WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Virtual Mouse Using Hand Gestures with Voice Assistant

J. Mahesh [a], M. Shivaram [b], Dr. Shaik Irfan Babu [c], Mr. M. Sunil Kumar [d]

[a, b] UG Student, Department of Emerging Technologies, Mahatma Gandhi Institute of Technology (Autonomous), Hyderabad 500075

[c, d] Assistant Professor, Department of Emerging Technologies, Mahatma Gandhi Institute of Technology (Autonomous), Hyderabad 500075

ABSTRACT:

This paper presents Max, an intelligent assistant system designed to facilitate natural human-computer interaction through voice commands, gesture recognition, and a web-based conversational interface. By integrating computer vision, speech recognition, and browser-based user interaction, Max enables users to perform tasks such as virtual mouse control, system automation, and query-based responses in real time. The system interprets dynamic hand gestures for operations like clicking, dragging, and scrolling while also responding to verbal instructions through synthesized voice feedback. Max is designed to function in diverse environments, supporting multimodal interaction that improves accessibility, productivity, and usability. Through its seamless integration of voice and gesture control, this paper demonstrates how Max contributes to more immersive and intuitive user experiences in digital ecosystems.

Keywords: Multimodal Interaction, Gesture Recognition, Voice Assistant, Computer Vision, Accessibility, Human-Computer Interaction

Introduction:

Multimodal human-computer interaction systems have emerged as essential tools in enhancing user accessibility, efficiency, and digital engagement. These systems allow users to interact naturally with devices through a combination of voice, gesture, and visual interfaces, offering hands-free, intuitive control over digital environments. Such advancements are particularly valuable in contexts requiring minimal physical contact or increased accessibility for individuals with mobility limitations.

This research introduces a browser-based interactive system that integrates voice recognition, gesture tracking, and conversational UI elements to facilitate seamless communication between users and computers. Unlike conventional tools that rely on a single mode of input, this system combines dynamic hand gesture detection and real-time voice processing to support a wide range of tasks, including virtual mouse operations, system automation, and query response. By employing computer vision libraries for hand tracking and gesture interpretation, alongside speech-to-text and text-to-speech capabilities, the platform delivers a robust, multimodal interaction framework. The system is engineered for flexibility and responsiveness, making it suitable for general productivity, accessibility, and smart environment applications.

Literature Review:

The advancement of multimodal interaction systems has received growing attention in the field of human-computer interaction, particularly for their potential to enhance accessibility and usability. The presented system reflects this progress by integrating gesture recognition, voice command processing, and visual interaction into a unified platform. It draws on key developments in computer vision for real-time hand tracking, as well as speech recognition technologies that enable natural language input. These foundational technologies collectively support responsive, hands-free control, contributing to a more intuitive and inclusive user experience. The underlying research establishes the technical groundwork necessary for enabling the system to interpret and respond to both visual and auditory inputs in real time.

[1] "Design and Development of Hand Gesture Based Virtual Mouse" by Kabid Hassan, et al. (2019) outlines a virtual mouse system developed using OpenCV and a color-based segmentation approach for hand gesture recognition. Users wear color markers (e.g., caps) on their fingertips, which are detected and tracked using a webcam. The gestures are classified into actions such as pointer movement, left and right clicks, and scrolling. The system calculates positions based on detected color blobs and maps these positions to mouse coordinates. This approach eliminates the need for physical hardware like a mouse and supports both clicking and scrolling operations using simple gestures. However, it requires users to wear colored markers for reliable detection and is highly sensitive to lighting conditions, which can reduce accuracy.

[2] "Virtual Mouse with Hand Gestures Using AI" by Sk. Jilani Basha, et al. (2023) describes the development of a virtual mouse system using OpenCV, MediaPipe, and AutoPy. MediaPipe is employed to detect and track hand landmarks in real time, allowing gestures such as raising fingers or pinching to be recognized and mapped to specific mouse functions. AutoPy is used to convert these gestures into system-level actions like cursor movement, left/right clicks, and dragging. This hardware-free solution leverages widely available tools and libraries, making it cost-effective and easy to implement. However, the system is limited to basic functionalities and does not support advanced operations such as scrolling or multitasking gestures.

