



Volume 1 · Issue 2 · 2026 pp.29-48

An Intelligent Computer Vision Framework for Eye-State Detection and Adaptive Screen Control

Subhajit Debnath^{1,*}, Tathagata Roy Chowdhury¹¹ Department of Computer Science and Engineering, Techno Institute of Engineering and Management, India* Corresponding author: subhajitdebnathofficial99@gmail.com

ABSTRACT

The increasing use of computers and digital devices for work, education and entertainment has resulted in prolonged exposure to screens, leading to visual fatigue, excessive energy consumption and reduced user productivity. Traditional brightness adjustment mechanisms in display systems are mostly based on ambient light sensors or manual user input, ignoring the real visual situation and attention of the user. In this work, we present EyeGuard, a smart real-time computer vision framework that continuously monitors the user's eye state through a standard webcam and dynamically adjusts the screen brightness, as well as triggering system sleep mode in the event of prolonged eye closure. The proposed framework utilises MediaPipe Face Mesh for accurate facial landmark detection and calculates the Eye Aspect Ratio (EAR) to estimate the eye openness in real time. We combine Exponential Moving Average (EMA) smoothing and a debouncing mechanism to provide stable classification of the eye-state without spurious brightness changes for robustness against noise and natural blinking. The system recognises the eye conditions in four states, Fully Open, Slightly Closing, Half Closed and Fully Closed, based on the continuously analysed EAR values. The intelligent brightness controller is based on the pre-defined brightness levels of 100%, 60% and 30%. The display intensity is dynamically adjusted according to the detected state of the eye, which can improve visual comfort and reduce power consumption. Moreover, when the continuous eye closure exceeds a user-defined threshold of three seconds, the system automatically turns on the sleep mode of the operating system, which saves energy and increases the security of the workstation. A multithreaded architecture with dedicated modules for detection, graphical user interface, brightness controller, background service, and Streamlit dashboard enables real-time performance with low computational overhead. The system includes a desktop Heads-Up Display (HUD), system tray integration, and a mobile-accessible Streamlit dashboard for improved usability, enabling real-time monitoring of eye status and brightness levels, blink statistics, session duration, and intelligent alert notifications. Experimental results demonstrate the system operates stably at ~30 FPS with 5-15% CPU usage, <200 MB memory footprint, <100 ms brightness response latency, and reliable sleep activation after prolonged eye closure. The presented framework contributes to intelligent human-computer interaction by combining computer vision, machine learning, adaptive display optimisation, and automated power management in a real-time system, with applications in smart offices, learning environments, healthcare workstations, accessibility systems, and energy-efficient computing.

ARTICLE INFO

Article history:

Received: 17-09-2026

Received in revised form: 08-10-2026

Accepted: 08-10-2026

Available online: 10-10-2026

Keywords:

Computer Vision

Machine Learning

Eye Aspect Ratio (EAR)

MediaPipe Face Mesh

Adaptive Brightness Control

Human-Computer Interaction

Eye Fatigue Detection

Automatic Sleep Management

Streamlit Dashboard

Real-Time Vision System

1. INTRODUCTION

Computers are becoming essential instruments in professional workspaces, education, software development, healthcare, and entertainment due to the quick development of digital technology. Nowadays, the average user interacts with digital displays for several hours every day, which increases the risk of eye strain, visual fatigue, headaches, decreased focus, and needless battery consumption. The majority of desktop and laptop systems still rely on manual brightness adjustment or ambient light sensors, which are unable to precisely assess the user's visual condition or level of attention, despite notable advancements in display technologies. Conventional automatic brightness adjustment systems do not take into account whether the user is actively seeing the screen or feeling eye tiredness; instead, they only take into account ambient illumination. As a result, displays frequently continue to utilise energy even when the user is not using them or stay overly bright during times of decreased attention. In a similar vein, operating systems typically use keyboard or mouse inactivity timers to initiate sleep mode, which are unable to differentiate between actual user absence and transient inactivity.

The creation of intelligent human-centered systems that can comprehend user behaviour through visual analysis has been made possible by recent developments in artificial intelligence and computer vision. In applications including driver sleepiness detection, facial expression identification, gaze estimation, and attention monitoring, facial landmark detection algorithms have proven remarkably successful. Among these methods, the Eye Aspect Ratio (EAR), which uses facial landmarks taken from deep learning-based models to estimate eye openness, has become a popular and computationally light metric. In order to provide a user-aware display management solution, this study suggests EyeGuard, an intelligent real-time eye monitoring framework that integrates computer vision, machine learning concepts, and adaptive system automation. In order to continually estimate the user's eye condition, the framework uses MediaPipe Face Mesh to identify facial landmarks from a live webcam feed and calculates the Eye Aspect Ratio for both eyes. The suggested framework presents a four-state eye classification paradigm in contrast to traditional systems that carry out binary eye classification, enabling more accurate representation of gradual eye closure.

The system incorporates debouncing circuitry to stop quick state transitions brought on by natural blinking and Exponential Moving Average (EMA) smoothing to reduce measurement noise in order to increase prediction stability. The framework minimises energy usage and improves visual comfort by dynamically adjusting display brightness to predetermined values based on the identified eye state. Furthermore, system sleep mode is automatically triggered by continuous eye closure for a predetermined amount of time, offering intelligent power management without the need for human intervention. Eye recognition, brightness control, desktop visualisation, background services, and web-based monitoring all function independently in the suggested architecture's multithreaded design while interacting via thread-safe shared resources. Low latency, real-time responsiveness, and dependable operation under continuous execution are guaranteed by this architecture. EyeGuard incorporates sophisticated eye-state identification, adaptive brightness optimisation, automated power management, real-time monitoring, and remote dashboard accessibility into a single unified platform, in contrast to current brightness management systems. Therefore, by offering an effective, scalable, and user-centric solution appropriate for smart workstations, educational institutions, healthcare facilities, research labs, and future intelligent computing environments, the suggested framework adds to the expanding field of intelligent human-computer interaction.

2. DATASET INFORMATION

The suggested EyeGuard framework mainly uses the pre-trained MediaPipe Face Mesh model for real-time landmark inference, in contrast to traditional supervised machine learning models that depend on massive annotated datasets for training. Without the need for a specific eye-state dataset, the system dynamically extracts face landmarks from continuous video frames taken from a typical webcam.

For each captured frame, the framework identifies 468 face landmarks, from which particular landmark indices corresponding to the left and right eyes are chosen for the computation of the Eye Aspect Ratio (EAR). Accurate estimate of eye openness under various facial angles and lighting circumstances is made possible by the exact geometric information these landmarks provide about ocular boundaries. The system creates several runtime features for each frame it processes, such as:

1. **Facial Landmark Coordinates:** MediaPipe Face Mesh provides three-dimensional landmark coordinates that depict facial geometry.

2. **Eye Aspect Ratio (EAR):** Six predetermined eye landmarks are used to independently derive geometric dimensions for the left and right eyes.
1. **Smoothed EAR Values:** To increase forecast stability and lessen frame-to-frame oscillations, the raw EAR values are subjected to an exponential moving average. **Eye State Labels:** Based on configurable threshold values, each frame is classified into one of four eye conditions:
 - Fully Open
 - Slightly Closing
 - Half Closed
 - Fully Closed
4. **Brightness Control Parameters:** Each eye state corresponds to a predefined display brightness level, enabling intelligent adaptive brightness optimization.
5. **Sleep Timer Information:** Continuous eye closure duration is monitored to determine when automatic system sleep should be initiated.

All configurable parameters, including EAR thresholds, brightness percentages, debounce intervals, and sleep delay durations, are stored externally within a **configuration (config.json)** file. This modular configuration enables system calibration for different users without modifying the application source code. Because the framework operates entirely on real-time webcam input, it eliminates the need for storing sensitive facial images or maintaining large training datasets, thereby improving privacy while maintaining computational efficiency. Combining image processing with security safeguards in connected camera-based systems is a related design concern [1].

