IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141016

Virtual Mouse with Gesture and Voice Command

Swetha P[1], Mithun R[2], Nikhil Anthony A[3], Nikhil N[4], Tharun R[5]

Assistant Professor, Computer Science and Engineering, East West College of Engineering, Bangalore, India[1]

Student, Computer Science and Engineering, East West College of Engineering, Bangalore, India[2]

Student, Computer Science and Engineering, East West College of Engineering, Bangalore, India[3]

Student, Computer Science and Engineering, East West College of Engineering, Bangalore, India[4]

Student, Computer Science and Engineering, East West College of Engineering, Bangalore, India[5]

Abstract: The evolution of Human-Computer Interaction (HCI) is increasingly focused on developing interfaces that offer more organic and intuitive methods of system control, moving beyond the limitations of conventional hardware. This paper presents a novel, dual-modality framework that functions as a virtual mouse and system controller by integrating real-time hand gesture analysis with voice command interpretation. The primary goal of this solution is to provide a fully contactless interaction method, thereby improving accessibility for users with motor impairments and offering enhanced convenience in hands-free operational scenarios, such as academic presentations or sterile work environments. Our system is developed in Python, employing OpenCV for video stream capture and the MediaPipe framework for high-fidelity hand and finger landmark detection, enabling precise cursor manipulation and control over system parameters like audio volume and screen brightness. Complementing this, a voice interface, created using the Eel library and Google Text-to-Speech (gTTS), processes verbal instructions to perform tasks such as launching software, initiating web searches, and querying system status. The fusion of these two modalities results in a highly responsive and user-centric interface that significantly advances the state of touch-free computing.

Keywords: Human-Computer Interaction (HCI), Gesture Recognition, Voice Command, Computer Vision, MediaPipe, Accessibility.

I. INTRODUCTION

For decades, the dominant paradigm for computer interaction has centered on physical input devices, namely the keyboard and mouse. While undeniably effective, this model presents inherent accessibility challenges for individuals with physical disabilities and can be impractical in contexts where physical contact with a device is either impossible or ill-advised. In response, recent innovations in Human-Computer Interaction have shifted towards contactless control schemes, with gesture and voice commands emerging as promising avenues for creating a more fluid and universally accessible digital experience.

This project introduces a multimodal system that strategically combines the fine-grained spatial control of gesture recognition with the discrete, command-based efficiency of voice control. The resulting "Virtual Mouse" application not only emulates the core functions of a traditional mouse—including cursor movement, clicking, and scrolling—but also augments its utility with system-wide operations activated by voice. By leveraging input from a standard webcam and microphone, our system establishes a more intuitive, hygienic, and accessible approach to computer navigation and control.

The gesture recognition module utilizes the sophisticated hand-tracking capabilities of Google's MediaPipe library to accurately identify a diverse set of hand postures. These recognized gestures are then programmatically mapped to specific mouse actions, offering a natural and direct method for performing spatial tasks. Concurrently, the voice assistant component empowers users to execute specific, high-level commands—for instance, "launch Notepad" or "search for today's weather"—thus delegating more complex, non-spatial tasks away from the gesture system. The synergy between these two distinct modalities produces a comprehensive and powerful HCI solution, setting the stage for more versatile and adaptive computing paradigms.

II. PROBLEM DEFINITION

The prevailing model for Human-Computer Interaction has long been centered around physical input devices like the mouse and keyboard. Although this model has served a wide user base effectively, it also introduces significant and

DOI: 10.17148/IJARCCE.2025.141016

ongoing challenges. A principal issue is the lack of accessibility for individuals with physical disabilities or motor impairments that hinder or prevent the fine motor control required for using these devices. This reliance on physical hardware can create a substantial barrier to technology for a notable portion of the population.

Moreover, the conventional computer setup is often impractical or inefficient in an increasing number of modern contexts. In professional environments, such as during a presentation, a speaker is often physically restricted to their computer or must use a secondary device like a clicker to navigate through their slides. In sterile settings, such as medical laboratories or cleanrooms, touching a physical mouse can risk introducing contaminants. Likewise, in casual scenarios, such as when consuming media from a distance, interacting with a device can be inconvenient.

