# Driver Drowsiness Analytics

M. Panvi Tej
Department of Computer Science and Engineering
Mahindra University
B.Tech, Computer Science and Engineering
Email: panvitejm@gmail.com

***Abstract*—Driver fatigue, inattention, and micro-sleep are among the leading causes of road accidents. This paper presents a vision-based driver drowsiness analytics system implemented in Python using OpenCV and Google MediaPipe. The system tracks multiple behavioural cues including Eye Aspect Ratio (EAR), a yawn–chin distance ratio, hand-to-face proximity, and head vertical motion. These features are aggregated over a temporal window to compute a drowsiness index and a facial anomaly score. To avoid unstable predictions, we introduce asymmetric exponential smoothing, where risk scores increase quickly but decay slowly. The system outputs (i) a processed video with real-time overlays and (ii) an analytical dashboard summarizing per-frame metrics, status distribution, and behaviour history. Such analytics can support driver assistance systems, offline evaluations, and research on driver behaviour.**

***Index Terms*—Driver drowsiness, computer vision, MediaPipe, eye aspect ratio, temporal analytics, anomaly detection, behavioural monitoring.**

## I. Introduction

Road safety is a critical challenge worldwide, with a significant number of crashes linked to drowsy or inattentive driving [**?**]. Traditional approaches such as lane departure warning and steering pattern monitoring focus on vehicle dynamics, but they do not directly capture the internal state of the driver. As a result, vision-based driver monitoring systems that analyze the driver's face and behaviour have gained increasing attention [**?**], [1].

In this project, we design an end-to-end *driver drowsiness analytics* system that operates on video input and produces both:

- a **real-time annotated video** with on-screen status indicators, and
- a **post-processing analytics dashboard** that summarises driver behaviour over time.

The core of the system uses Google MediaPipe to obtain facial and hand landmarks in each frame. From these landmarks, we derive geometric features such as Eye Aspect Ratio (EAR), a yawn–chin ratio, hand-to-face distance, and head motion. These features are accumulated over a sliding temporal window and combined into a drowsiness index and an anomaly score. Finally, asymmetric exponential smoothing is applied so that risk increases are reflected quickly while decreases are gradual, reducing rapid score oscillations and making the analytics more interpretable.

## II. Related Work

Vision-based drowsiness detection has been widely studied. Many works rely on eyelid closure metrics such as PERCLOS (percentage of eyelid closure over time) and blink duration [2]. The Eye Aspect Ratio (EAR), computed from eye landmarks, is a popular geometric measure that falls when the eyes are closed and rises when they are open. Other approaches use mouth landmarks to detect yawning, or head pose and nodding patterns to infer fatigue.

Recent systems leverage deep learning models to classify drowsy versus alert states directly from images or video clips. However, these models may require large annotated datasets and significant computational resources. In contrast, landmark-based approaches using MediaPipe provide a lightweight alternative that is easier to interpret and implement, which is particularly suitable for student projects and embedded systems.

The proposed work combines several of these ideas:

- EAR-based eye closure detection.
- Normalized mouth opening for yawning estimation.
- Hand-to-face proximity as a proxy for eye rubbing or face touching.
- Temporal aggregation and anomaly scoring to capture behavioural trends.
- A practical analytics dashboard for post-run evaluation.

## III. System Overview

Fig. 1 illustrates the high-level architecture of the proposed system from an analytical perspective. The pipeline can be divided into three stages:

1) **Feature extraction** using MediaPipe.
2) **Temporal drowsiness and anomaly modelling**.
3) **Visualization and analytics**.

### A. Input Acquisition and Pre-Processing

The input is a video file or camera stream. Each frame is read using OpenCV, converted from BGR to RGB, and passed to three MediaPipe solutions:

- **Face Detection**: provides a bounding box and rough face region.
- **Face Mesh**: provides 468 facial landmarks, including eyes, lips, nose, and chin.
- **Hands**: provides 21 landmarks per detected hand.