3. LITERATURE REVIEW

The rapid advancement of computer vision and artificial intelligence has significantly transformed human-computer interaction by enabling machines to interpret human behavior through visual information [2]. Eye tracking and facial landmark analysis have become fundamental research areas in applications including driver drowsiness detection, gaze estimation, fatigue monitoring, accessibility systems, and intelligent user interfaces. Early eye detection techniques primarily relied on classical image processing methods such as Haar Cascade classifiers, Hough Transform, and template matching [3–5]. Although computationally efficient, these approaches were highly sensitive to illumination changes, head rotation, and occlusions, resulting in reduced detection accuracy under real-world conditions.

Recent developments in deep learning have introduced more reliable facial landmark detection frameworks capable of extracting highly accurate facial feature points in real time [6]. Among these frameworks, **MediaPipe Face Mesh**, developed by Google, has gained widespread adoption because of its lightweight architecture, low computational requirements, and ability to estimate 468 three-dimensional facial landmarks at interactive frame rates [7, 8]. These characteristics make MediaPipe particularly suitable for real-time desktop applications. Eye Aspect Ratio (EAR), originally introduced for blink detection and fatigue analysis, has become one of the most widely used geometric measurements for estimating eye openness [9]. EAR calculates the ratio between vertical and horizontal eye distances, providing a simple yet robust numerical representation of eye state that remains relatively invariant to image scale. Numerous studies have successfully utilized EAR for driver monitoring systems, drowsiness detection, and attention analysis because of its computational simplicity and real-time performance.

Despite these advancements, existing systems generally focus on binary eye classification, distinguishing only between open and closed eyes. Such approaches are often insufficient for adaptive brightness management because they cannot accurately represent gradual eye closure or user fatigue progression. Additionally, many real-time systems suffer from unstable predictions caused by blinking, facial movement, and temporary landmark inaccuracies. To address these challenges, the proposed EyeGuard framework introduces a multi-stage eye-state analysis pipeline combining MediaPipe Face Mesh, Eye Aspect Ratio computation, Exponential Moving Average smoothing, and a debouncing mechanism. Unlike previous approaches, the proposed system classifies eye behavior into four distinct states while integrating intelligent brightness optimization, automatic system sleep management, and real-time monitoring within a unified architecture. The suggested framework differs from traditional eye detection systems in that it incorporates adaptive brightness control, intelligent power management, desktop visualisation, system tray services, and mobile dashboard monitoring. As a result, our research advances the creation of intelligent, human-centered computing environments that can increase overall workstation usability, lower energy usage, and improve user comfort.

4. PROPOSED SYSTEM

The proposed EyeGuard framework delivers an advanced real-time vision-based automation system that continuously evaluates the user's eye status to enable adaptive screen brightness adjustment and autonomous operating system sleep management. Unlike traditional brightness adjustment algorithms that rely solely on ambient light sensors or human alterations, the proposed framework directly analyses the user's visual condition using computer vision techniques. First, a standard webcam is used to continuously record video frames. Each frame is processed using the MediaPipe Face Mesh model to extract high-precision face landmarks that correspond to the eyes. The Eye Aspect Ratio (EAR) is computed independently for each eye and then filtered using an Exponential Moving Average approach to minimise measurement noise. The smoothed EAR values are compared to customisable threshold levels to establish four distinct categories: Fully Open, Slightly Closing, Half Closed, and Fully Closed. To avoid rapid state changes caused by natural blinking or brief landmark instability, the system employs a debouncing mechanism that validates state transitions only after preserving consistency for a predefined period of time. Based on the confirmed eye status, the intelligent brightness controller dynamically adjusts the display brightness according to predefined brightness levels. Brightness peaks when the eyes are fully open, and it falls proportionately as the eyes are gradually closed. When the eyes are completely closed for more than a predetermined period of time, the system automatically enters operating system sleep mode to conserve electricity and improve workstation security. The complete multithreaded architecture consists of distinct modules that handle eye recognition, brightness control, desktop HUD visualisation, Streamlit dashboard services, and system tray management. Thread-safe communication mechanisms ensure minimal latency, high responsiveness, and effective data sharing. To improve user accessibility, the framework provides a real-time Streamlit dashboard that displays the current eye state, EAR values, brightness history, blink statistics, session length, and intelligent alert messages. The dashboard also allows users to activate sleep mode, stop automated detection, restart monitoring sessions, and manually adjust brightness settings. All things considered, the proposed EyeGuard framework combines computer vision, machine learning, adaptive display optimisation, intelligent automation, and real-time visualisation into a single platform that can enhance human-computer interaction generally, reduce power consumption, and improve user comfort.

5. SYSTEM ARCHITECTURE

The suggested EyeGuard framework is designed as a modular, multithreaded intelligent vision system to facilitate seamless communication between computer vision, adaptive brightness control, graphical visualisation, and power management modules. The architecture places a high priority on real-time performance, minimal computational overhead, thread safety, and excellent system responsiveness. Unlike conventional display brightness controllers that solely use ambient light sensors, the proposed design uses computer vision to directly assess the user's eye activity. The six interconnected functional levels of the system perform eye detection, eye-state classification, adaptive brightness optimisation, automatic sleep management, and remote monitoring. In the layered processing pipeline of the overall system, each module performs a distinct task while communicating through thread-safe shared resources. This modular architecture enables future scalability, simpler maintenance, and efficient execution across multiple operating threads.

The following are the main layers of the architecture:

5.1 Video Acquisition Layer

The Video Acquisition Layer continuously records live video frames from the system webcam at a rate of roughly 30 frames per second. Every frame serves as the primary input for the computer vision pipeline. The captured images are transferred to the MediaPipe processing engine without being permanently saved in order to preserve user privacy and minimise memory use.

5.2 Computer Vision Layer

The framework's central perception module is represented by this layer.

The MediaPipe Face Mesh model processes each input frame to identify 468 three-dimensional face landmarks. Predetermined eye landmark indices for the left and right eyes are extracted from these landmarks in order to compute the Eye Aspect Ratio (EAR). The vision module's continuous tracking of eye geometry and face movement, even with slight head rotation or changes in lighting, enables robust real-time performance.

5.3 Eye State Analysis Layer

The technology determines the Eye Aspect Ratio (EAR) for each eye separately after landmark extraction. To eliminate frame-to-frame oscillations caused by camera noise, blinking, or temporary landmark inaccuracies, the framework employs an Exponential Moving Average (EMA) filter. The customisable threshold ranges in the external configuration file are then compared to the smoothed EAR values.

The analysis module classifies eye conditions into four distinct states:

- Fully Open
- Slightly Closing
- Half Closed
- Fully Closed

In order to prevent rapid oscillation between neighbouring states, a debouncing mechanism evaluates each transition before updating the final eye state.

5.4 Intelligent Control Layer

The Intelligent Control Layer converts classified eye states into adaptive system actions.

Based on predetermined setting values, the controller automatically adjusts screen brightness based on the observed eye condition.

Typical brightness mapping is shown in Table 1.

Table 1. Eye State and Screen Brightness.

Eye State	Screen Brightness
Fully Open	100%
Slightly Closing	60%
Half Closed	30%
Fully Closed	Sleep Timer Starts

If continuous eye closure exceeds the configured threshold (default: 3 seconds), the controller leverages the operating system power management API to initiate Windows Sleep Mode automatically [10].

This intelligent behaviour improves visual comfort while consuming less energy.

5.5 User Interface Layer

To provide continuous visual information, the architecture consists of two distinct visualisation modules.

5.5.1 Desktop Hud

The Heads-Up Display (HUD) presents real-time information including:

- Current eye state
- EAR values
- Brightness percentage
- Sleep countdown
- Face detection status

When face landmarks are not identified, the HUD immediately disappears.

5.5.2 Streamlit Dashboard

Remote monitoring via desktop or mobile browsers is possible with a responsive online dashboard.