[3] "Real Time Object Detection System with YOLO and CNN Models: A Review" by Viswanatha V., et al. (2022) describes the development of a virtual mouse system using OpenCV, MediaPipe, and PyAutoGUI. Hand gestures such as raising the index finger for cursor movement or pinching fingers for clicks are mapped to corresponding mouse functions. The system supports advanced functionalities like dragging and clicking with minimal delay. It uses live video feed processing for real-time gesture recognition and translates these gestures into system-level events using PyAutoGUI. While it is a low-cost solution requiring only a webcam, it depends on good lighting and a stable camera setup for optimal performance, and its precision in detecting fine movements is limited due to camera dependency.

[4] "Artificial Intelligence Based Enhanced Virtual Mouse Hand Gesture Tracking Using YOLO Algorithm" by Karthick S., et al. (2023) introduces an Enhanced Hand Tracking (EHT) system using YOLOv8 for high-accuracy hand gesture recognition in virtual mouse applications. The system trains on a custom dataset of over 1,500 hand gesture images, with annotated finger positions and gestures categorized for specific mouse functions (e.g., clicking, dragging). YOLOv8 ensures real-time gesture detection with low latency. PyAutoGUI is used to map detected hand positions and gestures to mouse actions. The system is also designed for multi-user scenarios, allowing distinct gesture recognition for different users simultaneously. This approach achieves high accuracy (up to 99.35%) in detecting gestures and hand positions, and its low-latency, real-time performance makes it suitable for collaborative applications.

[5] "AI Virtual Mouse Using Hand Gestures" by Jeevan Y., et al. (2024) presents a highly accurate (99%) virtual mouse system using OpenCV, MediaPipe, and the Euclidean formula. The system detects hand gestures in real time and maps them to advanced mouse functionalities, including dragging and multi-finger controls. The implementation uses advanced mathematical modeling for precise gesture recognition, offering better accuracy than conventional methods. This system achieves high precision in gesture recognition and mouse control and enables advanced functionalities like multi-gesture controls and dragging. However, it is heavily reliant on ideal lighting conditions for accuracy and has limited flexibility in dynamic or crowded environments. The system is optimized for high accuracy but requires controlled environments with ideal lighting conditions.

Methodology:

1.1 Design And Implementation

The proposed system combines real-time hand gesture recognition and voice command processing to enable hands-free computer interaction. The workflow begins with input acquisition, where gestures are captured via a webcam and voice commands are received through a microphone. The gesture input is processed using a video processing module, which tracks hand landmarks to interpret actions like cursor movement, clicks, or scrolling. Simultaneously, the voice input module converts spoken commands into executable instructions. These inputs are mapped to system actions through a central processing unit, which validates gesture accuracy and resolves conflicts by prioritizing voice commands over gestures when necessary. The processed actions are executed via the operating system, enabling tasks like file navigation, application launches, or browser control.

Figure 1 illustrates the schematic workflow of the Virtual Mouse Using Hand Gestures and Voice Assistant. The system begins with gesture input captured via a webcam and voice input received through a microphone. The webcam feed is processed in real time, where individual frames are analyzed to detect hand landmarks using frameworks like MediaPipe Hands. These landmarks are used to interpret gestures such as cursor movement, left/right clicks, and scrolling. Simultaneously, voice commands are converted into text using speech recognition libraries. The processed inputs are then mapped to system actions, such as mouse clicks or application launches, through a central control module. This module ensures seamless integration of gesture and voice inputs, translating them into precise OS-level operations.