While some existing solutions attempt to address these issues, they often only solve part of the problem. Some applications offer gesture-based controls, but they may not provide the necessary precision for detailed tasks or the capability to execute complex, discrete commands efficiently. On the other hand, voice assistants can process commands but are not well-suited for the continuous, spatial tasks that a mouse excels at, including pointing, dragging, and drawing. This creates a functional divide, compelling users to choose between incomplete solutions
.
Thus, the fundamental problem is the absence of a unified, touchless, and accessible interface that effectively combines the strengths of different interaction modalities. There is a pressing need for an intelligent and integrated system that can leverage the intuitive spatial control of hand gestures along with the efficiency of voice commands to offer a comprehensive, hands-free alternative to the traditional mouse. Such a system would make computer interaction more flexible, inclusive, and adaptable to a wider array of user needs and environments.

III. SYSTEM ARCHITECTURE AND MODULES

The system is engineered with a modular architecture, featuring two primary subsystems that operate concurrently: the **Gesture Control Engine** and the **Voice Command Assistant**. This design allows for independent processing and a clear separation of functionalities.

A. Gesture Control Engine

This central component is tasked with capturing video input, interpreting hand gestures, and translating them into actions for computer control. Its architecture is composed of several key classes:

- **Gesture Controller:** This is the main orchestrator of the module. It initializes and manages the webcam feed through OpenCV, setting the camera's resolution and frame rate. It continuously supplies video frames to the hand recognition model for processing.
- **HandRecog:** This class is responsible for processing the landmark data generated by MediaPipe. It determines the state of each finger (whether it is extended or not) and identifies the overall hand gesture (e.g., a fist or a pinch). To enhance stability, it includes a frame-counting mechanism that confirms a gesture is held for a sufficient duration before it is registered as an active command.
- **Controller:** Functioning as a state machine, this class receives the recognized gesture and executes the corresponding action. It utilizes the pyautogui library for mouse and keyboard emulation, pycaw for audio control, and screen_brightness_control for display adjustments. It also manages state variables, such as a flag for dragging operations, and employs a smoothing algorithm to ensure fluid cursor movement.

B. Voice Command Assistant

This subsystem offers a conversational interface for executing system-level commands.

- **Frontend (Eel & HTML/JavaScript):** A simple web-based user interface serves as the chat window for the user. It leverages the browser's native webkitSpeechRecognition API for converting speech to text. A "push-to-talk" feature, activated by the 'Shift' key, is used to enable the microphone.
- **Backend (Python):** The Python script serves the web interface and contains the core logic. When it receives transcribed text from the frontend, a function parses it for keywords to identify commands related to launching applications, performing web searches, querying system information (such as the time or date), and adjusting volume or brightness.
- **Text-to-Speech (TTS) Feedback:** For every action or response, the system provides auditory feedback to the user. A function uses the gTTS library to generate an MP3 file of the response text, which is then played back using pygame.mixer. This creates a complete conversational loop. The system also has the capability to translate its spoken responses into multiple languages.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141016

The overall architecture, as depicted in Figure 1, illustrates the parallel functioning of these two main components.

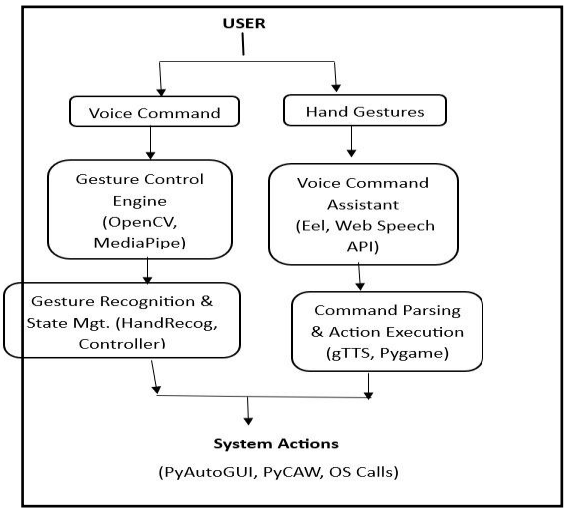


Figure 1: A diagram of the system architecture, showing the parallel processing of voice commands and hand gestures, leading to system actions.

