RESEARCH ARTICLE OPEN ACCESS

A Real-Time Driver Drowsiness Detection Systems

Sanjay R*, Dr. K. Thenmozhi**
*(Department of Information Technology, Dr. N.G.P. Arts and Science College, Coimbatore, India)
**(Department of Information Technology, Dr. N.G.P. Arts and Science College, Coimbatore, India
Email: thenmozhi@drngpasc.ac.in)

----------**********----------

Abstract:

Driver drowsiness is a leading cause of road fatalities worldwide. This paper presents SafeDrive Monitor, a real-time driver drowsiness detection system developed using Python, Flask, OpenCV, and Google MediaPipe Face Landmarker. The system continuously captures webcam frames, extracts 478 3D facial landmarks, and computes the Eye Aspect Ratio (EAR) bilaterally to determine eye closure state. When the EAR remains below a threshold of 0.25 for 2.5 consecutive seconds, the system classifies the driver as drowsy and triggers an audible alarm. A Flask-based web interface streams the annotated live feed and provides session control via start/stop routes. Evaluation on a CPU-only machine demonstrates approximately 28 frames per second with alarm latency under 2.7 seconds. The system requires no specialized hardware, offering a cost-effective, deployable road safety solution. Limitations including platform-specific audio and lighting sensitivity are identified, and directions for future work are discussed.

Keywords — **Driver Drowsiness Detection, Eye Aspect Ratio, MediaPipe Face Landmarker, OpenCV, Flask, Real-Time Monitoring, Road Safety, Computer Vision.**

----------**********----------

I. INTRODUCTION

Road traffic accidents attributable to driver drowsiness represent a critical public health challenge. The World Health Organization (WHO) estimates that road traffic injuries are the eighth leading cause of death globally, with fatigue and inattention cited in a significant proportion of collisions. The National Highway Traffic Safety Administration (NHTSA) in the United States estimates that drowsy driving is responsible for approximately 91,000 police-reported crashes, 50,000 injuries, and 800 deaths annually. The actual figures are believed to be substantially higher due to under-reporting.

Drowsiness impairs cognitive performance in ways analogous to alcohol intoxication. A driver who has been awake for 18 consecutive hours exhibits impairment equivalent to a blood alcohol concentration (BAC) of 0.05%, and after 24 hours this rises to the equivalent of 0.10% BAC. Unlike alcohol or drug impairment, drowsiness is difficult to self-assess accurately, making external monitoring systems particularly important.

Existing driver monitoring technologies broadly fall into three categories: vehicle-based methods (lane deviation, steering wheel oscillation, pedal pressure), physiological methods (EEG, EOG, ECG sensors), and visual/behavioural methods (facial analysis, eye tracking, head pose estimation). While physiological methods offer high accuracy, they require intrusive wearable sensors unsuitable for mass deployment. Vehicle-based approaches suffer from delay and false triggers on poor roads. Visual methods, enabled by advances in deep learning and computer vision, offer a non-intrusive, camera-only solution compatible with consumer hardware.

This paper presents SafeDrive Monitor, a web-based, real-time drowsiness detection system that employs Google MediaPipe's Face Landmarker model to extract precise per-frame facial geometry and computes the Eye Aspect Ratio (EAR) to infer eye closure. The system is packaged as a Flask web application that streams an annotated MJPEG feed to a browser, requiring only a standard webcam and a general-purpose CPU. The contributions of this work are:

- A complete, open-source implementation integrating MediaPipe Face Landmarker in VIDEO mode with real-time EAR-based drowsiness logic.
- A lightweight Flask web architecture enabling browser-based monitoring without requiring desktop application installation.
- Empirical evaluation of detection latency, frame rate, and alarm responsiveness on consumer hardware.
- Identification of limitations and a roadmap for cross-platform, multi-cue enhancement.

The remainder of this paper is structured as follows. Section II reviews related literature. Section III details the system methodology. Section IV describes the implementation. Section V presents experimental results and discussion. Section VI concludes with future directions.

II. LITERATURE REVIEW

The detection of driver drowsiness has attracted sustained research interest across multiple disciplines including computer vision, biomedical engineering, and human factors. This section reviews the major streams of prior work relevant to the proposed system, organized by detection modality.

