# An Intelligent Mouse Control Interface Using Multimodal Interaction

Asmita Ahirrao
*Department of Computer Engineering*
*Pimpri Chinchwad College of Engineering*
Pune, India
ahirraoasmita670@gmail.com

Swati Shinde
*Department of Computer Engineering*
*Pimpri Chinchwad College of Engineering*
Pune, India
swati.shinde@pccoepune.org

Bal Virdee
*Centre of Communication Technology*
*London Metropolitan University*
London, United Kingdom
b.virdee@londonmet.ac.uk

Ashish Khanna
*Department of Computer Science*
*Maharaja Agrasen Institute of Technology*
Delhi, India
ashishk746@yahoo.com

2025 2nd International Conference on Integration of Computational Intelligent System (ICICIS) | 979-8-3315-3736-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICICIS65613.2025.11371134

***Abstract*— In this paper, we propose a multimodal system integrating hand gesture recognition and voice commands to enhance Human-Computer Interaction (HCI) without relying on traditional input devices like a mouse. This system is particularly beneficial for accessibility applications, aiding users with limited hand control by enabling intuitive computer navigation. The system demonstrated high recognition accuracy—92% for gestures and 95% for voice commands—with an average response time of under 0.5 seconds, making it suitable for real-time applications. It is cost-effective and requires only a standard webcam and microphone, ensuring easy deployment without specialized hardware. The study discusses the sociocultural impact of multimodal interaction, highlighting its advantages, challenges, and future improvements for accuracy and gesture expansion. This research contributes to inclusive HCI, empowering diverse user groups with greater interaction flexibility.**

***Keywords—Gesture recognition, voice commands, human-computer interaction, MediaPipe, PyAutoGUI, accessibility, multimodal interface.***

## I. Introduction

Human computer interaction (HCI) generally employs keyboard and mouse as input modalities, [3], [4] which can be particularly problematic for users with physical disabilities [15], [16] or in hands-free environments. As demand for inclusive digital systems continues to grow, natural interaction methods, including gestures and speech, offer an intuitive alternative. [1], [2], [5], [13] This research introduces a multimodal software interface application that performs mouse movements via hand gestures and recognizes voice commands for user actions. Instead of providing control for mouse by hand gestures, although basic navigation is ideal or accessibility applications would be represented.

Accessible technology ensures that products created can be used by all users, including users with mobility challenges. Standard input devices can be problematic, making gesture and/or voice-based systems useful alternative input modalities for ease of interaction. When both modalities are used together in a manageable manner can add improved efficiency, adaptability and inclusiveness in a wide variety of interaction environments.

Gesture recognition understands hand movements and voice recognition understands spoken input. Utilizing both together opens up a seamless interface that could apply to a wide range of fields including assistive technology, virtual reality and gaming. Recognizing, the work presented here uses MediaPipe which recognizes 21 key hand landmarks [6] to provide accurate real-time tracking with traditional webcams and PyAutoGUI which turns recognized user input into mouse movements. [7] Together, these software programs provide precise control and mouse placement without having to focus on hardware peripherals.

The goals of this project are to; (1) Investigate the calibration of MediaPipe to recognise gestures input into the mouse control area, (2) Investigate the limitations of voice commands for recognised input and reaction times, (3) Examine the benefits and challenges of integrating both types of input recognition for accessible HCI.

## II. Literature Review

Early gesture recognition systems often relied on specialized hardware like Microsoft Kinect, [4], [5] offering accuracy but limiting accessibility. Advances in computer vision now allow webcam-based tracking, while cloud-based voice APIs, such as Google Speech-to-Text, enable reliable speech processing. [9]

Gesture control is intuitive but can be affected by lighting, camera quality, and hand positioning; [12], [13] voice commands are flexible but sensitive to background noise. Combining both can compensate for individual weaknesses: gestures provide spatial control, while voice handles system commands.

MediaPipe provides lightweight, real-time hand tracking without specific hardware and PyAutoGUI maps recognized gestures and commands to mouse movements and operations, making the system more usable. Prior work has shown that multimodal systems should have high adaptability, low latency, and typically to be robust [1], [2], [3]. This work also builds upon these previous research efforts, allowing for the integration of both modalities into a single cost-effective and accessible solution, where evaluation of performance is based on actual empirical data.