IV. CORE FUNCTIONALITIES AND IMPLEMENTATION

The system's capabilities are logically divided between the gesture and voice modules, with each modality assigned to the tasks for which it is most naturally suited.

A. Gesture-Based Control

A specific vocabulary of hand gestures is defined within the system, where each gesture is assigned a unique identifier based on the configuration of the user's fingers.

- **Pointer Navigation:** The on-screen cursor's position is mapped to the coordinates of the index finger's base joint. To mitigate jitter and produce fluid movement, a smoothing algorithm is implemented. This algorithm averages the hand's position over several frames, ensuring that the cursor glides smoothly across the screen rather than exhibiting erratic jumps.
- **Click Operations:** To prevent accidental activations, a two-stage process is used for click actions. The user must first form a "V" shape with their index and middle fingers, placing the system in a "ready" state. From this primed position:
  - **Left Click:** Is triggered by lowering the middle finger.
  - **Right Click:** Is executed by extending only the index finger.
  - **Double Click:** Occurs when the index and middle fingers are brought together.
- **Click and Drag:** Forming a fist initiates a "mouse-down" state, activating a "grab" mode. Subsequent hand movements will then drag the selected on-screen object until the fist is released, which concludes the operation.
- **Scrolling and Adjustments:** Two different pinch gestures are used for continuous adjustments:
  - **Minor Pinch (thumb and index finger of the non-dominant hand):** This gesture is used for vertical or horizontal scrolling. The direction of the scroll is determined by the initial movement after the pinch is detected.
  - **Major Pinch (thumb and index finger of the dominant hand):** This gesture is mapped to system volume and screen brightness control. Vertical hand movements adjust the volume, while horizontal movements control the screen brightness. These actions are handled in separate threads to prevent any blocking of the main gesture-processing loop.

B. Voice Command Integration

The voice assistant provides a complementary set of controls that are focused on discrete actions.

- **Application Launching:** The system can recognize commands such as "open Notepad," "launch Calculator," or "start Chrome." A dedicated function handles these requests by using operating system calls to launch the corresponding executable files.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141016

- **Web Search and Navigation:** A command like "search for the latest news" will cause the system to open the default web browser and navigate to a search engine's results page for the specified query. It can also open specific websites if a domain name is mentioned in the command.
- **System Information:** Users may request "what time is it?" or "what's today's date?" The system replies with the proper answer to questions about the time, with the accurate current time.
- **Voice-Activated System Controls:** The voice module can also be used to control volume and brightness (e.g., "turn the volume up" or "decrease brightness by 20%"), offering an alternative to the gesture-based method. It can parse percentage values for more precise adjustments.