A. Physiological Signal-Based Methods

Early and highly accurate drowsiness detection systems relied on physiological signals. Electroencephalography (EEG) captures cortical activity directly associated with sleep states, and studies by Yin et al. [1] demonstrated classification accuracies exceeding 90% in distinguishing alert from drowsy states using EEG power spectral density features. Electrooculography (EOG) measures eye movement potentials and has been used to detect slow eye movements characteristic of sleep onset [2]. Electrocardiography (ECG)-based heart rate variability (HRV) analysis provides additional correlates of fatigue [3].

Despite their accuracy, physiological methods require subjects to wear electrodes or specialized hardware, making them unsuitable for everyday driving scenarios. Electrode placement is time-consuming, signals are susceptible to motion artefacts, and long-term skin contact causes discomfort. These drawbacks have motivated a shift toward camera-based behavioural approaches.

B. Vehicle Dynamics-Based Methods

An alternative non-intrusive approach monitors vehicle behaviour rather than the driver directly. Metrics such as lateral lane deviation [4], steering wheel entropy, throttle pedal activity, and following distance variation have been used as proxies for driver alertness. Krajewski et al. [5] achieved reasonable drowsiness classification using steering wheel movement features extracted at 10 Hz. However, such methods exhibit significant lag — detectable only after the vehicle has already begun to deviate — and produce high false-positive rates on winding or bumpy roads. They are also vehicle-specific and require OBD integration.

C. Eye-Blink and Eyelid-Based Detection

Among visual behavioural cues, eye state is the most reliable indicator of drowsiness. PERCLOS (Percentage of Eyelid Closure Over the Pupil) was established as the gold-standard drowsiness measure by Dinges et al. [6] and is formally defined as the proportion of time the eye is at least 80% closed over a rolling measurement window. PERCLOS-based systems using infrared cameras demonstrated strong correlation with subjective sleepiness ratings and performance degradation.

Soukupova and Cech [7] introduced the Eye Aspect Ratio (EAR), a geometrically elegant metric computable from six 2D facial landmarks per eye. The EAR is defined as the ratio of the sum of the

two vertical eye distances to twice the horizontal eye distance, approximating the degree of eye openness in a single scalar value robust to minor scale and illumination changes. Their work demonstrated real-time blink detection at frame rates exceeding 30 fps using the dlib 68-point face model, establishing EAR as a widely adopted baseline for camera-based drowsiness detection.

Subsequent work by Dwivedi et al. [8] combined EAR with Mouth Aspect Ratio (MAR) for yawning detection, improving system robustness by incorporating a second drowsiness cue. Their two-cue system achieved an F1 score of 0.91 on the YawDD dataset.

D. Deep Learning Approaches

The availability of large annotated datasets and advances in convolutional neural networks (CNNs) have enabled end-to-end drowsiness classification without hand-crafted features. Weng et al. [9] proposed a hierarchical temporal deep belief network trained on facial video sequences, demonstrating improved performance over EAR baselines in controlled environments. Ghoddoosian et al. [10] introduced a two-stream CNN architecture combining optical flow and appearance features for drowsiness detection robust to illumination changes.

Recurrent architectures, particularly Long Short-Term Memory (LSTM) networks, have been applied to capture temporal dynamics of drowsiness progression. Pan et al. [11] combined CNN-extracted spatial features with LSTM-modeled temporal dependencies, achieving state-of-the-art results on the NTHU Driver Drowsiness Detection dataset. Transformer-based models have more recently been applied, leveraging self-attention mechanisms to model long-range temporal dependencies in eye and head pose sequences [12].

While deep learning methods achieve superior accuracy, they typically require GPU inference hardware, large training datasets, and substantial engineering effort for deployment. They are therefore less suitable for lightweight, embedded, or browser-based applications than geometry-based approaches.

E. MediaPipe and Lightweight Landmark Models

MediaPipe [13], developed by Google Research, is a cross-platform framework for building real-time machine learning pipelines. The MediaPipe Face Landmarker model provides 478 3D facial landmarks per frame, derived from a lightweight neural network optimized for CPU inference. Unlike the dlib 68-point model, MediaPipe landmarks include iris tracking and detailed per-eyelid points, enabling more precise EAR computation with a larger set of reference landmarks.

Bazarevsky et al. [14] described the BlazeFace detector underlying MediaPipe, demonstrating sub-millisecond face detection on mobile CPUs. The Face Landmarker model extends this with mesh-level geometry, enabling robust landmark localization under moderate head pose variation and partial occlusion.

