generated parameters within their ranges. Adjusting membership functions and switching thresholds aimed to reduce switches and collisions. The performance of the optimized controls was assessed in 100 random scenarios,

V. Demonstration

This project demonstration setup includes a computer (with MATLAB) and android phone with internet connectivity. User inputs are taken via the designed app and users are notifed when there is a switch, that is decided by the advisory control

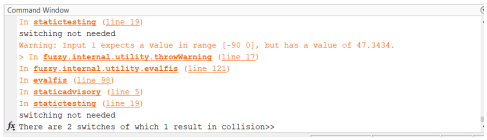

Fig. 13: Test results of static advisory

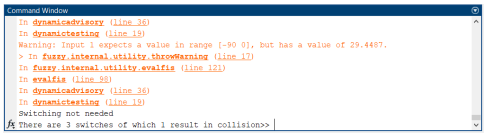

Fig. 14: Test results of dynamic advisory

in the computer. The app notifies the user about the switch as shown.

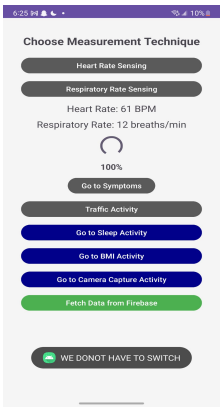

Fig. 15: Indication of switching to the user

VI. Conclusion

Conclusively, this project improves the switching decision of the advisory control with the help of additional parameters like intoxication level, sleep duration and BMI. The project makes use of Firebase for a seamless integration of the front end and back end, Machine-learning for measuring intoxication level and fuzzy logic for decision making.

VII. Project Resources

The source code, project documentation, and additional resources related to this project can be found in the GitHub repository:

https://github.com/shubhamX1438/GROUP39_CSE535.git

References

[1] T. Oyama, S. Tano and T. Arnould, "A tuning method for fuzzy inference with fuzzy input and fuzzy output," Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, Orlando, FL, USA, 1994, pp. 876-881 vol.2, doi: 10.1109/FUZZY.1994.343852.

[2] TensorFlow and Keras Documentation in Python.
- https://www.tensorflow.org/api_docs/python/tf/all_symbols
- https://www.tensorflow.org/guide/keras

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (June 2017), 84–90. https://doi.org/10.1145/3065386

[4] For dataset we have referred to the following github repository: https://github.com/ebenezer-isaac/intoxicated-face-identification