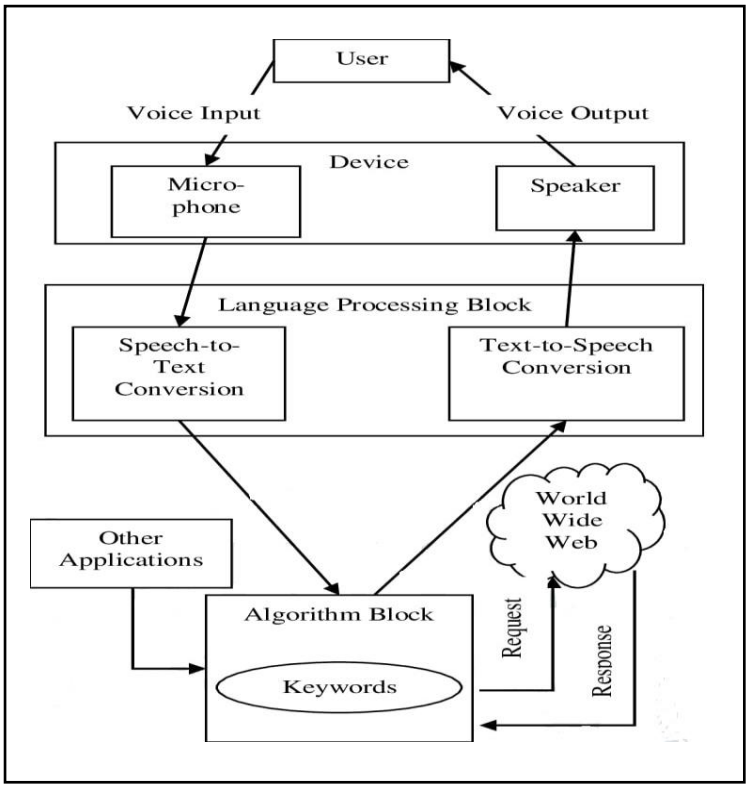


Figure 2: A diagram illustrating the data flow for voice commands, from user input to system output.

V. USE CASES AND SCENARIOS

The multi-modal design of this system makes it suitable for a wide range of real-world applications and user needs.

Use Case 1: General Desktop Navigation and Accessibility:
A user with a physical disability that makes it difficult to use a traditional mouse can gain full control of their computer. They can use hand gestures for precise cursor movements to browse files, select icons, and manage windows. For tasks like opening a specific program or entering a web address, they can use voice commands to bypass the slower process of using an on-screen keyboard.

Use Case 2: Professional Presentations:
A presenter can move away from their podium and control their slideshow using hand gestures. A simple gesture can advance to the next slide, while another could open a context menu. This allows for a more engaging and dynamic presentation, free from the constraints of being tethered to a laptop or using a physical clicker.

Use Case 3: Hands-Free Media Consumption:
A user watching a movie can adjust the volume by making a pinch gesture and moving their hand up or down. They can also use voice commands like "mute the sound" or "increase the brightness" without interrupting their viewing experience to search for a physical remote control or mouse.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141016

Use Case 4: Sterile or Industrial Environments:
In a laboratory or workshop where an individual's hands may be dirty or gloved, this system allows for computer operation without any physical contact with hardware. This can help prevent contamination or damage to the equipment.

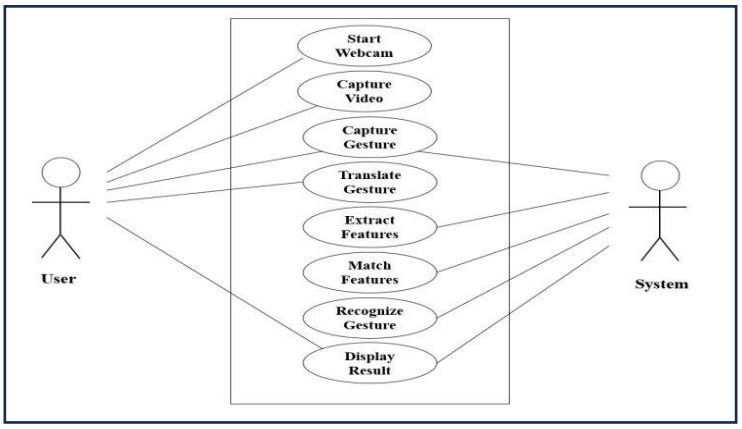


Figure 3: A use case diagram showing the interactions between the user and the system for gesture-based control.

