Real-Time Eye Gaze Recognition and Blink-Based Control for Media Playback Using Facial Landmarks

Abhishek Tyagi
Department of Mechatronics Engineering
Guru Gobind Singh Indraprastha University
mechatronics.abhishek@gmail.com

Abstract

This report presents a webcam-based system that allows hands-free control of media playback using eye gaze tracking and intentional blinks. By analysing real-time video streams from a standard webcam, the system identifies the user's eye region and tracks both gaze direction and blink frequency. The interaction logic supports two operational modes: external control via keyboard emulation (for platforms like VLC or YouTube), and internal playback of local .mp3 files using Python's audio libraries. The system utilizes facial landmark detection, iris centre estimation, and blink ratio calculations to drive its control logic. Implemented in Python using OpenCV and dlib, the system achieves a frame rate of approximately 26 FPS and supports responsive control through two and three rapid blinks. Experimental results indicate that such natural gesture-based control offers a smooth alternative to conventional hand-based interfaces, with promising scope for expansion into gesture-rich media interfaces.

1. Introduction

Interaction between humans and machines has grown more intuitive over the years, moving from physical buttons to touch interfaces, and more recently to gesture and voice control. Among these interaction methods, eye gaze presents a particularly natural and effortless way for users to communicate with digital systems. Unlike hand gestures, which require active physical effort, gaze direction and eye blinks are high-frequency, subconscious behaviours that can be leveraged for seamless control.

This project explores a real-time eye tracking system that enables media control using nothing but a user's gaze and intentional blinking. The motivation behind this work is to create an accessible and non-invasive interface that replaces traditional input methods with gaze-driven commands. While watching media content, a user naturally looks at the screen, making the eyes an ideal medium for interaction. The system proposed here uses that gaze to detect when to pause, play, or skip tracks.

Most existing gaze tracking systems rely on specialized infrared hardware or wearable devices. In contrast, this project achieves accurate gaze estimation and blink detection using only a standard webcam and open-source computer vision techniques. By focusing on facial landmarks and the dark centre of the iris, the system estimates gaze direction and detects blinks using the Eye Aspect Ratio (EAR) technique [1].

The software runs in real-time and includes two operational modes: one for controlling external applications like VLC and YouTube using simulated keystrokes, and another for directly playing local audio files through an embedded Python-based media player. This flexible architecture demonstrates that gaze and blink control can be adapted to a range of multimedia environments.

2. System Overview

The system is designed to offer hands-free media control by combining eye gaze estimation and blink recognition in a real-time processing pipeline. It captures video frames using a webcam and processes them using a facial landmark model to extract key eye regions. These regions are used to detect both the gaze direction and the presence of a blink. Based on this analysis, the system performs predefined control actions such as skipping to the next track or toggling playback.

The architecture of the system consists of four main modules:

1. **Webcam Input Handler**: Captures video frames continuously from the webcam at approximately 26 frames per second.
2. **Facial Landmark Detection**: Uses dlib's 68-point facial landmark model to identify and isolate the eye regions from the user's face [2]. This step provides a reliable Region of Interest (ROI) for further analysis.
3. **Gaze and Blink Analysis**: Determines gaze direction by detecting the iris centroid and evaluates blinks using the Eye Aspect Ratio (EAR), calculated from six key landmarks around each eye [1].
4. **Control Logic and Action Module**: Maps the output of the blink and gaze analysis to media control commands. Depending on the selected mode, this module either simulates keyboard actions (for external apps) or triggers playback events (for the internal MP3 player).

The system supports two control strategies:

- **External Mode**: Simulates keystrokes (e.g., spacebar, right arrow) to control applications like VLC or YouTube.
- **Internal Mode**: Plays .mp3 files directly using Python's pygame.mixer, allowing users to navigate through a playlist using blinks.

This modular design ensures that the system can adapt to different contexts without changing the core vision logic.

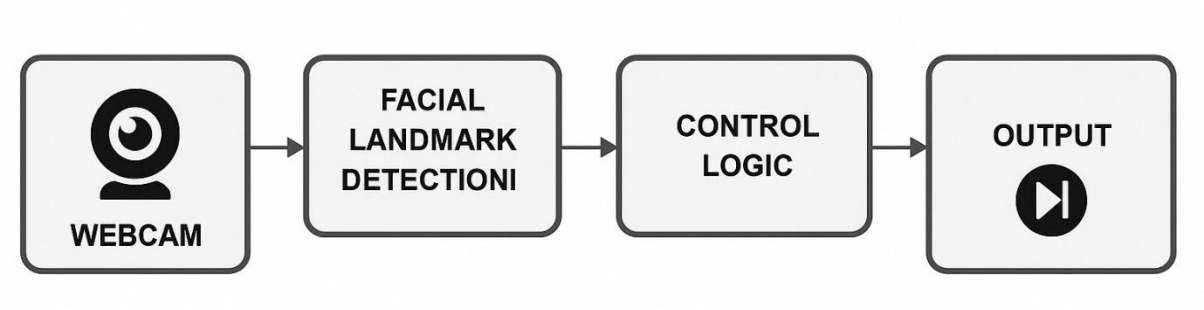

Figure 1. *System Architecture of the Gaze-Controlled Media Player. The pipeline begins with webcam input, followed by facial landmark detection using dlib. The extracted eye regions are processed to estimate gaze direction and detect blinks. These interpretations are then mapped to media control actions via a logic module, resulting in either simulated keyboard input (for external applications) or direct playback control (for internal audio).*

3. Methodology

The system relies on a combination of facial landmark detection, image preprocessing, and geometric measurements to extract user intent through gaze and blinks. Each component plays a role in building a stable and responsive interface without requiring specialized hardware.