Recent implementations of EAR-based detection using MediaPipe by Sharma et al. [15] demonstrated improved robustness compared to dlib-based counterparts, particularly under head yaw angles up to ±30 degrees, owing to the 3D nature of the landmark predictions which partially compensate for perspective distortion.

F. Web-Based and Embedded Deployment

Most prior drowsiness detection systems have been demonstrated as standalone desktop applications or embedded in prototype vehicle hardware. Web-based deployment, which enables cross-device access without installation overhead, has received less attention. Flask-based video streaming architectures [16] using MJPEG multipart HTTP responses provide a practical approach to browser-delivered computer vision, as employed in this work. Browser-native approaches using TensorFlow.js and WebRTC have also been explored [17], though they require more complex client-side implementation.

The proposed SafeDrive Monitor system addresses the gap identified in the literature: a complete, non-intrusive, CPU-only, web-deployed drowsiness detection system combining MediaPipe's state-of-the-art face landmarker with

the established EAR metric in a lightweight, reproducible open-source package.

III. METHODOLOGY

A. Eye Aspect Ratio (EAR)

The Eye Aspect Ratio is computed from six facial landmark coordinates arranged around each eye: p1 and p4 at the outer and inner eye corners, and p2, p3, p5, p6 positioned along the upper and lower eyelids. The EAR formula is:

$$EAR = (||p2-p6|| + ||p3-p5||) / (2 \times ||p1-p4||)$$

In the proposed system, MediaPipe landmark indices 33, 160, 158, 133, 153, 144 are used for the left eye and 362, 385, 387, 263, 373, 380 for the right eye. The bilateral EAR is the mean of both eyes, improving robustness when one eye is momentarily occluded or partially off-camera. During natural eye opening, EAR remains approximately 0.25–0.35. During full closure it approaches zero.

B. Drowsiness Decision Logic

The detection pipeline applies two sequential criteria to classify a driver as drowsy. First, the bilateral EAR must fall below the closure threshold ($\theta = 0.25$). Second, this sub-threshold condition must persist continuously for a minimum duration ($\tau = 2.5$ seconds). These parameters were selected to minimise false positives from natural blinks (typically 150–400 ms) while capturing pathological eye closure indicative of microsleep onset.

State is managed via two variables: eye_closed_start records the timestamp of the first sub-threshold frame, and alarm_on prevents repeated alarm triggers within a single drowsy episode. The alarm resets and the system returns to the AWAKE state as soon as EAR exceeds θ.

C. System Architecture

The system is structured as a three-layer architecture: (1) the sensing layer acquires raw frames from the webcam via OpenCV VideoCapture; (2) the processing layer performs landmark detection and EAR-based inference in detection.py; and (3) the presentation layer delivers annotated frames as a browser-rendered MJPEG stream via Flask. A plain-text flag file (detection_active.txt) provides stateful on/off control without requiring a database or session management.

TABLE I SYSTEM COMPONENT SUMMARY

Component	Role / Technology
app.py (Flask)	Web server, routing, MJPEG streaming
detection.py	EAR calculation, MediaPipe inference, alarm
MediaPipe Face Landmarker	478 3D landmark detection (VIDEO mode)
OpenCV (cv2)	Webcam capture, frame annotation
winsound	Audible alarm — 2500 Hz beep (Windows)
HTML/CSS/JS	Browser-based UI and live video display
detection_active.txt	Lightweight state flag (on/off control)

IV. IMPLEMENTATION

A. Backend: app.py

The Flask application defines four HTTP routes. The index route (GET /) renders the main HTML template. The /start and /stop routes write '1' or '0' respectively to detection_active.txt and redirect to the index. The /video route returns a multipart/x-mixed-replace MJPEG response generated by the generate_frames() generator in detection.py, conditional on the detection flag. This design cleanly separates control logic from streaming logic.

The file-based state management pattern is intentionally minimal: it avoids the complexity of Flask sessions or database state, and ensures that webcam and MediaPipe resources can be released when detection is stopped, preventing unnecessary CPU utilisation during idle periods.

B. Detection Engine: detection.py

At module load time, the MediaPipe FaceLandmarkerOptions are configured with VIDEO running mode and a single-face target, and a FaceLandmarker instance is created from the bundled model asset (models/face_landmarker.task, 3.76 MB). OpenCV VideoCapture is opened on device index 0 (the default webcam).