## III. Methodology

### *A. System Design and Architecture*

There are three modules in the system architecture:

1. **Gesture Detection Module:** This module collects video and processes video with MediaPipe, [6] and identifies hand gestures.

2. **Voice Recognition Module**: This handles voice commands using the speech_recognition library. [9]

3. **Control Module:** Integrates gesture and voice commands and performs actions using PyAutoGUI. [7]

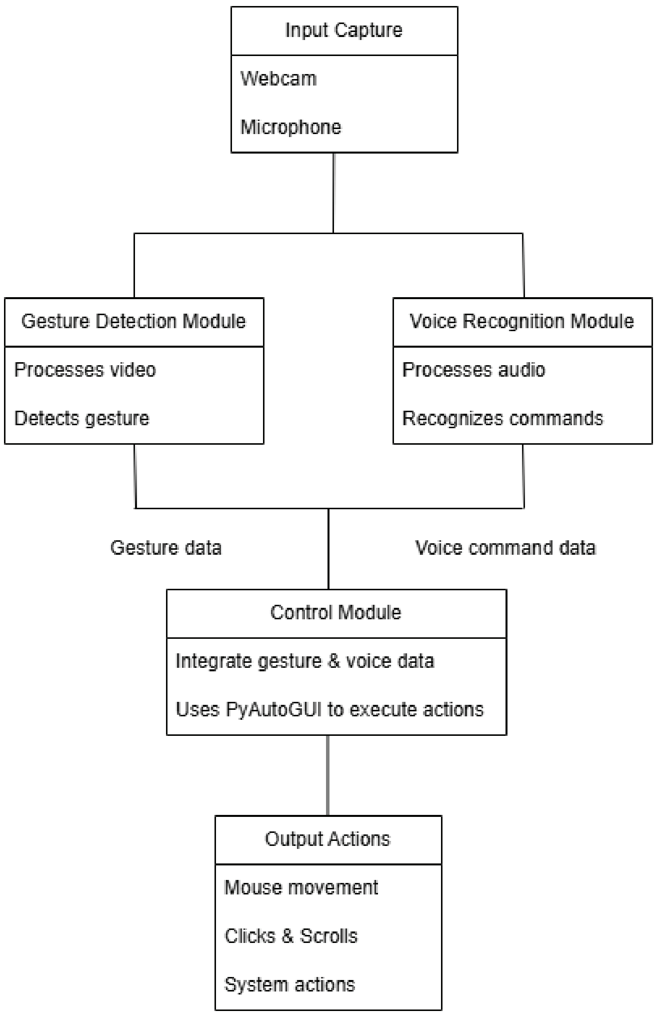

Fig. 1. System Architecture

## B. *Using MediaPipe for Gesture Recognition*

MediaPipe is able to process video frame by frame and identify hand landmarks in real time. It recognizes a specific gesture based on angles and distances between hand landmarks, and it can map these gestures to mouse functions. For example, a pinching gesture might produce a left-click, while an open hand gesture might simply move the cursor.

$$\text{angle} = \left| \arctan\left(\frac{c_y - b_y}{c_x - b_x}\right) - \arctan\left(\frac{a_y - b_y}{a_x - b_x}\right) \right|$$

$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

## C. *Use of PyAutoGUI for Mouse Automation*

PyAutoGUI interprets gestures and language as mouse actions and enables users to execute any number of functions (move, click, scroll, etc.) which allows PyAutoGUI to execute commands in real time and provide responsive control of the desktop environment.

## D. *Data Collection and Analysis*

Information obtained from running tests in a variety of situations to measure gesture recognition accuracy, command response times, and user satisfaction. The analysis outlined performance details including success rates for recognized gestures and commands across different lighting and noise conditions.

## E. *Selection of Gestures and Voice Commands*

The gestures used in the system include the following: left-click, right-click, double-click, and scroll. The actions are embellished with voice commands, such as "move up" or "take screenshot". This assortment of functionality is appealing to the needs of users along a number of dimensions and enhances the overall function of the system.