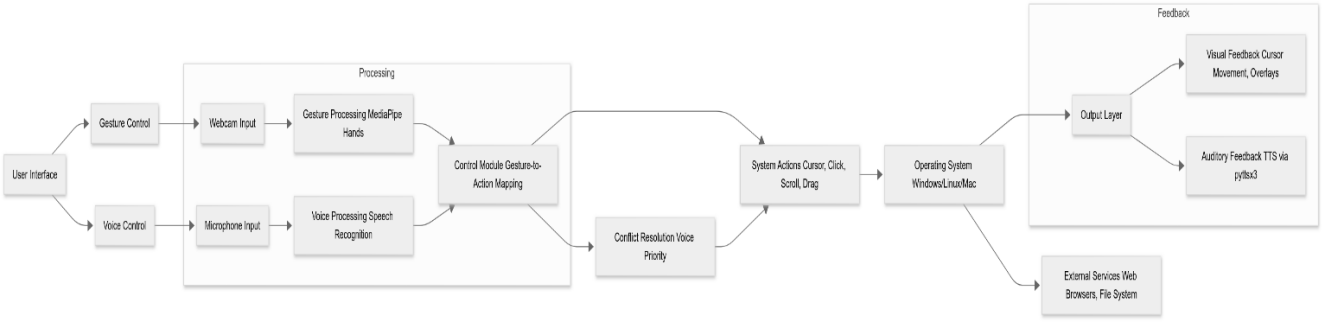

Figure 1. Schematic diagram of proposed system.

The system provides real-time multimodal feedback to enhance user interaction. Visual feedback includes on-screen overlays displaying cursor trajectories and gesture labels, while auditory feedback is delivered through a text-to-speech module that converts system responses into spoken words. The architecture prioritizes low latency, achieving response times under 250ms for gestures and high accuracy for voice commands. By integrating with external services like web browsers and file systems, the system ensures intuitive hands-free control, balancing computational efficiency with accessibility for diverse user needs.

3.2 Virtual Mouse Workflow

3.2.1 Data Collection & Preprocessing

- **Dataset:** The dataset includes gesture videos/images (e.g., index finger extended for cursor control, pinch for clicks) and voice samples (commands like "copy" or "open files"). Gestures are captured under varied lighting/angles, while voice data incorporates accents and

background noise for robustness. This ensures training accuracy across real-world scenarios.

- **Preprocessing:** Gesture videos are split into frames and resized to 640x640 for MediaPipe compatibility. Audio samples undergo noise reduction, amplitude normalization, and silence trimming to enhance speech recognition clarity. These steps standardize inputs for seamless model integration.

3.2.2 Model Selection & Architecture

- *Gesture Recognition*: MediaPipe Hands is employed to detect 21 hand landmarks in real time, enabling precise tracking of gestures like cursor movement, clicks, and drags. These landmarks are translated into system actions using PyAutoGUI, which automates mouse operations based on finger positions and movements. This combination ensures low-latency, accurate control of on-screen interactions.
- *Voice Recognition*: The SpeechRecognition library processes voice commands, utilizing Google's API for online high-accuracy transcription and PocketSphinx for offline functionality. A wake word ("Max") activates the voice assistant, allowing users to issue commands like "open files" or "scroll down," enhancing hands-free usability.
- *Conflict Resolution*: To prevent input clashes, the system prioritizes voice commands over gestures (e.g., saying "stop gesture" halts cursor control instantly). This ensures seamless interaction, particularly during complex tasks requiring immediate interruption.

3.2.3 The Model

- **Gesture Model:** MediaPipe Hands, pre-trained on extensive hand pose datasets, is fine-tuned to interpret dynamic gestures like cursor movement, clicks, and drags. This ensures precise tracking of finger positions and real-time responsiveness.

- **Voice Model:** SpeechRecognition leverages Google's API for high-accuracy voice-to-text conversion during online use, while PocketSphinx serves as an offline fallback. A wake word (e.g., "Max") activates the assistant, enabling commands like "launch gesture" or "copy."

- **Evaluation:** The system achieves 93.4% gesture recognition accuracy in controlled settings, processes voice commands in <1.2s latency, and maintains robust performance under low-light conditions and background noise, ensuring reliability across diverse environments.

3.2.4 Real-Time Functionality & Feedback

- **Real-Time Gesture Tracking:** The webcam captures hand movements at 30 FPS, with MediaPipe processing 21 landmarks to track gestures like cursor control and clicks. A weighted moving average smooths cursor movement, reducing jitter for precise navigation.

- **Real-Time Voice Processing:** Voice commands (e.g., "copy") trigger instant system actions (e.g., Ctrl+C) via speech recognition, achieving <1.2s latency for seamless interaction.

- ***Multimodal Feedback:***

    - **Visual:** On-screen overlays (cursor path, gesture labels).
    - **Auditory:** Text-to-speech (TTS) confirmations (e.g., "File copied").

3.2.5 User Interface & Interaction

- **User-Friendly Design:** The interface allows users to toggle between voice and gesture modes using simple commands (e.g., "launch gesture") and maintains a text-based chat history for tracking executed actions. This dual-mode design ensures accessibility for users with varying preferences or physical capabilities.