The generate_frames() generator function implements the core detection loop. Each iteration reads a frame from the capture device, converts it from BGR to RGB, wraps it in a mediapipe.Image object, and calls detect_for_video() with an incrementing timestamp (33 ms per frame, approximating 30 fps). If face landmarks are detected, the generator extracts the six left-eye and six right-eye landmark coordinates in pixel space, computes the bilateral EAR via calculate_ear(), evaluates the drowsiness criteria, and applies OpenCV overlays. The frame is then JPEG-encoded and yielded as a multipart boundary segment.

The calculate_ear() function uses Python's math.dist() for Euclidean distance calculation, avoiding NumPy overhead for this lightweight computation. Landmark coordinates are scaled from normalised [0,1] to pixel space by multiplying by the frame's height and width.

C. Frontend Interface

The web interface is a single-page HTML5 application styled with a custom CSS design system using CSS custom properties for theming. The header features a gradient background (indigo-to-purple) with an animated floating car emoji logo. The main layout presents the live video section alongside a control panel card containing Start Detection and Stop Detection anchor-buttons.

The live feed is embedded as an HTML <img> element targeting the /video Flask route, which the browser continuously polls via HTTP streaming. JavaScript event listeners on the image element update a pulsing status indicator dot from 'Ready' to 'Active' on stream load, and back to 'Inactive' on error. The footer identifies the project as a Final Year Project developed using MediaPipe and Flask.

V. RESULTS AND DISCUSSION

A. Experimental Setup

Evaluation was conducted on a Windows 10 desktop with an Intel Core i5-8400 processor (6 cores, 2.8 GHz base, no discrete GPU) and 8 GB RAM. A standard USB 720p webcam was used at 30 fps capture rate. The Flask development server was run locally and accessed via Google Chrome. Testing was performed under typical indoor office lighting (approximately 300–500 lux) with the subject seated 50–70 cm from the camera.

B. Performance Metrics

The following performance characteristics were measured over a 10-minute monitoring session:

- Effective frame rate: 27–29 fps (limited by MediaPipe inference latency of 28–35 ms per frame).
- EAR computation overhead: less than 0.1 ms per frame (negligible).
- Alarm trigger latency: 2.51–2.68 seconds from onset of sustained eye closure, consistent with the 2.5-second threshold.
- CPU utilisation: 35–42% on a single core during active detection; drops to near 0% when detection is stopped.
- Memory footprint: approximately 180 MB resident (including Python runtime, MediaPipe model, and OpenCV).

TABLE II PERFORMANCE SUMMARY

Metric	Measured Value
Effective Frame Rate	~28 fps
MediaPipe Inference Latency	28–35 ms / frame
EAR Threshold (θ)	0.25
Drowsy Time Threshold (τ)	2.5 seconds
Alarm Trigger Latency	2.51–2.68 seconds

Alarm Frequency	2500 Hz, 1000 ms duration
CPU Utilisation (active)	35–42% (single core)
Memory Footprint	~180 MB
Platform	Windows 10, CPU-only
Hardware Required	Standard USB webcam

C. Qualitative Observations

The system reliably annotated frames with the current EAR value and displayed colour-coded status overlays (green AWAKE banner; red DROWSY banner). Alarm triggering was consistent and reproducible across repeated test sessions. The web interface rendered without perceptible latency on the local network.

Two categories of false positives were observed. First, prolonged voluntary blinks exceeding approximately 400 ms occasionally triggered a short DROWSY classification before the driver re-opened their eyes. Second, subjects wearing glasses experienced occasional landmark jitter at the inner canthus landmark (index 133/362), producing transiently low EAR values. These could be mitigated by temporal smoothing (e.g., applying an exponential moving average to EAR values before thresholding) or by adopting a PERCLOS-style windowed measure rather than a simple continuous timer.

D. Comparison With Prior Work

Compared to dlib-based EAR implementations [7], the MediaPipe-based approach demonstrated greater robustness to moderate head yaw (up to approximately ±25°) owing to 3D landmark predictions. Compared to CNN-based approaches [9][10], the proposed system trades some accuracy for significantly lower hardware requirements and simpler deployment, consistent with the practical goals of the project.

E. Limitations

Several limitations are acknowledged. The winsound module used for audio alarm is Windows-exclusive, restricting cross-platform deployment. Accuracy degrades significantly under low ambient lighting (below approximately 100 lux). The system detects only a single drowsiness cue (eye closure); yawning, head nodding, and microsleep episodes with open eyes are not captured. Detection is limited to a single face, and the system has not been validated for drivers with visual impairments or who habitually wear tinted glasses.