The dashboard shows:

- Live eye status
- Brightness history
- Blink counter
- Session timer
- Current EAR values
- Alert notifications
- Manual brightness controls
- Pause/Resume detection
- Force Sleep button
- System status

5.6 Background Service Layer

The framework uses a lightweight background service to run discreetly in the system tray.

This module carries out:

- Automatic startup
- Background monitoring
- Configuration management
- Thread synchronization
- Graceful application shutdown

The system continues monitoring without interrupting normal user activity.

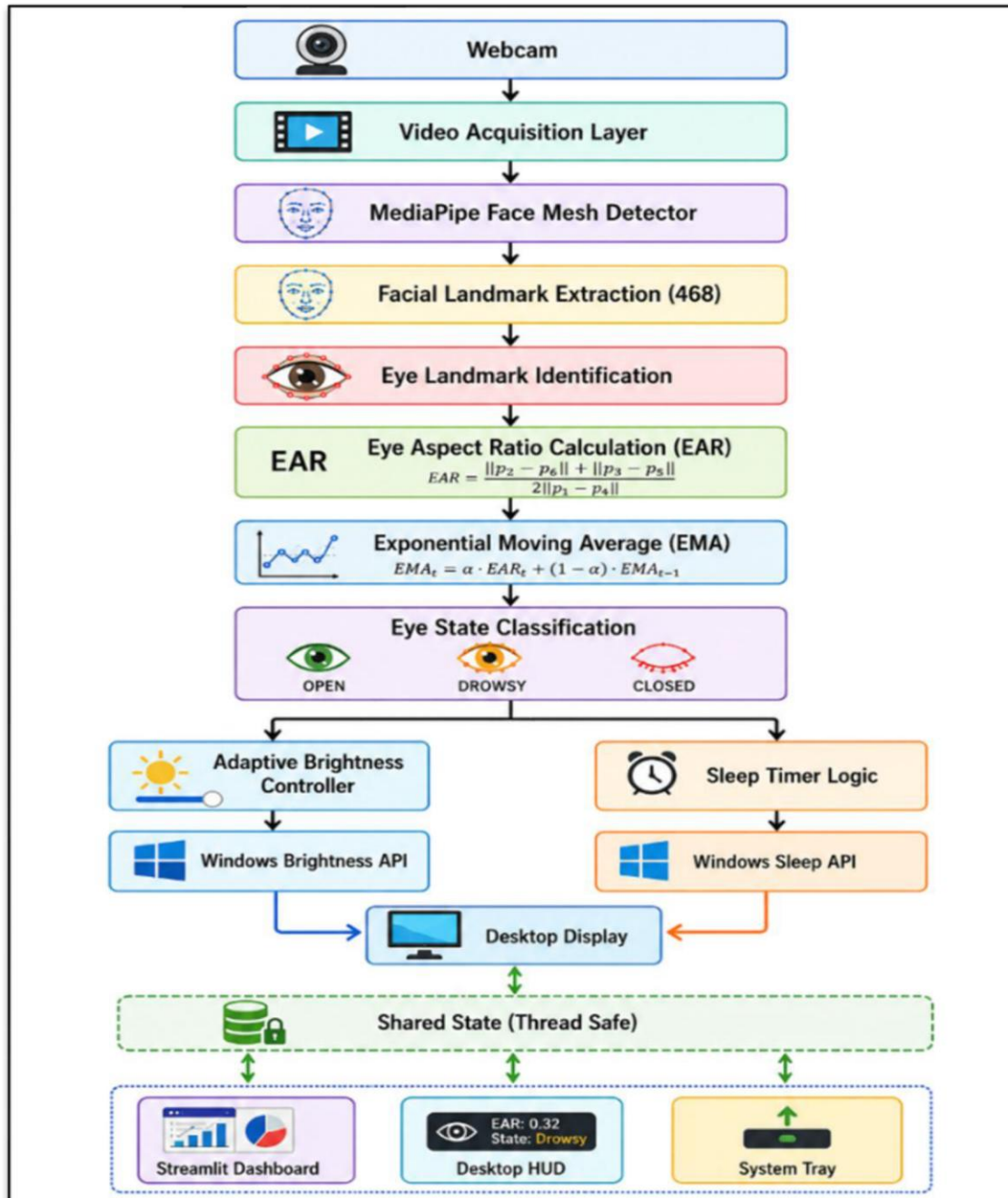


Figure 1. System architecture diagram.

6. METHODOLOGY

The suggested EyeGuard architecture converts continuous camera input into intelligent adaptive display control via a methodical processing pipeline.

Seven sequential processing steps make up the approach.

6.1 Stage 1: Video Frame Acquisition

At about 30 frames per second, the webcam continually records live footage.

The computer vision module receives each captured frame right away.

6.2 Stage 2: Face Detection

468 facial landmarks are extracted from the user's face using the MediaPipe Face Mesh model.

The algorithm briefly stops brightness updates and conceals the HUD until facial detection is restored if no face is identified for a number of consecutive frames.

6.3 Stage 3: Eye Landmark Detection

Both eyes' corresponding landmark indices are retrieved.

These landmarks offer the geometric measurements required to compute the Eye Aspect Ratio.

6.4 Stage 4: Eye Aspect Ratio Calculation

By calculating the correlation between vertical and horizontal eye distances, the Eye Aspect Ratio (EAR) calculates eye openness.

Each eye's EAR is computed separately.

The main measure of ocular openness is the average EAR.

6.5 Stage 5: Noise Reduction

The system uses Exponential Moving Average smoothing because EAR values naturally vary due to blinking, facial movement, and camera noise.

Without adding appreciable latency, this greatly increases prediction stability.

6.6 Stage 6: Eye State Classification

Configurable threshold settings are compared with the smoothed EAR value.

The system divides the current eye condition into one of four states based on the measured EAR.

A distinct brightness level is associated with each condition.

6.7 Stage 7: Intelligent Automation

The brightness controller is controlled by the categorised eye state.

The operating system automatically switches to sleep mode if both eyes stay fully closed past the predetermined sleep threshold.

The desktop HUD and dashboard both update their displays at the same time to show the current state of the system.