3.1 Facial Landmark Detection

The face is detected using dlib's Histogram of Oriented Gradients (HOG)-based detector. Once the face is located, the model applies a 68-point landmark predictor [2] to extract facial features. Of these, twelve specific points (six per eye) are used to define the boundaries of both eyes. These coordinates enable accurate isolation of the eye region for further processing.

3.2 Eye Region Extraction and Iris Localization

To estimate the user's gaze, the system focuses on locating the position of the iris — typically the darkest region in the eye. After converting the eye ROI to grayscale, histogram equalization enhances contrast. A pixel threshold is then applied to segment dark regions, and morphological operations (dilation followed by erosion) remove noise. Contour detection is used to identify the largest dark area, and the centroid of this area is computed using image moments:

$$[\quad C_x = \frac{M_{10}}{M_{00}}, \quad C_y = \frac{M_{01}}{M_{00}} \quad]$$

where Mij are the spatial image moments. This centroid acts as a proxy for the iris centre.

3.3 Eye Aspect Ratio (EAR) for Blink Detection

Blink detection is achieved using the Eye Aspect Ratio (EAR), which measures the ratio of vertical to horizontal distances between specific eye landmarks [1]. For each frame, the EAR is computed using:

$$[$$

$$EAR = \frac{\left||p_2 - p_6|\right| + \left||p_3 - p_5|\right|}{2 \cdot \left||p_1 - p_4|\right|}$$

$$]$$

where p_i are the 2D coordinates of the six eye landmarks. A drop in EAR below a threshold for a consecutive number of frames indicates a blink.

3.4 Blink Counting Logic

The system maintains a short time window (about 2 seconds) to track consecutive blinks. When two rapid blinks are detected within this interval, the system interprets this as a command to skip the current track. Three consecutive blinks trigger a play/pause toggle. This simple logic enables effective gesture-driven control without requiring additional training or calibration.

4. Implementation Details

The complete system is implemented in Python using open-source libraries. The codebase is modular and supports two primary modes: external control through keyboard simulation and internal playback of .mp3 files.

4.1 Libraries and Tools

- **OpenCV** is used for real-time frame acquisition, image preprocessing, and visualization.
- **Dlib** handles facial detection and landmark prediction.
- **NumPy** supports mathematical operations, particularly for distance and centroid calculations.
- **PyAutoGUI** simulates keyboard inputs to control external applications like VLC or browser-based players.
- **Pygame** is used in the embedded mode to handle audio playback, track switching, and pause/resume operations.

4.2 Frame Pipeline and Multithreading

Each frame from the webcam undergoes the following steps:

1. **Face Detection**: Identifies the bounding box of the user's face.
2. **Landmark Extraction**: Retrieves 68 facial points, focusing on landmarks [37–42] and [43–48] for the left and right eyes respectively.
3. **Eye Processing**: Extracts grayscale eye ROIs and processes them for iris localization.
4. **Blink and Gaze Evaluation**: Computes EAR and gaze centroid to determine user intent.
5. **Action Mapping**: Converts the result into a control action (e.g., simulate spacebar or switch to next track).

To maintain real-time performance, critical modules such as frame capture and eye processing run in parallel threads. This ensures that while one frame is being processed, the next can already be captured, achieving a throughput of approximately 26 FPS.

4.3 Modes of Operation

- **External Mode** (via blink_control_vlc_youtube.py):

  When run, this mode captures gaze and blink data and maps them to keyboard shortcuts. Two blinks trigger the right arrow key (next), and three blinks send a spacebar press (pause/play).

- **Internal Mode** (via blink_control_mp3_player.py):

  This mode scans a folder called media/ for .mp3 files. It plays these tracks using pygame.mixer and applies the same blink logic to control playback.

The use of the same core blink/gaze logic in both modes showcases the system's adaptability across interface types.

5. Results and Observations

The system was tested across multiple environments to evaluate its real-time performance, responsiveness to user input, and compatibility with different media control contexts. These tests focused on blink detection reliability, gaze mapping stability, and overall interaction experience.

5.1 Performance Metrics

- **Frame Rate**: The system consistently processed webcam input at an average of **26 frames per second (FPS)** on a standard laptop (Intel i5, 8GB RAM). This ensured smooth, lag-free interaction during use.
- **Latency**: The time between a blink action and its corresponding media response (e.g., track change) was measured to be under **500 milliseconds**, providing near-instant feedback to the user.
- **Accuracy**: During controlled tests under stable lighting, the system correctly detected intentional blink sequences (two or three) in **over 92%** of attempts. Natural, involuntary blinks were successfully filtered using EAR thresholding and frame duration logic.