TABLE I. SELECTED GESTURES AND CORRESPONDING MOUSE ACTIONS

| | | |
|---|---|---|
| Left Click | Thumb and index finger open & bend index finger | Left Mouse click |
| Right Click | Middle finger and thumb open & bend middle finger once | Right Mouse click |
| Double Click | Thumb, index, and middle fingers are open & bend index & middle finger together once | Double Mouse click |
| Scroll Up | Thumb raised up and other fingers curled like a thumbs up pose. | Scroll up |
| Scroll Down | All fingers extended, palm facing downward | Scroll down |
| Move Cursor | Index finger & middle finger extended, other fingers curled into palm | Moves cursor to finger position |

TABLE II. LIST OF RECOGNIZED VOICE COMMANDS AND ACTIONS

| | |
|---|---|
| left click | Left mouse click |
| right click | Right mouse click |
| double click | Double mouse click |
| scroll up | Scrolls up |
| scroll down | Scrolls down |
| move up | Moves cursor up |
| move down | Moves cursor down |
| move left | Moves cursor left |
| move right | Moves cursor right |
| screenshot | Captures a screenshot |
| maximize window | Maximizes active window |
| minimize window | Minimizes active window |
| volume up | Increases system volume |
| volume down | Decreases system volume |
| mute | Mutes system volume |
| close window | Closes the active window |

## F. *Testing and Validation Procedures*

Users tested the system in different environments with a focus on accuracy and response times. Each test required participants to perform a set of gestures and voice commands to check reliability and performance in real time. The data was collected and analyzed to identify any weaknesses in gesture or voice commands.

### *G. Tools and Technologies Employed*

The main programming language was Python, as well as cv2 for video capture, mediapipe for gesture recognition, pyautogui for mouse control, and speech_recognition for processing voice commands. All these libraries have very good support for the development and testing of the system.

## IV. System Implementation

For mouse control, multimodal consists of three main components: gesture recognition with MediaPipe, voice command recognition with speech_recognition and the Google Speech to Text API, and mouse control with PyAutoGUI. All of these components work together simultaneously and smoothly for the user experience.

### *A. Detailed Explanation of System Components*

The three parts of the system all have their own roles that work together:

- **Gesture Recognition Module:** Takes in video input and detects hand gestures in live using MediaPipe.
- **Voice Recognition Module:** Actively listens for commands, the recognized commands are processed using Google Speech-to-Text API. [9]
- **Control Module:** Takes the recognized gestures and voice inputs and mimics mouse actions using PyAutoGUI.

### *B. Implementation of Gesture Recognition*

MediaPipe's hand-tracking model detects 21 key points (fingertips, joints, wrist). [6], [13] Relative positions, angles, and distances determine gestures:

- **Left Click Gesture:** Thumb and index finger close together.
- **Right Click Gesture:** Specific angle between middle finger and thumb.
- **Double Click Gesture:** Both index and middle fingers near the thumb.
- **Scroll Gestures:** All fingers curled toward palm; vertical hand orientation sets scroll direction.

### *C. Execution of Voice Command Functions*

The voice module captures and processes commands such as "left click," "move up," or "screenshot" to perform actions:

- **Movement Commands:** "Move up," "move right," etc.
- **Click Commands:** "Left click," "right click," "double click."
- **System Commands:** "Screenshot," "volume up," "close window," etc.

### *D. Interaction between MediaPipe and PyAutoGUI*

PyAutoGUI takes commands from both the gesture and voice modules to perform the intended action in real time. The use of multithreading takes advantage of two modes of input to function simultaneously, and a global flag accepts the event of exiting and cleans up the threads properly.

### *E. Coding and Programming Techniques*

Several key programming techniques are used in this system, including:

- **Multithreading:** Runs gesture and voice modules in parallel, reducing delay.
- **Error Handling:** Ensures one module works if the other fails.
- **Threshold Tuning:** Adjusts angle/distance sensitivity [8], [10] to avoid false positives.