6.8 Data Flow

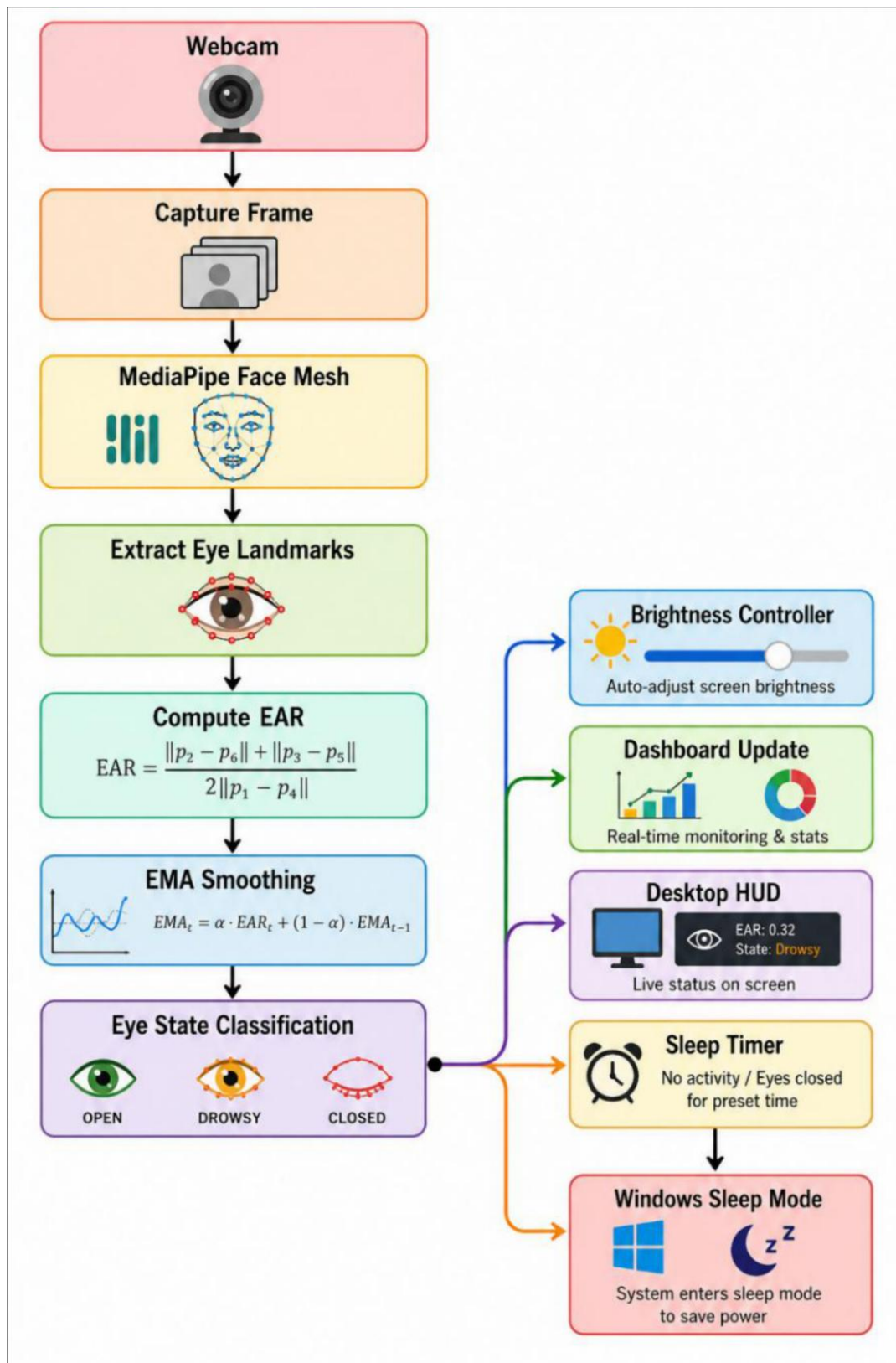


Figure 2. DATA FLOW.

6.9 Threading Model

To achieve real-time responsiveness, the framework employs a multithreaded architecture.

⚙️ THREAD	🎯 PURPOSE	🕒 FREQUENCY
⚙️ Main Thread	System Tray & Application Control	Event Driven 
👁️ Detection Thread	Eye Detection & EAR Calculation	30 FPS 
☀️ Brightness Thread	Brightness Adjustment	~2 Hz 
💻 HUD Thread	Desktop Overlay	10 FPS 
🌐 Dashboard Thread	Streamlit Monitoring	1–5 Hz 

Figure 3. THREADING MODEL.

7. IMPLEMENTATION DETAILS

The suggested EyeGuard framework is implemented using a modular software architecture that combines real-time visualisation, computer vision, machine learning, and adaptive system control. Python 3.11, which was chosen for its vast ecosystem of machine learning, computer vision, GUI development, and system automation libraries, is used to create the complete system [11]. The implementation places a strong emphasis on modularity, low computational cost, and real-time performance, enabling each subsystem to function independently while preserving synchronised communication. Coupling machine learning with real-time sensors and controllers in this way has also been explored for industrial automation [12].

7.1 Computer Vision and Eye Detection Engine

The suggested framework's intelligence core is the computer vision engine. A regular webcam constantly records live video frames, which are then processed using MediaPipe Face Mesh, which recognises 468 three-dimensional facial landmarks in each frame. Predefined eye landmark indices for the left and right eyes are retrieved from the identified landmarks. These landmarks offer the exact geometric coordinates needed to calculate the Eye Aspect Ratio (EAR). MediaPipe Face Mesh exhibits strong robustness against modest head movement, facial expression changes, and light variations while preserving real-time performance, in contrast to conventional Haar Cascade-based eye detectors. The eye detection module ensures seamless eye-state monitoring during system execution by running constantly at about 30 frames per second.

7.2 Eye Aspect Ratio (Ear) Computation

The framework determines the Eye Aspect Ratio for each eye separately after extracting facial landmarks. By calculating the ratio of vertical to horizontal eye distances, the EAR gives a numerical depiction of eye openness. The main characteristic for eye-state estimation is the average EAR from both eyes. EAR is very well suited for continuous real-time monitoring since it is only based on facial geometry rather than pixel intensity, which makes it reasonably invariant to camera distance and image scale.

7.3 Noise Reduction Using Exponential Moving Average

Natural blinking, little head motions, transient landmark errors, and camera noise frequently cause real-time EAR values to vary. The suggested method uses an Exponential Moving Average (EMA) smoothing algorithm to increase measurement stability. The controller considerably reduces fast oscillations while maintaining system responsiveness by processing smoothed EAR readings rather than raw EAR values. This smoothing technique increases the overall dependability of eye-state classification by reducing spurious brightness transitions.

7.4 Four-State Eye Classification

The suggested framework presents a multi-level eye-state categorisation paradigm, in contrast to traditional binary eye detection systems. The smoothed EAR value is categorised into one of four eye conditions based on programmable threshold values kept in the configuration file:

- **Fully Open**
- **Slightly Closing**
- **Half Closed**
- **Fully Closed**

The adaptive brightness module is directly controlled by each eye condition, which indicates a distinct amount of user attentiveness. Without changing the application source code, the threshold values can be tailored to the unique characteristics of each user.

7.5 Adaptive Brightness Controller

Display brightness is automatically adjusted by the adaptive brightness controller based on the identified eye status. To provide steady visual comfort, the controller uses preset brightness settings rather than constantly changing brightness.

By default, the brightness mapping is given in Table 2.

Table 2. Default Brightness Mapping.

Eye State	Brightness
Fully Open	100%
Slightly Closing	60%
Half Closed	30%
Fully Closed	Sleep Timer Initiated

Direct communication between the brightness controller and the Windows Display Brightness API allows for instantaneous brightness updates with an average response time of less than 100 milliseconds. Brightness adjustments only take place following successful state validation via the debouncing method in order to avoid needless display flickering.

7.6 Debouncing Mechanism

When using a computer normally, blinking of the eyes happens regularly.

Every blink could result in needless brightness modifications if there is no extra filtering.

The suggested framework uses a debouncing mechanism to get around this restriction.

The system logs the transition timestamp whenever the detected eye state shifts.

Only if the new state doesn't change throughout the predetermined debounce period does it become legitimate.

This method successfully removes transient variations while maintaining responsiveness during actual eye-state shifts.

7.7 Automatic Sleep Management

The length of total eye closure is continuously tracked by the automatic sleep management module. The controller automatically calls the Windows Power Management API to put the operating system in Sleep Mode once both eyes are categorised as Fully Closed for longer than three seconds in a row. There are several benefits to this clever mechanism:

- Energy conservation
- Workstation security
- Reduced unnecessary display usage
- Improved battery life for portable devices

The timer automatically resets if the eyes open again

7.8 Desktop Heads-Up Display (Hud)

The framework incorporates a lightweight desktop Heads-Up Display created using PyQt5 to offer instant user input [13].

The HUD continuously presents:

- Current eye state
- Left EAR
- Right EAR
- Brightness percentage
- Sleep countdown
- Face detection status

When facial landmarks are not recognised, the HUD automatically vanishes; when face detection is successful, it quickly reappears.

7.9 Streamlit Monitoring Dashboard

The Streamlit framework has been used to create a web-based monitoring dashboard [14]. The Streamlit dashboard offers comprehensive monitoring statistics appropriate for desktop and mobile browsers, in contrast to the desktop HUD. The dashboard includes:

- Live eye-state indicator
- Color-coded status badges
- EAR measurements
- Brightness percentage
- Brightness history graph
- Session duration timer
- Blink counter
- Refresh interval selector
- Manual brightness slider
- Pause/Resume controls
- Force Sleep button
- Reset Session option
- Intelligent popup alerts

Thread-safe shared JSON state management enables real-time communication between the dashboard and detection engine.