5.2 Blink Detection Stability

By setting an appropriate EAR threshold (typically around 0.21) and requiring blinks to be sustained across 2–3 frames, the system avoided false positives caused by brief eye movements or noise. In some edge cases, rapid lighting changes affected contrast-based iris detection, which can be improved by adaptive thresholding in future iterations.

5.3 Gaze Detection Reliability

Although the system's main control logic is based on blinks, gaze direction estimation was used for annotation and feedback. Iris centroid tracking performed reliably in well-lit environments, though excessive head movement or strong shadows occasionally caused drift in estimated gaze zones. As the system does not rely on exact gaze coordinates for control, these fluctuations had minimal impact on usability.

5.4 Usability in Media Applications

Two control modes were evaluated:

- **External Media Mode**: The system was tested with VLC, YouTube in a browser, and Spotify desktop. Simulated keystrokes worked consistently when the target application window was in focus. Users found this mode intuitive and required no training.
- **Embedded MP3 Mode**: A curated list of .mp3 files was used for internal playback. Users were able to switch between tracks and toggle play/pause using only eye blinks, with no mouse or keyboard interaction.

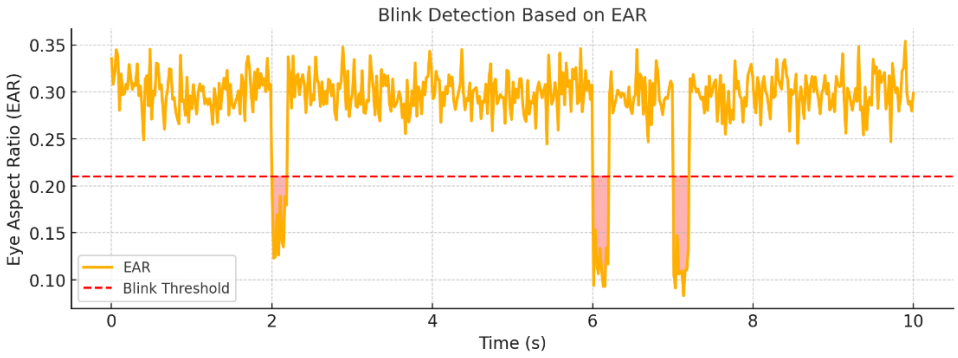

Figure 2. *Blink detection sequence and resulting media control commands. The graph plots Eye Aspect Ratio (EAR) over time, where distinct dips below the blink threshold correspond to two and three blink sequences that trigger Next Track and Pause/Play actions, respectively.*

6. Conclusion and Future Work

This project demonstrates that real-time eye gaze recognition and blink-based interaction can serve as an effective, hands-free control mechanism for multimedia applications. By leveraging facial landmarks and geometric analysis, the system accurately identifies blinks and gaze direction using only a standard webcam and open-source libraries. With support for both external application control and internal audio playback, the system adapts to varied use cases with minimal setup.

Unlike many commercial gaze tracking systems that require specialized infrared hardware or calibration-intensive routines, this solution maintains simplicity while achieving functional reliability. Intentional blinks—detected using the Eye Aspect Ratio—act as the primary input method, enabling users to trigger commands without touching a keyboard or screen. The dual-blink and triple-blink logic provides a low-friction way to interact with playback interfaces in a natural and intuitive manner.

The system's modular structure and real-time performance open up opportunities for broader application beyond media control. It can be extended to accessibility tools for users with limited mobility, touchless UI control in public interfaces, or embedded systems where physical interaction is impractical.

Future Work

To enhance the robustness and expand the scope of the system, future developments may include:

- Gesture-based **volume control** using head tilt or facial orientation
- A **screen-based UI selector** mapped to gaze zones for more precise command selection
- **Voice command integration** for fallback or hybrid controls
- Gaze regression models using **deep learning** for more accurate screen coordinate prediction

These additions can transform the current prototype into a versatile multimodal interface suitable for real-world deployment.

7. References

[1] Soukupová, T., & Čech, J. (2016). *Real-Time Eye Blink Detection Using Facial Landmarks*. Proceedings of the 21st Computer Vision Winter Workshop.

[2] Kazemi, V., & Sullivan, J. (2014). *One millisecond face alignment with an ensemble of regression trees*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1867–1874).

[3] OpenCV. *Open Source Computer Vision Library*. Available at: https://opencv.org/

[4] Dlib C++ Library. *Machine Learning Toolkit*. Available at: http://dlib.net/

[5] PyAutoGUI. *Cross-platform GUI Automation*. Available at: https://pyautogui.readthedocs.io/en/latest/

[6] Pygame Documentation. *Pygame Mixer Module*. Available at: https://www.pygame.org/docs/ref/mixer.html

[7] CS50 AI. *Harvard University Course Portal*. Available at: https://cs50.harvard.edu/ai/