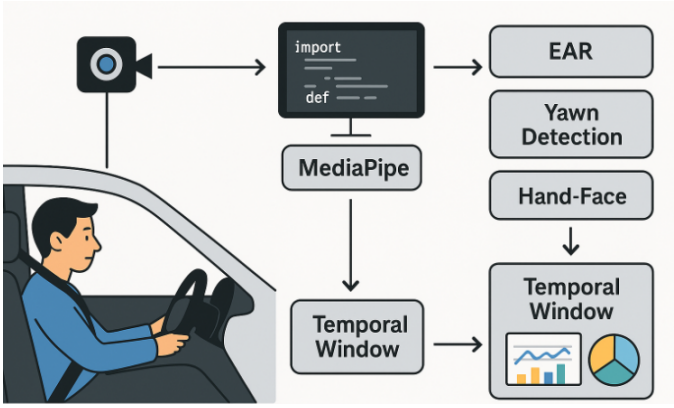


Fig. 1. Overall driver drowsiness analytics pipeline: video frames are processed with MediaPipe to extract landmarks, features (EAR, yawn–chin ratio, hand–face distance, head motion) are computed, temporal windows and smoothing generate scores, and final results are visualised in an annotated video and analytics dashboard.

The face detection is used to estimate a coarse face center and size. The Face Mesh output is used for precise feature geometry, and the hand landmarks are used to calculate hand-to-face distance.

## IV. Feature Extraction

### A. Eye Aspect Ratio (EAR)

The Eye Aspect Ratio (EAR) is a widely used metric for eye closure detection. For each eye, six landmarks are used: two horizontal corners and four vertical points. Let $p_1, \ldots, p_6$ be these points. The EAR is defined as:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\,\|p_1 - p_4\|}. \tag{1}$$

The numerator corresponds to the sum of the vertical eyelid distances, while the denominator corresponds to the horizontal eye width. When the eye is open, $EAR$ is relatively high; when the eye is closed, $EAR$ decreases. The system computes EAR for both eyes and uses their average. If this average falls below a threshold (e.g. $EAR < 0.23$) the frame is considered to have "eyes closed".

### B. Yawn–Chin Ratio

To detect yawning, the system measures the vertical distance between the upper lip and the chin and normalizes it by an estimate of face width. Let $d_{\text{chin-lip}}$ denote the distance between the upper lip landmark and the chin landmark, and $d_{\text{face}}$ the distance between two outer eye landmarks:

$$R_{\text{yawn}} = \frac{d_{\text{chin-lip}}}{d_{\text{face}}}. \tag{2}$$

A large value of $R_{\text{yawn}}$ indicates that the mouth is wide open. If $R_{\text{yawn}}$ exceeds a threshold (e.g. 0.22), the frame is tagged as "yawning".

### C. Hand-to-Face Proximity

Hand landmarks from MediaPipe Hands are used to estimate the average position of the index fingertip and middle fingertip. Let $(h_x, h_y)$ denote this hand point, and $(f_x, f_y)$ denote the estimated face center. The Euclidean distance between hand and face is:

$$d_{\text{hand-face}} = \sqrt{(h_x - f_x)^2 + (h_y - f_y)^2}. \tag{3}$$

This distance is normalized by a face size estimate (e.g. the face bounding box size), producing a ratio. If the ratio falls below a threshold (e.g. 0.9), the system considers the hand to be "near the face". This may correspond to rubbing eyes, touching the face, or partially occluding the camera.

### D. Head Motion

To detect head nodding or vertical movements, the system records the vertical coordinate of the nose across frames. Within a temporal window, it computes the difference between the maximum and minimum vertical positions. If this difference exceeds a preset threshold, the behaviour is flagged as head nodding.

## V. Temporal Drowsiness and Anomaly Modelling

### A. Sliding Window and Binary Histories

To capture behaviour over time, the system maintains several binary histories using Python `deque` structures:

- $H_{\text{eye}}$: 1 if eyes closed in a frame, 0 otherwise.
- $H_{\text{yawn}}$: 1 if yawning detected, 0 otherwise.
- $H_{\text{hand}}$: 1 if hand near face, 0 otherwise.
- $H_{\text{moderate}}$: 1 if both eyes closed and yawning, 0 otherwise.