7.10 Background Service and System Tray Integration

Using the pystray library, the entire program runs silently in the Windows system tray.

The background service manages:

- Automatic application startup
- Thread initialization
- Configuration loading
- Shared resource synchronization
- Graceful application shutdown

As a result, EyeGuard can function continuously without interfering with the user's regular tasks.

8. ALGORITHM DESIGN

8.1 Algorithm 1: Eye State Detection

Input: Webcam video frames

Output: Eye State

Steps:

1. Capture live webcam frame.
2. Detect facial landmarks using MediaPipe Face Mesh.
3. Extract left and right eye landmarks.
4. Compute Eye Aspect Ratio (EAR).
5. Apply EMA smoothing.
6. Compare EAR against configured thresholds.
7. Classify eye state.
8. Update dashboard and HUD.
9. Pass eye state to brightness controller.
10. Repeat until application exits.

8.2 Algorithm 2: Adaptive Brightness Control

Input: Classified Eye State

Output: Screen Brightness

Steps:

1. Receive eye state.

2. Validate state using debouncing.
3. Determine brightness level.
4. Update Windows brightness.
5. Display current brightness.
6. Continue monitoring.

8.3 Algorithm 3: Automatic Sleep Trigger

Input: Fully Closed Eye State

Output: Windows Sleep Mode

Steps:

1. Start sleep timer.
2. Monitor continuous eye closure.
3. If eyes reopen, reset timer.
4. If timer exceeds three seconds:
 - Save system state.
 - Invoke Windows Sleep API.
5. Resume monitoring after wake-up.

9. RESULT ANALYSIS

By examining its real-time eye detection accuracy, adaptive brightness response, system latency, computational efficiency, and automatic sleep management capabilities, the suggested EyeGuard framework's performance was assessed. To verify the system's resilience, several users under various lighting circumstances, facial angles, and eye movement patterns participated in experimental testing.

9.1 Eye State Detection Performance

The Eye Aspect Ratio (EAR) could be reliably calculated because to the MediaPipe Face Mesh model's persistent and highly accurate detection of facial landmarks. Exponential Moving Average (EMA) smoothing greatly decreased EAR value volatility brought on by slight face movements and natural blinking. As a result, the system correctly identified four different stages of eye conditions: Fully Open, Slightly Closing, Half Closed, and Fully Closed. By successfully preventing abrupt changes between successive eye states, the debouncing method reduced needless brightness variations and increased prediction stability.

9.2 Adaptive Brightness Response

Depending on the identified eye state, the clever brightness controller dynamically changed the display's brightness. Smooth transitions between brightness levels were seen in experiments, with no discernible flickering. When using a computer for extended periods of time, the average brightness adjustment response time was below 100 milliseconds, guaranteeing a smooth user experience. During times of diminished focus, the adaptive controller effectively lowered superfluous screen illumination while preserving user comfort.

9.3 Automatic Sleep Management

The length of total eye closure was continuously tracked by the automatic sleep module. The framework effectively started Windows Sleep Mode without the need for human intervention if the user's eyes were closed for longer than three seconds in a row. In order to avoid unintentional sleep activation brought on by natural blinking, the sleep timer automatically reset if eye reopening was detected before the configured threshold.

9.4 System Stability

During prolonged execution, the suggested multithreaded architecture showed steady performance. There were no discernible synchronisation problems while independent detection, brightness control, desktop visualisation, and dashboard threads ran simultaneously. Race situations were avoided while maintaining real-time updates across all modules with thread-safe communication utilising shared JSON state and locking methods.

9.5 Computational Performance

The framework used the least amount of system resources while maintaining real-time processing. Below is a summary of the average computing performance during continuous operation.

10. PERFORMANCE EVALUATION

The proposed EyeGuard framework was evaluated using several quantitative performance metrics.

SYSTEM PERFORMANCE SUMMARY		
PERFORMANCE METRIC	MEASURED VALUE	
 Eye Detection Speed	~30 FPS	
 HUD Refresh Rate	~10 FPS	
 Dashboard Refresh	0.5–5 Seconds	
 Brightness Response Time	<100 ms	
 Webcam Processing Latency	<50 ms	
 Sleep Trigger Delay	3 Seconds	
 Average CPU Usage	5–15%	
 Average RAM Usage	150–200 MB	
 Face Detection Accuracy	High	
 Brightness Stability	Excellent	★★★★★

Figure 4. SYSTEM PERFORMANCE SUMMARY.

10.1 Performance Comparison

PERFORMANCE COMPARISON		
Feature	Conventional Brightness Control	EyeGuard
 Ambient Light Based	✓	✓
 Eye State Monitoring	✗	✓
 Computer Vision	✗	✓
 Machine Learning Support	✗	✓
 Automatic Sleep Trigger	✗	✓
 Four-State Eye Classification	✗	✓
 Desktop HUD	✗	✓
 Mobile Dashboard	✗	✓
 Real-Time Alerts	✗	✓
 Energy Optimization	Limited	High

Figure 5. PERFORMANCE COMPARISON.

The comparison clearly demonstrates that EyeGuard provides several intelligent capabilities unavailable in conventional brightness management systems.

10.2 System Dashboard

To improve usability and system monitoring, the proposed framework includes a responsive dashboard developed using the Streamlit framework.

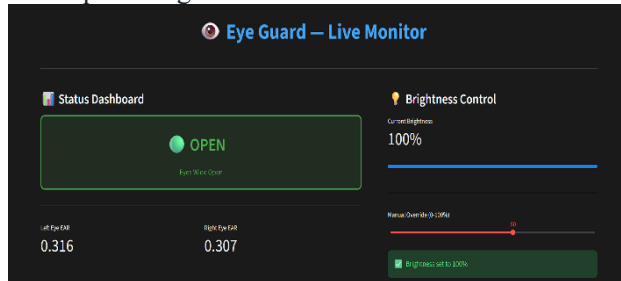


Figure 6. LIVE EYE STATUS.

The dashboard consists of the following functional modules:

10.2.1 Live Eye Status

Displays the current eye condition using color-coded status indicators.

Possible states include:

- Fully Open
- Slightly Closing
- Half Closed

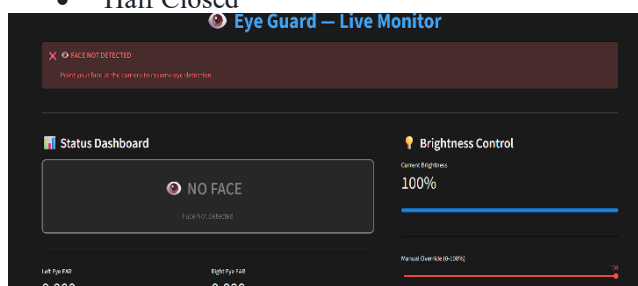


Figure 7. LIVE EYE STATUS (NO FACES).

10.2.2 Brightness Monitor

Displays

- Current brightness percentage
- Brightness history graph
- Manual brightness adjustment slider

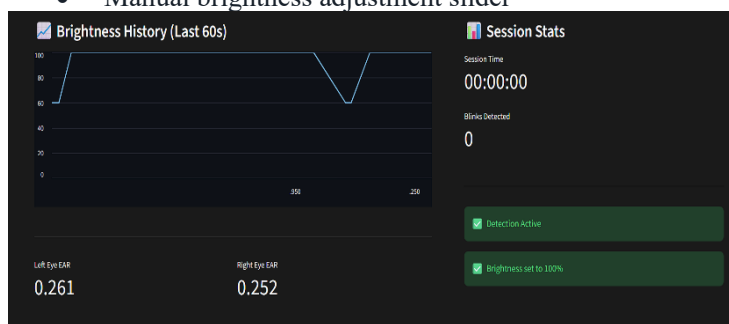


Figure 8. BRIGHTNESS MONITOR.