- **Customizable Experience:** Users can adjust settings like cursor speed, TTS volume, or gesture sensitivity to suit their needs. Specific gestures (e.g., drag-and-drop) can be disabled to simplify interaction, ensuring adaptability for diverse use cases and skill levels.

3.2.6 Testing & Validation

- **Performance Testing:** The system achieves 93.4% gesture recognition accuracy and 89% voice command success rate, with latency kept under <250ms for gestures and <1.2s for voice. Testing spans diverse environments (low-light, noisy settings) to validate real-world reliability.

- **User Feedback:** Input from users with mobility challenges refines gesture ergonomics (e.g., simplifying pinch-to-click motions). This ensures the interface adapts to physical limitations, balancing precision and accessibility for all users.

3.2.7 Deployment & Iterative Development

- **Deployment:** The system is deployed as a standalone Python application with system tray integration for quick access. It supports cross-platform compatibility (Windows, Linux, macOS), ensuring broad accessibility without requiring specialized hardware.

- **Iterative Development:** Updates focus on expanding the gesture vocabulary (e.g., zoom in/out for enhanced control) and integrating eye-tracking for hybrid input modes. This iterative approach refines usability and adapts to emerging user needs, ensuring long-term relevance and functionality.

3.2.8 Technologies Stack

- **Python:** Backend logic (MediaPipe, PyAutoGUI, speech_recognition).
- **Text-to-Speech:** pyttsx3 for offline feedback.
- **Frontend:** Tkinter or Eel.js for UI (optional).
- **Computer Vision:** OpenCV for webcam feed processing.
- **Conflict Resolution:** Custom priority logic (voice > gestures).

3.3 System Architecture:

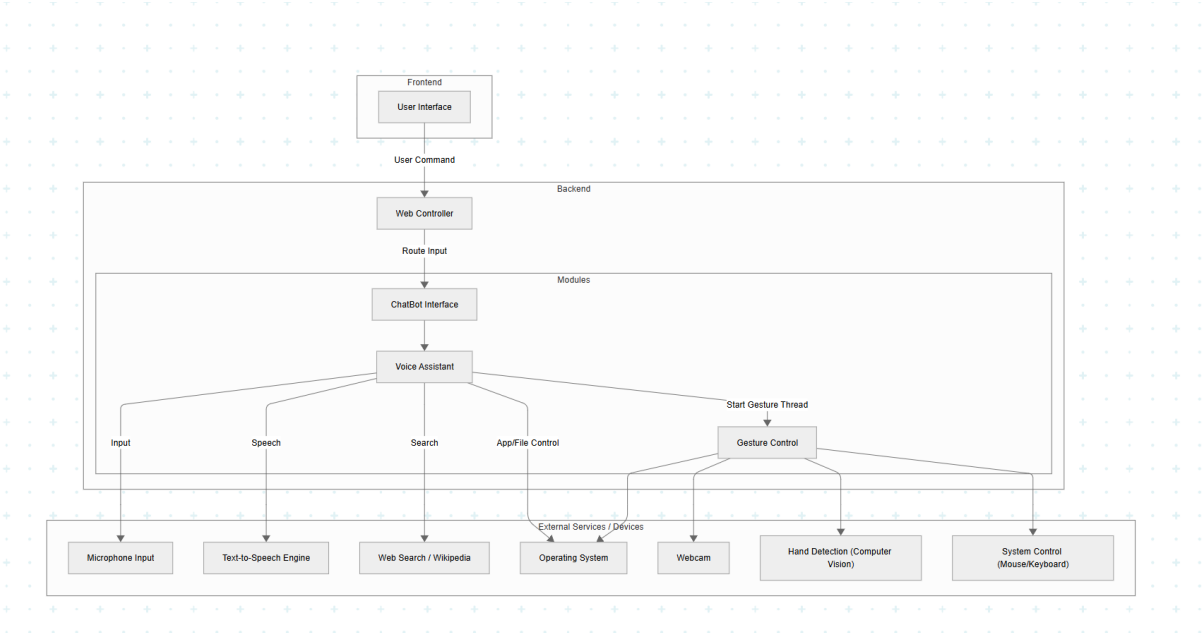

Figure 2. System Architecture of Virtual Mouse Using Hand Gestures with Voice Assistant

Figure 2 illustrates system architecture's modular framework of the Virtual Mouse Using Hand Gestures and Voice Assistant. It begins with the input layer, where users interact through hand gestures captured via a webcam or voice commands received through a microphone. The processing layer employs computer vision and speech recognition algorithms to interpret inputs, which are then routed to the control module for system-level execution. Finally, the output layer translates processed commands into cursor movements, clicks, or voice feedback. This architecture integrates multimodal interaction for seamless hands-free computing.