The window length is defined by `WINDOW_SECONDS`, multiplied by the video frame rate. At each frame, the oldest entries are dropped automatically as new ones are appended.

From these histories, the system computes ratios:

$$r_{\text{eye}} = \frac{1}{N}\sum H_{\text{eye}}, \tag{4}$$

$$r_{\text{yawn}} = \frac{1}{N}\sum H_{\text{yawn}}, \tag{5}$$

$$r_{\text{hand}} = \frac{1}{N}\sum H_{\text{hand}}, \tag{6}$$

$$r_{\text{mod}} = \frac{1}{N}\sum H_{\text{moderate}}, \tag{7}$$

where $N$ is the number of frames in the current window. These ratios form the basis of the temporal analytics, describing how frequently each behaviour occurred in the recent past.

### B. Temporal Drowsiness Score

A temporal drowsiness index is computed from these ratios by a weighted sum:

$$S_{\text{temp,raw}} = 60\,r_{\text{eye}} + 25\,r_{\text{yawn}} + 10\,r_{\text{hand}} + 20\,r_{\text{mod}}. \tag{8}$$

The score is clipped to the range $[0, 100]$ for interpretability. Higher values of $S_{\text{temp,raw}}$ correspond to more frequent recent eye closure, yawning, and combined events, and can be visualised as a continuous risk curve over time.

### C. Anomaly Score

In parallel, a facial anomaly score is computed by inspecting patterns in the EAR, yawning, head motion, and hand-to-face behaviour over the same window:

- standard deviation of EAR ($\sigma_{\text{EAR}}$) to detect irregular blinking patterns;
- yawn ratio frequency to detect frequent yawning;
- vertical nose movement amplitude for head nodding;
- ratio of frames with hand near face.

If these exceed thresholds, additive contributions are made to an anomaly score $S_{\text{anom,raw}}$, which is again clipped to $[0, 100]$. The system also keeps textual notes such as "Irregular eye closure pattern" or "Frequent yawning in recent window" for analytical interpretation.

### D. Asymmetric Exponential Smoothing

Direct use of $S_{\text{temp,raw}}$ and $S_{\text{anom,raw}}$ can lead to unstable behaviour because scores may jump rapidly when short-term patterns change. To stabilise the display and advice, the system applies asymmetric exponential smoothing.

Let $S_{\text{raw}}(t)$ be a raw score at time $t$, and $S_{\text{smooth}}(t-1)$ the previously smoothed score. The new smoothed score is:

$$S_{\text{smooth}}(t) = \begin{cases} S_{\text{smooth}}(t-1) + \alpha_{\text{up}}\big(S_{\text{raw}}(t) - S_{\text{smooth}}(t-1)\big), \\ \qquad \text{if } S_{\text{raw}}(t) > S_{\text{smooth}}(t-1), \\ S_{\text{smooth}}(t-1) - \alpha_{\text{down}}\big(S_{\text{smooth}}(t-1) - S_{\text{raw}}(t)\big), \\ \qquad \text{otherwise.} \end{cases} \tag{9}$$

Here, $\alpha_{\text{up}}$ is larger than $\alpha_{\text{down}}$, meaning that scores rise relatively quickly when risk increases but decay slowly when risk decreases. This produces a "sticky" risk curve that is less sensitive to short, transient improvements but still responsive to emerging drowsiness. From an analytics point of view, this smoothing makes trends easier to interpret and reduces noise in the time-series plots.

This smoothing is applied separately to the temporal drowsiness score and the anomaly score.

### E. Advice Logic

The final driver state is determined from a combined score:

$$S_{\text{combined}} = 0.7\, S_{\text{temp,smooth}} + 0.3\, S_{\text{anom,smooth}}. \tag{10}$$

Based on $S_{\text{combined}}$, the system assigns the following levels:

- **Alert**: $S_{\text{combined}} < 30$.
- **Behavioural Fatigue**: $30 \le S_{\text{combined}} < 60$.
- **Critical Anomalous Behaviour**: $S_{\text{combined}} \ge 60$.