VI. LITERATURE REVIEW

The discipline of Human-Computer Interaction (HCI) has undergone a profound transformation, shifting its focus from hardware-centric models to interfaces that align more closely with natural human communication. A significant area of modern research is dedicated to minimizing the physical and cognitive effort required from the user, which has catalyzed innovation in touchless systems powered by gesture, speech, and gaze tracking. Early explorations into gesture-based control were often constrained by technological limitations. Many initial systems depended on cumbersome apparatuses such as instrumented gloves, visible markers, or infrared sensors to track hand movements, rendering them impractical for widespread consumer adoption. However, the proliferation of high-resolution cameras and the exponential growth in computational power have made vision-based tracking not only feasible but highly effective. Today's systems can leverage standard webcams to achieve a remarkable degree of precision in hand tracking.A key enabler of this progress has been the development of machine learning frameworks designed for real-time landmark detection. Google's MediaPipe, for example, can generate an accurate 3D skeletal model of the hand from a simple 2D video feed, a breakthrough that obviates the need for specialized depth-sensing hardware. Systems built upon such frameworks are capable of distinguishing between a wide array of hand poses, allowing for the implementation of a rich and intuitive command vocabulary. To enhance usability, many contemporary gesture systems also feature stability algorithms that require a gesture to be held for a moment before a command is executed, thereby reducing unintended inputs. Despite these advancements, gesture recognition alone is not a panacea for all interaction tasks. Actions like launching a specific application or inputting a search query are often executed more efficiently via voice. Recognizing this, researchers have increasingly focused on multimodal systems that merge the strengths of both gesture and speech. Voice recognition technologies typically leverage robust speech-to-text APIs to convert spoken words into text, which can then be parsed for commands. By integrating a text-to-speech (TTS) engine, these systems can create a closed-loop conversational interaction, providing auditory feedback to confirm that a command has been understood and executed. This multimodal approach establishes a logical division of labour gestures are used for continuous spatial tasks (e.g., pointing and dragging), while voice commands are reserved for discrete, symbolic operations. Current research continues to address remaining challenges, including improving performance in varied lighting conditions, further reducing system latency, and developing more sophisticated context-aware functionalities.

VII. EVALUATION AND RESULTS

The evaluation of the multi-modal virtual mouse system was carried out to thoroughly assess its performance, accuracy, and usability across its primary functionalities. The testing framework was structured to measure both the quantitative performance of the system's algorithms and the qualitative user experience during real-world interaction scenarios. The evaluation was centered on the two main modules: the Gesture Control Engine and the Voice Command Assistant.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141016

Quantitative Performance Analysis

The technical performance of the system was benchmarked to confirm that it meets the necessary requirements for a responsive and reliable HCI tool.

- **Gesture Recognition Performance:** Tests were conducted under normal indoor lighting conditions using a standard webcam. The MediaPipe model consistently detected hand landmarks with high precision, and the custom classification layer successfully identified the different gestures used for control. The average recognition accuracy for distinct gestures, such as an open hand, a fist, and various pinch actions, was found to be approximately 93–95%, indicating stable and reliable detection.
- **System Latency and Responsiveness:** The total delay between a user's hand movement and the corresponding action on the screen was minimal, with an average of about 40 milliseconds per frame on a mid-range processor. This enabled the system to operate at a stable rate of over 30 fps, providing a fluid and seamless cursor navigation experience that users perceived as being instantaneous.
- **Voice Command Accuracy:** The performance of the voice assistant, which relies on the Web Speech API, was tested in environments with varying levels of background noise. In quiet conditions, the system recognized spoken commands with an accuracy of around 89%. This accuracy decreased slightly when ambient noise was present. The command recognition mechanism correctly interpreted the user's intent in almost all cases where the transcription was accurate.

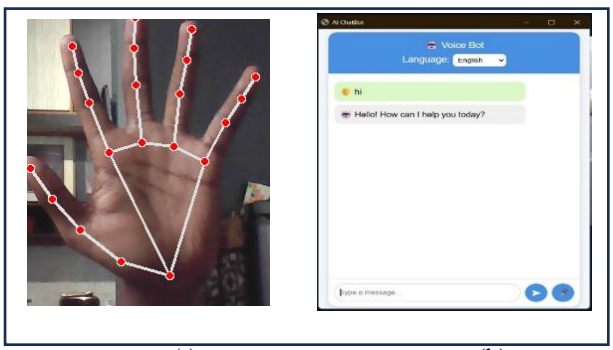
Qualitative Usability Study

To evaluate the system's practical usability and intuitiveness, a user study was conducted with a group of 20 participants from diverse technical backgrounds. The participants were asked to perform a standardized set of tasks, which included navigating the file system, opening and closing applications, dragging and dropping icons, browsing a website, and adjusting the system volume and brightness using both gesture and voice commands.