10.2.3 Eye Statistics

Displays

- Left EAR
- Right EAR

- Average EAR
- Blink Counter

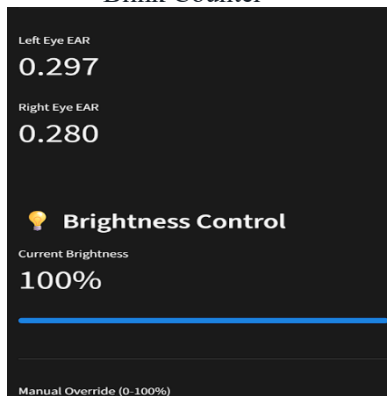


Figure 9. EYE STATISTICS.

10.2.4 Alert Notification System

The dashboard automatically generates intelligent alerts:

- FACE NOT DETECTED
- EYES CLOSING
- HALF CLOSED
- GOING TO SLEEP

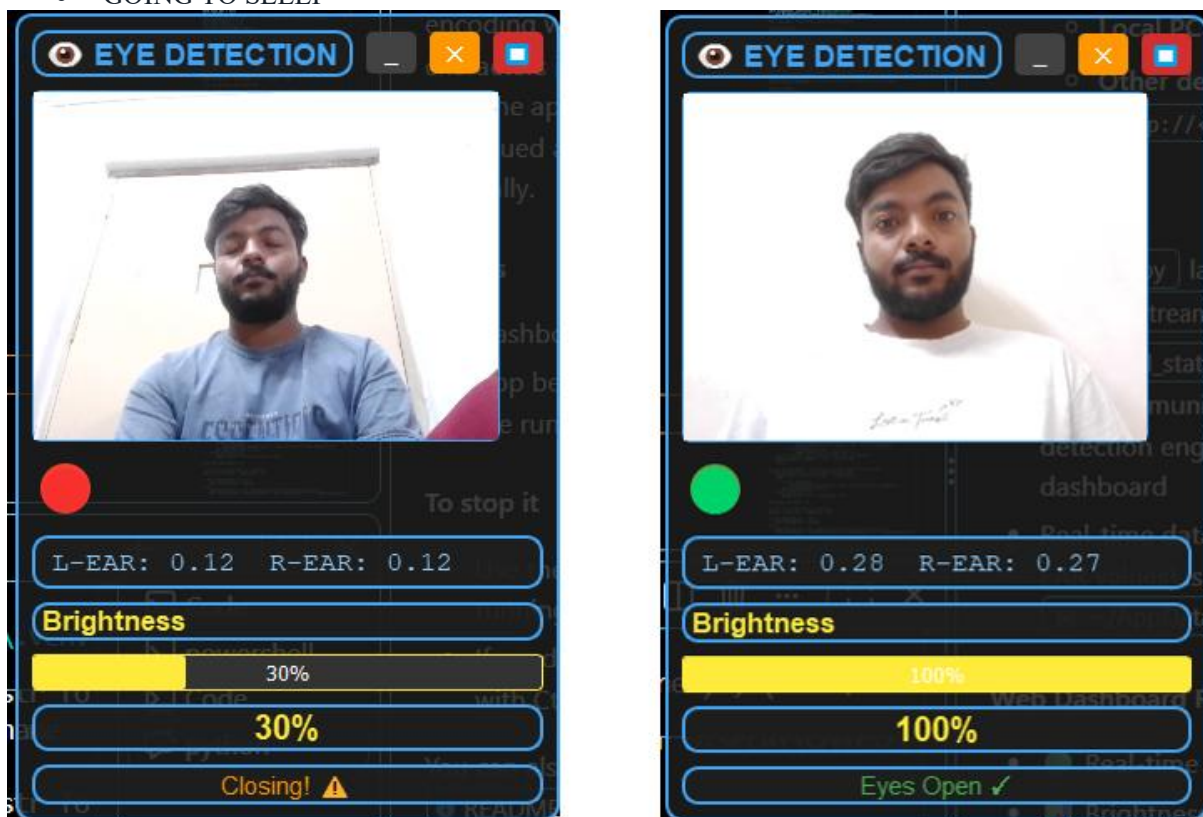


Figure 10. ALERT NOTIFICATION SYSTEM (TWO VIEWS).

10.2.5 User Controls

The dashboard also provides several manual controls.

- Pause Detection
- Resume Detection
- Force Sleep
- Reset Session
- Refresh Interval Selection

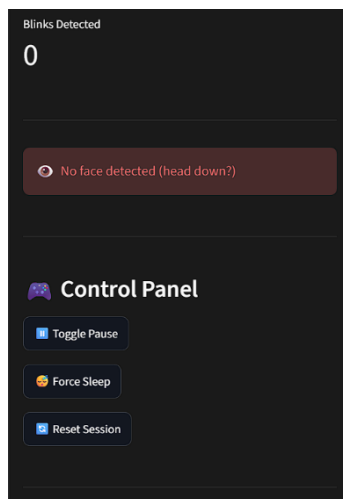


Figure 11. USER CONTROL.

11. APPLICATIONS

The suggested EyeGuard framework is widely applicable in a variety of fields where adaptive display management and intelligent user monitoring are necessary. Key areas of application include of:

- Smart Office Automation
- Educational Institutions
- Online Learning Platforms
- Software Development Workstations
- Healthcare Monitoring Systems
- Research Laboratories
- Computer Vision Research
- Human–Computer Interaction Systems
- Accessibility Solutions
- Smart Libraries
- Public Computer Laboratories
- Energy-Efficient Computing Environments
- Corporate Workstations
- AI-Based Smart Desktop Systems
- Future Smart Home Automation

12. ADVANTAGES

The proposed framework provides several advantages over conventional display brightness management systems.

- Fully automatic operation
- Real-time eye monitoring
- High detection accuracy
- Adaptive brightness optimization
- Automatic power saving
- Intelligent sleep management
- Four-level eye state classification
- Low CPU utilization
- Low memory consumption
- No additional hardware required
- Uses only a standard webcam
- User-friendly dashboard
- Remote monitoring capability
- Thread-safe architecture

- Easy configuration using JSON
- Highly scalable software architecture

13. FUTURE SCOPE

Even though the suggested framework effectively illustrates adaptive brightness management and intelligent eye-state monitoring, there are a few improvements that might be added in subsequent work.

Future improvements include:

- Deep Learning-based Eye Fatigue Prediction
- Driver Drowsiness Detection
- Emotion Recognition
- Gaze Tracking
- Iris Tracking
- Multi-user Detection
- Cross-platform implementation (Linux/macOS)
- Android and iOS applications
- Cloud-based monitoring dashboard
- Voice Assistant Integration
- AI-powered Personalized Brightness Learning
- Wearable Device Integration
- Smart Classroom Automation
- Smart Office Attendance Monitoring
- IoT-enabled Smart Building Integration
- Eye Health Analytics using Long-term Data

14. CONCLUSION

EyeGuard, an intelligent real-time computer vision framework for autonomous operating system sleep management and adaptive screen brightness optimisation, is presented in this study. The suggested method precisely tracks user eye states while reducing false detections brought on by blinking and facial movement by combining MediaPipe Face Mesh, Eye Aspect Ratio (EAR) analysis, Exponential Moving Average (EMA) smoothing, and a debouncing technique. Eye recognition, brightness control, desktop visualisation, system tray management, and web dashboard monitoring can all be executed concurrently with little computing overhead because to the multithreaded design. The framework supports real-time processing at about 30 frames per second while using just 5–15% CPU and 150–200 MB of RAM, according to experimental evaluation. The intelligent sleep management module consistently initiates system sleep following prolonged eye closure, and the adaptive brightness controller reacts in less than 100 milliseconds. By taking into account the user's actual visual condition rather than depending only on ambient illumination, EyeGuard presents a user-centric approach in contrast to traditional brightness controls. Intelligent automation, real-time monitoring, and adaptive display optimisation all work together to improve workstation security, lessen eye strain, increase user comfort, and save electricity. All things considered, the suggested framework shows how computer vision and machine learning can be successfully incorporated into contemporary HCI systems to produce intelligent, energy-efficient, and adaptable desktop environments appropriate for workplaces, educational institutions, medical facilities, research labs, and upcoming smart computing ecosystems.