If the face is not visible, a special status "Driver Not Visible" is used. If the hand is very close to the face but the drowsiness score is low, the system may report "Face Occluded".

Each status is associated with a short message and a list of suggestions (e.g., "Consider taking a short break", "Immediate rest is recommended") which are overlaid on the video and summarised statistically in the dashboard.

## VI. Implementation Details

The system is implemented in Python using the following major libraries:

- **OpenCV**: video input/output, frame processing, text overlays, simple plotting on frames.
- **MediaPipe**: Face Mesh, Face Detection, and Hands landmark detection.
- **NumPy**: array operations and distance computations.
- **Pandas and Matplotlib**: for the final analytics dashboard.

The core processing loop reads frames from the video, processes them through MediaPipe, updates temporal histories, computes scores, and writes the annotated frame to an output video file. The following pseudo-code summarises the main steps:

```
for each frame in video:
   detect face, face mesh, hands
   compute EAR, yawn_ratio, hand_near, head_y

   update histories H_eye, H_yawn, H_hand,
       H_moderate
   compute temporal_raw and ratios
   compute anomaly_raw and notes

   temporal_smooth = smooth_score(temporal_raw)
   anomaly_smooth = smooth_score(anomaly_raw)

   advice = get_driver_advice(temporal_smooth,
                        anomaly_smooth,
                        ratios, notes)

   draw overlays (text, bars, history) on frame
   save frame to output video
   store per-frame metrics for analytics
```

Listing 1. High-level processing loop (pseudo-code).

At the end of the video, a Pandas DataFrame is built that contains time stamps, raw and smoothed scores, event ratios, and status labels for every frame. This DataFrame is exported to a CSV file and used to generate plots such as temporal trends and status distributions.

## VII. Results and Analytics Dashboard

From an analytical point of view, the system provides two complementary outputs:

1) **Frame-level analytics in the video:** each frame includes instantaneous EAR and yawn ratio, boolean flags for eyes closed, yawning, and hand near face, windowed percentages of each event, the current status label, and horizontal bars showing the smoothed temporal and anomaly scores. This allows qualitative inspection of how the model responds to different behaviours.
2) **Aggregated analytics in the dashboard:** the exported DataFrame is used to compute and visualise:
   - time-series plots of temporal drowsiness and anomaly scores,
   - time-series plots of event ratios (eye closure, yawning, hand-to-face),
   - distributions of time spent in Alert, Behavioural Fatigue, and Critical states.

These analytics make it easy to understand not only whether the driver was drowsy, but also *when* and for how long, and which behaviours (eye closure, yawning, face touching) contributed most to the risk score. Such insights can be valuable for tuning thresholds, comparing different participants, or evaluating how external conditions (time of day, environment) affect fatigue.

## VIII. Conclusion and Future Work

This paper presented a driver drowsiness analytics system based on MediaPipe landmarks, temporal windowing, and asymmetric score smoothing. The system combines eye closure, yawning, hand-to-face proximity, and head motion into a drowsiness index and a facial anomaly score, and visualizes these in both an annotated video and an analytical dashboard.

Future work may include:

- integrating head pose estimation (pitch, yaw, roll) more explicitly;
- incorporating deep learning models for more robust state classification;
- evaluating the system on larger datasets with ground truth labels;
- deploying the model on embedded hardware for real-time in-car use.


## Acknowledgment

The authors would like to thank **Prof. Rahul Roy**, Department of Computer Science and Engineering, Mahindra University, for his guidance in selecting this project, his continuous support throughout the work, and his valuable feedback on both the analytical design and implementation of the system. They also thank the faculty and mentors at Mahindra University for their overall support.


## References


[1] Q. Ji, Z. Zhu, and P. Lan, "Real-time nonintrusive monitoring and prediction of driver fatigue," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 4, pp. 1052–1068, 2004.

[2] W. W. Wierwille and L. A. Ellsworth, "Evaluation of driver drowsiness by trained raters," *Accident Analysis & Prevention*, vol. 26, no. 5, pp. 571–581, 1994.