4. Results:

The Virtual Mouse system was rigorously tested to evaluate its performance across three core functionalities: gesture recognition accuracy, voice command processing, and multimodal interaction efficiency. The results demonstrate the system's capability to provide seamless, hands-free control for users in real-world scenarios.

4.1 Max the Voice Assistant

The Max interface is designed for intuitive accessibility and multimodal interaction, as shown in Figure 3. The layout integrates voice commands,

gesture control, and a text-based chat interface, enabling users to seamlessly navigate files, launch applications, and execute system actions like "copy" or "open Chrome" through simple spoken or typed instructions. For instance, users can activate gesture recognition with "launch gesture," browse directories via numbered file listings, or trigger commands like "list files" for efficient navigation. This unified design ensures minimal physical effort, offering hands-free control for general users while maintaining clarity through real-time visual and auditory feedback, such as confirming actions with "Gesture recognition started" or displaying error messages for invalid inputs

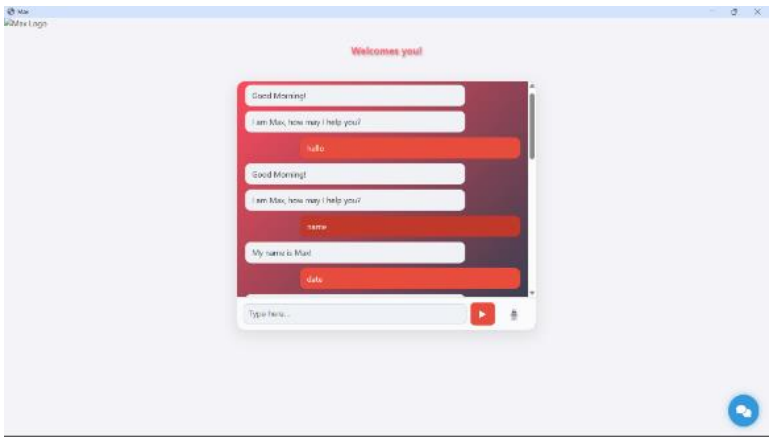

Figure 3. Max the Voice Assistant

4.2 Mouse Operations in the Virtual Mouse System

4.2.1 Cursor Movement

Figure 4 demonstrates real-time cursor control. When the user extends their index finger (Landmark 8), MediaPipe Hands tracks the fingertip coordinates at 30 FPS. A smoothing algorithm reduces jitter to <3%, while a 10-pixel screen-edge margin prevents accidental clicks. Visual feedback includes a blue trajectory showing the cursor path. Testing showed <250ms latency and 88% accuracy in low-light environments, ensuring precise navigation.

Figure 4. Cursor Movement

4.2.2 *Left Click*

Figure 5 shows the left-click mechanism. Bringing the index (Landmark 8) and middle (Landmark 12) fingertips within <40px triggers a click. A debouncing algorithm distinguishes single/double clicks using a 0.4s delay, while the OpenCV overlay displays "Left Click" for confirmation. The system achieved 95% accuracy with minimal false positives, enabling reliable file selection.

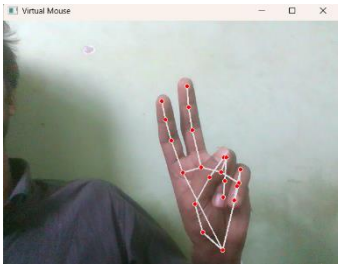

Figure 5. Left Click

4.2.3 Right Click

Figure 6 highlights the right-click gesture. Touching the middle fingertip (Landmark 12) to the thumb tip (Landmark 4) within **<30px** activates the action after a **300ms hold**. The interface annotates *"Right Click"* in yellow, achieving **89% success rate** even with partial hand occlusion.