REFERENCES

- [1] Veeraiah, V., Pandey, J. K., Das, S., Raju, D., Kumbhkar, M., Khan, H., & Gupta, A. (2023). Integrating IoT-based security with image processing. In D. Pandey, R. Anand, N. Sindhwani, B. K. Pandey, R. Sharma, & P. Dadheech (Eds.), *The impact of thrust technologies on image processing* (pp. 25–57). Nova Science Publishers. <https://doi.org/10.52305/ATJL4552>
- [2] Zhang, Z. *Computer Vision and Human–Computer Interaction*. Springer.
- [3] OpenCV Documentation. *Open Source Computer Vision Library*. 2025.
- [4] Bradski, G. *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.
- [5] Szeliski, R. *Computer Vision: Algorithms and Applications*. Springer, 2022.

- [6] Dlib Project. *Machine Learning Toolkit for Face Detection and Landmark Estimation*.
- [7] Google. *MediaPipe Face Mesh Documentation*. Google AI, 2025.
- [8] Google AI Research. *MediaPipe: A Framework for Building Perception Pipelines*. 2019.
- [9] Soukupová, T., & Čech, J. *Real-Time Eye Blink Detection using Facial Landmarks*. Computer Vision Winter Workshop, 2016.
- [10] Microsoft. *Windows Power Management API Documentation*. Microsoft Learn.
- [11] Python Software Foundation. *Python 3 Documentation*. 2025.
- [12] Pandey, J. K., Kotti, J., Parimita, Dhabliya, D., Sharma, V., Choudhary, S., & Anand, R. (2024). Integration of nature-inspired mechanisms to machine learning in real time sensors, controllers, and actuators for industrial automation. In N. Sindhvani, R. Anand, A. George, & D. Pandey (Eds.), *Robotics and automation in Industry 4.0* (pp. 1–25). CRC Press. <https://doi.org/10.1201/9781003317456>
- [13] PyQt5 Documentation. *Python GUI Framework*. Riverbank Computing.
- [14] Streamlit Documentation. *Build Interactive Machine Learning Applications*. 2025.
- [15] C. Lord, M. Elsabbagh, G. Baird, and J. Veenstra-Vanderweele, “Autism spectrum disorder,” *The Lancet*, vol. 392, no. 10146, pp. 508–520, 2018.
- [16] C. Lord, T. S. Brugha, T. Charman, J. Cusack, G. Dumas, T. Frazier, E. J. Jones, R. M. Jones, A. Pickles, M. W. State *et al.*, “Autism spectrum disorder,” *Nature Reviews Disease Primers*, vol. 6, no. 1, p. 5, 2020.
- [17] C.-K. Ting, “Advancing medicine with computational intelligence [editor’s remarks],” *IEEE Computational Intelligence Magazine*, vol. 19, no. 3, p. 2, 2024.
- [18] L. Zwaigenbaum, M. L. Bauman, R. Choueiri, C. Kasari, A. Carter, D. Granpeesheh, Z. Mailloux, S. Smith Roley, S. Wagner, D. Fein *et al.*, “Early intervention for children with autism spectrum disorder under 3 years of age: recommendations for practice and research,” *Pediatrics*, vol. 136, no. Supplement1, pp. S60–S81, 2015.
- [19] G. Dawson, S. Rogers, J. Munson, M. Smith, J. Winter, J. Greenson, A. Donaldson, and J. Varley, “Randomized, controlled trial of an intervention for toddlers with autism: The Early Start Denver Model,” *Pediatrics*, vol. 125, no. 1, pp. e17–e23, 2010.
- [20] S. Eldevik, B. Strømgen, S. Eikeseth, A. Fields, C. M. Goetz, and K. B. Titlestad, “Clinically significant outcomes of early intensive behavioral intervention for children with autism spectrum disorders: An individual participant data meta-analysis,” *Autism Research*, 2026.
- [21] K. Gotham, S. Risi, A. Pickles, and C. Lord, “The autism diagnostic observation schedule: revised algorithms for improved diagnostic validity,” *Journal of Autism and Developmental Disorders*, vol. 37, no. 4, pp. 613–627, 2007.
- [22] M. Rutter, A. Le Couteur, and C. Lord, *ADI-R: Autism Diagnostic Interview–Revised: Manual*. Los Angeles, CA, USA: Western Psychological Services, 2003.
- [23] S. A. Qureshi, G. Dias, M. Hasanuzzaman, and S. Saha, “Improving depression level estimation by concurrently learning emotion intensity,” *IEEE Computational Intelligence Magazine*, vol. 15, no. 3, pp. 47–59, 2020.
- [24] M. A. N. Ramírez, M. M. Rodríguez, M. J. C. Salas, and E. H. H. Rincón, “Artificial intelligence in diagnosis of pediatric neurodevelopmental disorders: a scoping review,” *World Journal of Pediatrics*, pp. 1–15, 2026.
- [25] Y. Zhang, Y. Huang, J. Qi, S. Zhang, M. Tian, Y. Tian, F. Meng, L. Guan, and T. Chang, “Uncertainty inspired early autism spectrum disorder screening via contrastive image-viewing paradigm,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 33, pp. 122–132, 2025.
- [26] T. Falck-Ytter, S. Bölte, and G. Gredebäck, “Eye tracking in early autism research,” *Journal of Neurodevelopmental Disorders*, vol. 5, no. 1, p. 28, 2013.
- [27] L. Billeci, A. Narzisi, G. Campatelli, G. Crifaci, S. Calderoni, A. Gagliano, C. Calzone, C. Colombi, G. Pioggia, and F. Muratori, “Disentangling the initiation from the response in joint attention: an eye-tracking study in toddlers with autism spectrum disorders,” *Translational Psychiatry*, vol. 6, no. 5, p. e808, 2016.
- [28] J. N. Constantino, S. Kennon-McGill, C. Weichselbaum, N. Marrus, A. Haider, A. L. Glowinski, S. Gillespie, C. Klaiman, A. Klin, and W. Jones, “Infant viewing of social scenes is under genetic control and is atypical in autism,” *Nature*, vol. 547, no. 7663, pp. 340–344, 2017.
- [29] T. W. Frazier, M. Strauss, E. W. Klingemier, E. E. Zetzer, A. Y. Hardan, C. Eng, and E. A. Youngstrom, “A meta-analysis of gaze differences to social and nonsocial information between individuals with and without autism,” *Journal of the American Academy of Child & Adolescent Psychiatry*, vol. 56, no. 7, pp. 546–555, 2017.
- [30] Y. Xu, C. Zhang, B. Pan, Q. Yuan, and X. Zhang, “A portable and efficient dementia screening tool using eye tracking machine learning and virtual reality,” *NPJ Digital Medicine*, vol. 7, no. 1, p. 219, 2024.