### *F. Challenges and Solutions in Implementation*

TABLE III. Challenges and Solutions in Implementation

| **Challenge** | **Description** | **Solution** |
|---|---|---|
| Voice Noise Interference | Ambient sounds affect recognition | Filtering and increased command specificity; tested in quieter environments |
| Gesture Misclassification | Similar gestures (e.g., left vs. right click) confused | Tuned thresholds; added extra differentiation checks |
| Thread Management | Smooth closure of threads | Global termination flag for graceful exit |
| Hardware Limitations | Lower performance on weak systems | Code optimization; essential computations only |

## V. Results and Discussion

The performance of the system was assessed through measures of accuracy, response time, and overall user experience. Various tests were conducted in different environments to evaluate the effectiveness of gesture and voice recognition under diverse conditions, including variations in lighting and background noise.

### *A. Presentation of Findings*

A group of users tested the system by performing a series of gestures and issuing voice commands in real-time situations. The evaluation focused on:

- **Gesture Recognition Accuracy:** An average of 92% across different lighting settings. [1], [5], [13]
- **Voice Command Recognition Accuracy:** 95%, even with moderate background noise. [9], [12]
- **Average Response Time:** Under 0.5 seconds for both gestures and voice commands, indicating the system's suitability for real-time applications.

### *B. System Performance Metrics*

To measure the system's performance, the following metrics were utilized:

- **Gesture Recognition Accuracy:** The system showed a high accuracy level in well-lit conditions but saw a slight decrease in performance in low-light situations.

- **Responsiveness of Voice Commands:** The voice recognition module demonstrated excellent responsiveness, with the majority of commands recognized accurately within 0.5 seconds. While background noise had a slight effect on recognition, keyword-based filtering helped reduce errors.