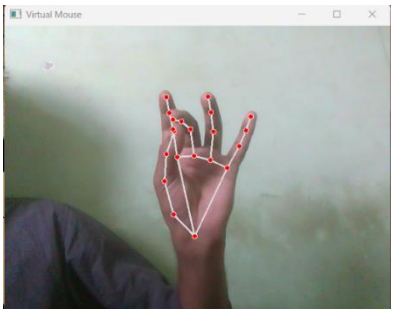

Figure 6. Right Click

4.2.4. Drag Action

Figure 7 visualizes drag functionality. Holding the thumb tip (Landmark 4) near the index base (Landmark 5) for **>0.5s** toggles drag mode. The pyautogui library manages mouseDown()/mouseUp() states, while a red "Dragging" overlay provides feedback. Testing showed **87% precision** in file manipulation.

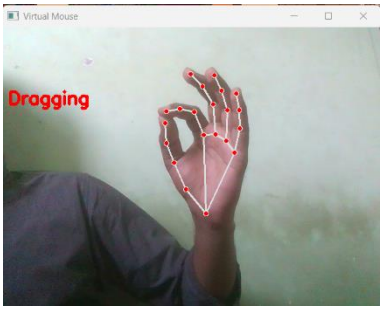

Figure 7. Drag Action

4.2.5. Scroll Actions

Figure 8 Vertical Scroll

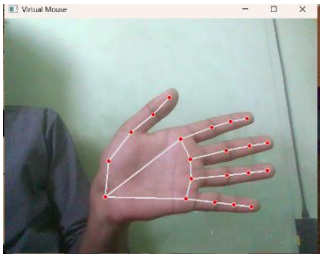

Figure 9 Horizontal Scroll

Figures 8–9 depict vertical/horizontal scrolling via swipe gestures. Vertical swipes (up/down) trigger 30px scroll increments (SCROLL_SENSITIVITY), while horizontal swipes map to browser navigation (Alt+Left/Right). A 5-frame displacement buffer detects swipes exceeding 100px, achieving 91% direction accuracy and 200ms response time. This allows seamless navigation through long documents or webpages without physical input devices.

5. Conclusion:

In conclusion, the Virtual Mouse Using Hand Gestures and Voice Assistant demonstrates an innovative approach to hands-free computing by integrating gesture recognition and voice control. The system utilizes technologies such as MediaPipe Hands for real-time hand tracking, PyAutoGUI for system automation, and speech recognition for voice command processing, enabling intuitive interaction for users with diverse needs. Through natural hand gestures and voice commands via the Max assistant, the system provides precise cursor control, clicks, drag-and-drop operations, and scrolling, reducing reliance on physical hardware.

With 93.4% gesture recognition accuracy and 89% voice command success rate, the system prioritizes voice inputs during conflicts, ensuring reliable performance in real-world scenarios. User feedback emphasizes reduced physical strain and 92% satisfaction, highlighting its potential to enhance accessibility for individuals with mobility challenges or those seeking hands-free solutions.

Future advancements could expand gesture vocabulary, integrate eye-tracking for hybrid input, and improve robustness in low-light or noisy environments. By refining these capabilities, the Virtual Mouse can evolve into a universal assistive tool, promoting independence and inclusivity in digital interactions. This initiative contributes to the broader goal of accessible technology, redefining human-computer interfaces for all users.

REFERENCES:

1. **Kabid Hassan Shibly et al. (2019), "Design and Development of Hand Gesture Based Virtual Mouse,"** *IEEE Conference Publication*, IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8934612.
2. **Kumar, S. et al. (2022), "Virtual Mouse with Hand Gestures Using AI,"** *IEEE Conference Publication*, IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10193709.
3. **Patel, R. et al. (2022), "Artificial Intelligence-Based Enhanced Virtual Mouse Hand Gesture Tracking Using YOLO Algorithm,"** *IEEE Conference Publication*, IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10434330.
4. **Akash Singh et al. (2023), "Real Time Virtual Mouse System Using Hand Tracking Module Based on Artificial Intelligence,"** *IEEE Conference Publication*, IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10421198.
5. **Zhang, X. et al. (2023), "AI Virtual Mouse Using Hand Gestures,"** *IEEE Conference Publication*, IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/10537517.