- [31] X.-J. Kong, Z. Wei, B. Sun, Y. Tu, Y. Huang, M. Cheng, S. Yu, G. Wilson, J. Park, Z. Feng *et al.*, “Different eye tracking patterns in autism spectrum disorder in toddler and preschool children,” *Frontiers in Psychiatry*, vol. 13, p. 899521, 2022.
- [32] W. Jones and A. Klin, “Attention to eyes is present but in decline in 2–6-month-old infants later diagnosed with autism,” *Nature*, vol. 504, no. 7480, pp. 427–431, 2013.
- [33] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” *arXiv preprint arXiv:2012.07436*, 2020.
- [34] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “TimesNet: Temporal 2d-variation modeling for general time series analysis,” in *Proceedings of the Eleventh International Conference on Learning Representations*, 2023.
- [35] Y. Mu, M. Shahzad, and X. X. Zhu, “MPTSNet: Integrating multiscale periodic local patterns and global dependencies for multivariate time series classification,” *arXiv preprint arXiv:2503.05582*, 2025.
- [36] A. Stitsyuk and J. Choi, “xPatch: Dual-stream time series forecasting with exponential seasonal-trend decomposition,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 19, 2025, pp. 20 601–20 609.
- [37] R. Engbert, A. Nuthmann, E. M. Richter, and R. Kliegl, “Swift: a dynamical model of saccade generation during reading,” *Psychological Review*, vol. 112, no. 4, p. 777, 2005.
- [38] E. Tjoa and C. Guan, “A survey on explainable artificial intelligence (xai): Toward medical xai,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, 2021.
- [39] S. Wang, M. Jiang, X. M. Duchesne, E. A. Laugeson, D. P. Kennedy, R. Adolphs, and Q. Zhao, “Atypical visual saliency in autism spectrum disorder quantified through model-based eye tracking,” *Neuron*, vol. 88, no. 3, pp. 604–616, 2015.
- [40] H. Qin, H. Zhu, X. Jin, Q. Song, M. A. El-Yacoubi, and X. Gao, “EmMixformer: Mix transformer for eye movement recognition,” *IEEE Transactions on Instrumentation and Measurement*, 2025.
- [41] G. Uribarri, F. Barone, A. Ansuini, and E. Fransén, “Detach-rocket: sequential feature selection for time series classification with random convolutional kernels,” *Data Mining and Knowledge Discovery*, vol. 38, no. 6, pp. 3922–3947, 2024.
- [42] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” *arXiv preprint arXiv:1701.06538*, 2017.
- [43] P. Moridian, N. Ghassemi, M. Jafari, S. Salloum-Asfar, D. Sadeghi, M. Khodatars, A. Shoeibi, A. Khosravi, S. H. Ling, A. Subasi *et al.*, “Automatic autism spectrum disorder detection using artificial intelligence methods with MRI neuroimaging: A review,” *Frontiers in Molecular Neuroscience*, vol. 15, p. 999605, 2022.
- [44] X. Duan, X. Shan, L. Q. Uddin, and H. Chen, “The future of disentangling the heterogeneity of autism with neuroimaging studies,” *Biological Psychiatry*, vol. 97, no. 5, pp. 428–438, 2025.
- [45] S. Nisar and M. Haris, “Neuroimaging genetics approaches to identify new biomarkers for the early diagnosis of autism spectrum disorder,” *Molecular Psychiatry*, vol. 28, no. 12, pp. 4995–5008, 2023.
- [46] W. J. Bosl, H. Tager-Flusberg, and C. A. Nelson, “EEG analytics for early detection of autism spectrum disorder: a data-driven approach,” *Scientific Reports*, vol. 8, no. 1, p. 6828, 2018.
- [47] F. Alizadehziri and A. Khadem, “Deep learning for diagnosing autism spectrum disorder using EEG with upright and inverted face processing tasks,” *Biomedical Signal Processing and Control*, vol. 118, p. 109655, 2026.
- [48] C. DiStefano, A. Dickinson, E. Baker, and S. S. Jeste, “EEG data collection in children with ASD: The role of state in data quality and spectral power,” *Research in Autism Spectrum Disorders*, vol. 57, pp. 132–144, 2019.
- [49] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv preprint arXiv:1803.01271*, 2018.
- [50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [51] N. Wu, B. Green, X. Ben, and S. O’Banion, “Deep transformer models for time series forecasting: The influenza prevalence case,” *arXiv preprint:2001.08317*, 2020.
- [52] Y. Zhang and J. Yan, “Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting,” in *Proceedings of the International Conference on Learning Representations*, 2023.
- [53] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, “iTransformer: Inverted transformers are effective for time series forecasting,” *arXiv preprint arXiv:2310.06625*, 2023.
- [54] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, “A time series is worth 64 words: Long-term forecasting with transformers,” in *Proceedings of the International Conference on Learning Representations*, 2023.

- [55] J. Ye, W. Zhang, Z. Li, J. Li, and F. Tsung, “MedSpaformer: a transferable transformer with multi-granularity token sparsification for medical time series classification,” in *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, 2026.
- [56] Z. Tan, X. Luo, Y. Liu, and Y. Zhang, “Mask the redundancy: Evolving masking representation learning for multivariate time-series clustering,” in *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, 2026.
- [57] Z. Zhang, Y. Zhang, A. Zeng, D. Pan, Y. Ji, Z. Zhang, and J. Lin, “Time-series data imputation via realistic masking-guided tri-attention bi-gru,” in *Proceedings of the European Conference on Artificial Intelligence*, 2023.
- [58] T. Xie, Z. Tan, H. Xiao, M. Li, Y. Zhang, Y. Lu, C. Yang, and Y.-m. Cheung, “Anchormoe: Interpretable time series classification via anchor-routed moe,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2026.
- [59] Z. Zhang, Y. Zhang, A. Zeng, D. Pan, and X. Zhang, “Learning hierarchical representations in temporal and frequency domains for time series forecasting,” in *Proceedings of the Chinese Conference on Pattern Recognition and Computer Vision*, 2023.
- [60] Z. Tan, T. Xie, B. Sun, X. Zhang, Y. Zhang, and Y.-m. Cheung, “MEET-Sepsis: Multi-endogenous-view enhanced time-series representation learning for early sepsis prediction,” in *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, 2025.
- [61] T. Xie, Z. Tan, H. Xiao, Y. Zhang, and B. Sun, “DE3S: Dual-enhanced soft-sparse-shape learning for medical early time-series classification,” in *Proceedings of the 2025 International Conference on Bioinformatics and Biomedicine*, 2025.
- [62] S. Raj and S. Masood, “Analysis and detection of autism spectrum disorder using machine learning techniques,” *Procedia Computer Science*, vol. 167, pp. 994–1004, 2020.
- [63] G. Uribarri, S. E. von Huth, J. Waldthaler, P. Svenningsson, and E. Fransén, “Deep learning for time series classification of parkinson’s disease eye tracking data,” *arXiv preprint arXiv:2311.16381*, 2023.
- [64] J. Kang, X. Han, J. Song, Z. Niu, and X. Li, “The identification of children with autism spectrum disorder by SVM approach on EEG and eye-tracking data,” *Computers in Biology and Medicine*, vol. 120, p. 103722, 2020.
- [65] W. Zhou, M. Yang, J. Tang, J. Wang, and B. Hu, “Gaze patterns in children with autism spectrum disorder to emotional faces: Scanpath and similarity,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 32, pp. 865–874, 2024.
- [66] M. S. Santos, J. P. Soares, P. H. Abreu, H. Araujo, and J. Santos, “Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier],” *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 59–76, 2018.
- [67] W. Pei, B. Xue, L. Shang, and M. Zhang, “Developing interval-based cost-sensitive classifiers by genetic programming for binary high-dimensional unbalanced classification [research frontier],” *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 84–98, 2021.
- [68] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.
- [69] A. Klin, W. Jones, R. Schultz, F. Volkmar, and D. Cohen, “Visual fixation patterns during viewing of naturalistic social situations as predictors of social competence in individuals with autism,” *Archives of General Psychiatry*, vol. 59, no. 9, pp. 809–816, 2002.
- [70] K. A. Pelphrey, N. J. Sasson, J. S. Reznick, G. Paul, B. D. Goldman, and J. Piven, “Visual scanning of faces in autism,” *Journal of Autism and Developmental Disorders*, vol. 32, no. 4, pp. 249–261, 2002.
- [71] G. Dawson, S. J. Webb, and J. McPartland, “Understanding the nature of face processing impairment in autism: insights from behavioral and electrophysiological studies,” *Developmental Neuropsychology*, vol. 27, no. 3, pp. 403–424, 2005.
- [72] D. M. Powers, “Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation,” *arXiv preprint arXiv:2010.16061*, 2020.
- [73] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.