VI. CONCLUSION AND FUTURE WORK

This paper presented SafeDrive Monitor, a real-time, web-based driver drowsiness detection system integrating Google MediaPipe Face Landmarker with Eye Aspect Ratio analysis in a lightweight Flask architecture. The system operates effectively on CPU-only consumer hardware at approximately 28 fps, triggers an audible alarm within 2.7 seconds of drowsiness onset, and requires no specialized sensors or GPU acceleration.

The system addresses a clear gap in the literature by providing a complete, open-source, browser-deployable implementation that combines state-of-the-art face landmarking with established EAR-based detection in a minimal, reproducible package. Experimental evaluation confirmed reliable operation under standard indoor conditions, with known limitations under adverse lighting and head pose extremes.

Future work will pursue several enhancements. First, cross-platform audio support will be implemented using the playsound or pygame library to enable deployment on Linux and macOS. Second, an exponential moving average filter will be applied to EAR values to suppress blink-induced false positives. Third, yawning detection via Mouth Aspect Ratio (MAR) will be incorporated as a complementary drowsiness cue. Fourth, head pose estimation using the MediaPipe face geometry module will be integrated to detect nodding and forward head slump. Finally, user-specific adaptive thresholding will be explored, wherein a short calibration phase estimates the individual's natural

EAR distribution and sets personalised detection thresholds accordingly.

ACKNOWLEDGMENT

The authors wish to thank their project supervisor and the Department of Computer Science for their continued guidance, feedback, and infrastructure support throughout the development of this Final Year Project. The authors also acknowledge the open-source contributions of the MediaPipe, OpenCV, and Flask communities whose tools made this work possible.

REFERENCES

[1] Y. Yin et al., "EEG-Based Mental Fatigue Measurement Using Multi-Class Support Vector Machines with Confidence Estimate," Clinical Neurophysiology, vol. 120, no. 7, pp. 1199–1209, 2009.

[2] M. Johns, "A New Method for Measuring Daytime Sleepiness: The Epworth Sleepiness Scale," Sleep, vol. 14, no. 6, pp. 540–545, 1991.

[3] A. Murata et al., "Electrocardiographic Analysis of Heart Rate Variability for Assessment of Mental Workload," Ergonomics, vol. 37, no. 2, pp. 239–246, 1994.

[4] L. M. Bergasa et al., "Real-Time System for Monitoring Driver Vigilance," IEEE Trans. Intelligent Transportation Systems, vol. 7, no. 1, pp. 63–77, 2006.

[5] J. Krajewski et al., "An Ergonomic Standpoint Based Analysis of Distal and Proximal Indicators for Detecting Sleepiness," in Proc. HCI, 2009.

[6] D. F. Dinges and R. Grace, "PERCLOS: A Valid Psychophysiological Measure of Alertness as Assessed by Psychomotor Vigilance," FHWA Publication No. FHWA-MCRT-98-006, 1998.

[7] T. Soukupova and J. Cech, "Real-Time Eye Blink Detection Using Facial Landmarks," in Proc. 21st Computer Vision Winter Workshop, 2016.

[8] R. K. Dwivedi et al., "Drowsiness Detection by Eye State Analysis Using OpenCV and Haar Cascade Classifier," in Proc. ICCCNT, 2014.

[9] C. H. Weng et al., "Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network," in Proc. ECCV Workshops, 2016.

[10] R. Ghoddoosian et al., "A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection," in Proc. CVPR Workshops, 2019.

[11] W. Pan et al., "Driver Drowsiness Detection with CNN-LSTM," in Proc. ICSP, 2020.

[12] Z. Liu et al., "Transformer-Based Driver Drowsiness Detection from Facial Landmarks," Sensors, vol. 22, no. 15, p. 5793, 2022.

[13] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," arXiv:1906.08172, 2019.

[14] V. Bazarevsky et al., "BlazeFace: Sub-Millisecond Neural Face Detection on Mobile GPUs," arXiv:1907.05047, 2019.

[15] A. Sharma et al., "Eye Aspect Ratio Based Driver Drowsiness Detection Using MediaPipe," in Proc. ICCCA, 2022.

[16] M. Grinberg, Flask Web Development, 2nd ed. O'Reilly Media, 2018.

[17] A. Papakostas et al., "Browser-Based Real-Time Drowsiness Detection with TensorFlow.js," in Proc. IISA, 2021.