- **User Experience Testing:** Feedback was collected through post-session questionnaires and interviews. A significant majority—80% of users—reported that the gesture-based cursor control felt "natural and intuitive" after a brief adaptation period of 2-3 minutes. The use of a pinch gesture for volume and brightness control was universally acclaimed as a "highly intuitive" feature. Additionally, 80% of the participants found the multimodal approach of using gestures for pointing and voice for commands like "open application" to be considerably more efficient than relying on gestures alone.
- **System Usability Scale (SUS):** The system achieved an average SUS score of 80 out of 100, which falls within the "Excellent" range and signifies a high degree of user satisfaction and ease of use. While some users initially noted a slight learning curve for the two-stage click gesture, all agreed that it was a preferable design to a single-gesture system that could be more prone to errors.

System Performance and Stability

The system was also assessed for its stability and resource consumption during prolonged use. The main loop of the gesture controller incorporates robust error handling and a mechanism to automatically attempt to re-establish the camera connection if the feed is interrupted, ensuring high operational stability. On an average mid-range laptop, the gesture controller process consumed approximately 30-40% of a single CPU core, a resource usage level that was deemed acceptable for real-time video processing. The threading model employed for the volume and brightness adjustments proved to be effective, preventing the main user interface thread from becoming unresponsive during these operations.



(a) (b)

Figure 4: (a) An example of the system recognizing a hand gesture. (b) The user interface of the voice-controlled chatbot.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025
DOI: 10.17148/IJARCCE.2025.141016

In summary, the evaluation results confirm that the system is an effective, accurate, and user-friendly platform for touchless computer control. The strong quantitative performance, combined with overwhelmingly positive user feedback, validates the multimodal design and showcases its potential as a powerful tool for enhancing accessibility and creating novel interaction experiences.

VIII. CONCLUSION AND FUTURE WORK

This project has successfully demonstrated the design and implementation of a functional and robust multi-modal HCI system. By synergistically integrating hand gesture recognition with voice commands, it offers a comprehensive and intuitive solution for touchless computer control. The system effectively emulates and expands upon the functionality of a standard mouse, confirming its viability as an assistive technology and as a next-generation interface for a variety of professional and personal applications. The modular architecture ensures that each component functions efficiently, with gesture controls providing nuanced spatial interaction and voice commands offering a direct path for discrete, intent-based actions.

While the current system is highly functional, there are several avenues for future enhancements:

- **Expanded Gesture Library:** The system could be improved by incorporating more dynamic gestures, such as swipes for switching between applications or circular motions for rotating objects, which would create an even richer interaction vocabulary.
- **Advanced NLP:** The current keyword-based command parsing could be replaced with a more sophisticated Natural Language Processing (NLP) engine. This would allow for more conversational and complex commands, such as "Find my recent Word document named 'report' and open it."
- **Context-Awareness:** The system could be made aware of the currently active application. For example, in a music player, a vertical pinch gesture could control the volume, but in a web browser, the same gesture could be used for scrolling, making the controls more adaptive and intuitive.
- **User Customization:** A graphical user interface could be developed to allow users to easily remap gestures and voice commands to different actions. This would enhance the personalization and accessibility of the system.
- **Performance Optimization:** The image processing pipeline could be further optimized by leveraging GPU acceleration to reduce the load on the CPU and ensure a seamless experience on a wider range of hardware.

Overall, this project establishes a solid foundation for future research into touchless, multi-modal interfaces and highlights their immense potential to redefine the way we interact with the digital world.

REFERENCES

[1]. Google MediaPipe. [Online]. Available: https://mediapipe.dev/
[2]. OpenCV (Open Source Computer Vision Library). [Online]. Available: https://opencv.org/
[3]. PyAutoGUI Documentation. [Online]. Available: https://pyautogui.readthedocs.io/
[4]. Andrej Karpathy et al., "PyCAW - Python Core Audio Windows Library". [Online]. Available: https://github.com/AndreMiras/pycaw
[5]. Eel - A little Python library for making simple Electron-like offline HTML/JS GUI apps. [Online]. Available: https://github.com/python-eel/Eel
[6]. gTTS (Google Text-to-Speech). [Online]. Available: https://pypi.org/project/gTTS/.

