## Carla based feature pipeline

### 1. 1 Overview

This document summarizes the work completed so far on our Feature Extraction part for a driver drowsiness detection system using CARLA.

We have:

- Implemented data capture within CARLA's 0.9.13 version, keeping the vehicle at 15km/hr constant and only controlling left and right.

- Extracted key features from steering angle and lateral offset.

- Built and evaluated a drowsiness classification pipeline(offline).

- Conducted offline analysis comparing \*normal\*, \*moderate\*, and \*drowsy\* driving states.

### 1.2 Data Collection

- Recorded 5 minutes of driving data per condition (awake, moderate fatigue, drowsy) across 1 participant.

- Logged CSV fields:

- `timestamp`

- `steering\_angle\_rad(Carla.VehicleControl.Steer)(which I don’t think so is correct)

- `lateral\_offset`

- `lane\_id`

### 1.3 Feature Extraction

Implemented Python scripts to compute:

1. \*\*Steering Metrics\*\*

- Mean, STD, Min, Max, Median

- Skewness, Kurtosis

- Shannon Entropy

- Reversal rate (per minute)

A graph showing different sizes of objects

AI-generated content may be incorrect. A graph of a steering angle comparison

AI-generated content may be incorrect.

2. \*\*Lateral and Lane Metrics\*\*

- Standard deviation of lateral position (SDLP)

- Lane departure frequency (Hz)

- Lane keeping ratio

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

- Sliding‐window extraction (5 s windows) for real‐time feature streams, but done in an offline manner.

### 1.4 Offline Analysis

- `drowsiness\_analysis.py`:

- Reads CSVs for \*normal\*, \*moderate\*, \*drowsy\* sessions.

- Computes all above features per session.

- Generates comparative boxplots for steering angle and lateral offset.

- Summary CSV exported with rows for each condition.

### 1.5 Next Steps

1. \*\*Data Expansion\*\*: Collect 10 min per condition for ≥ 10 or 5 drivers.

2. \*\*Model Tuning\*\*: Retrain with a larger dataset, experiment with additional features (i.e. Camera Module)

3. Validating our model is working properly and is able to recognise the driving pattern.

## Camera feature update

1. Trained the fine-tuned Dlib model using the frames and the landmarks (annotations-auto.s2) from the dataset

2. Compared the trained landmarks with the annotations-auto, Following attached are few visualization results:

A child taking a selfie

AI-generated content may be incorrect.



3. Wrote a script that approximates the difference between the both landmarks positions based on the Euclidian distance, following is the printed results and plotted histogram:  
A graph with numbers and a line

AI-generated content may be incorrect.

***\*Results Clearly show that approximately all the predicted landmarks are 30pxls away***

Printed Results:  
  
=== SUMMARY ===

* Total Compared: 99102
* Mean Distance: 30.95
* Max Distance: 167.78
* Min Distance: 24.66

4. Moving forward in the pipeline to see the results on the EAR and MAR for the video 4-3,

Following are the results:

A graph showing a line of orange and blue lines

AI-generated content may be incorrect.

\****Results are even worse than the before***

5. Searching if we can fine-tune the media pipe but answer is no its a GOOGLE’S framework and we can’t fine tune it will compromise the backbone structure of it

6. Moving to our last best bet which is using the annotation-manual given in the dataset “***Face landmarks (obtained manually):****2D (in pixels) and 3D (in millimeters) annotations of 68 face landmarks for 720 hand-selected frames (averaging 51 per subject), resulting from manual processing;”*

Did not saved the results but it was not any better

7. We were ignoring the simple d-lib pre-trained model just because it converts the IR to RGB, so we printed the IR to RGB image just to see how does it look:  
Following are the results:  
  
**rgb = cv2.cvtColor(gray, cv2.COLOR\_GRAY2RGB)**  
It just **duplicates the single grayscale channel into 3 identical channels**. The result:

* Visually: It **looks identical** to the original IR image.
* Mathematically: No new information is introduced, and the image is not altered.
* Purpose: Some models just require a 3-channel input format.

8. After knowing that rgb conversion is just duplication, not changing any information we can now use the pre-trained d-lib model, for that we used   
  
shape\_predictor\_68\_face\_landmarks.dat

Following are the results:

comparison done random 5 frames of the 4-3  
A person taking a selfie

AI-generated content may be incorrect.  
  
A person with green dots on her face

AI-generated content may be incorrect.

A graph with purple lines

AI-generated content may be incorrect.

***\*Each frames wise mean distance <8 for the ones its >8 the person was moving too much get accurate landmarks***

* Mean Euclidean Distance (ALL frames in 4-3): 3.55 pixels
* Based on 1777 valid frames and 120836 total landmark pairs

Ear and MAR graph:  
  
  
A graph of blue and orange lines

AI-generated content may be incorrect.

Results are appropriate, checked all the peaks are actual yawns and even when there is no

## ROS & Docker update –

Regarding Feature Extraction and Fatigue Estimation team,

we have successfully installed Docker to support the modular development and integration of our software components.

We have developed a solid understanding of Docker’s key concepts such as images, containers, and volumes, and how these facilitate reproducibility and ease of collaboration across teams.

Currently, we are focusing on integrating ROS2 (Robot Operating System 2) with Docker.

We created nodes for Camera and Carla sensor. This integration is critical for interfacing with data streams from the physical setup and ensuring smooth communication with other components of the system, such as those responsible for data acquisition and real-time feedback.