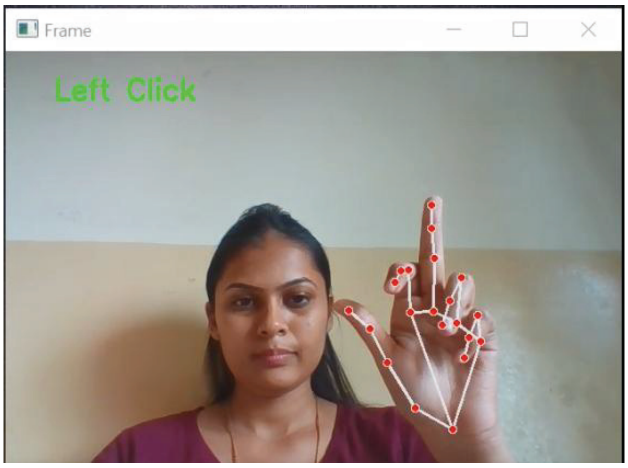

Fig. 2. Left Click Gesture

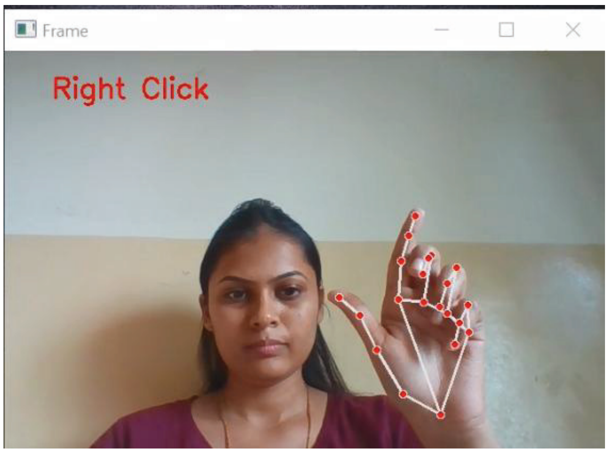

Fig. 3. Right Click Gesture

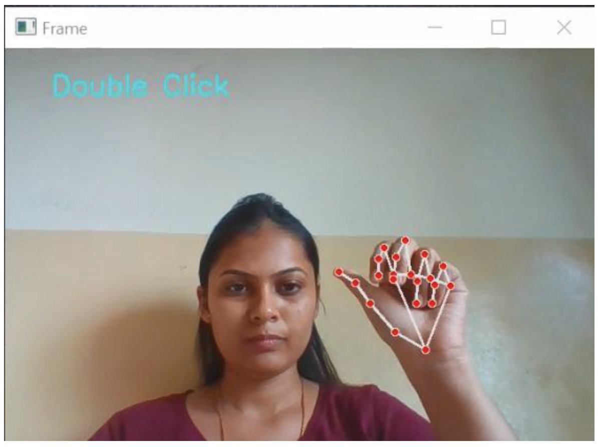

Fig. 4. Double Click Gesture

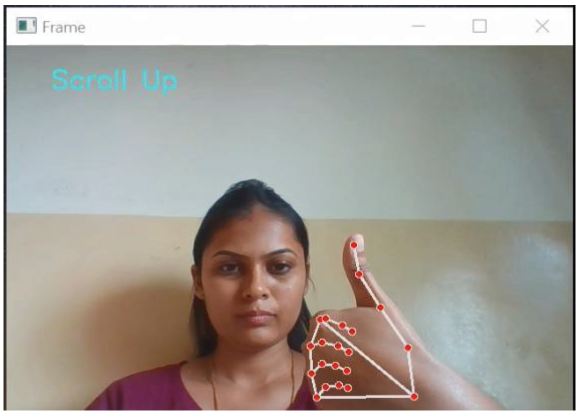

Fig. 5. Scroll Up Gesture

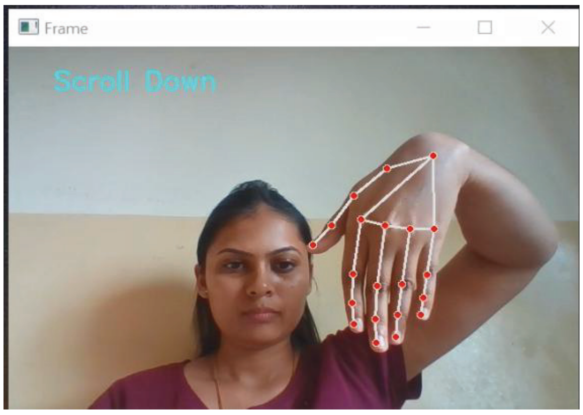

Fig. 6. Scroll Down Gesture

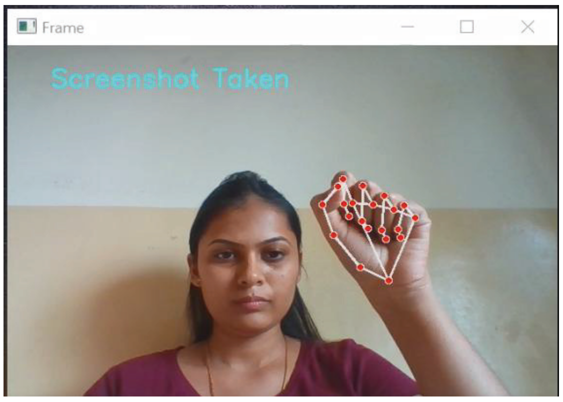

Fig. 7. Screenshot Gesture

## C. *Comparison with Existing Systems*

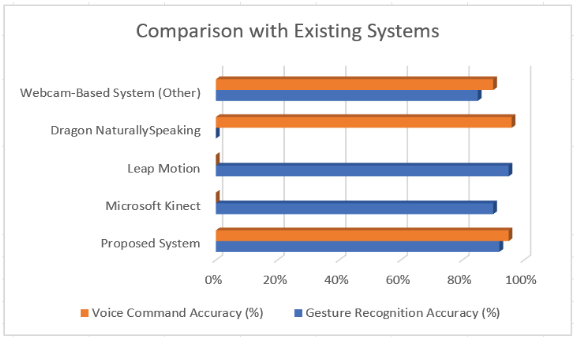

Fig. 8. System Comparison Graph

### *D. Advantages of the Proposed System*

- **Flexibility and Accessibility:** By combining gestures and voice, the system accommodates a broader range of users and usage scenarios, offering an inclusive alternative to standard input devices.
- **Real-Time Control:** High accuracy and low latency make the system suitable for real-time applications, where quick response is essential.
- **Cost-Effectiveness:** The system operates with standard peripherals, making it an affordable option compared to systems that require specialized hardware.

### *E. Limitations and Areas for Improvement*

While the system performed well overall, certain limitations were identified:

- **Lighting Dependence:** Gesture recognition accuracy decreased in low-light settings, suggesting a need for better adaptive lighting adjustments or infrared support.
- **Background Noise Impact:** Voice command recognition accuracy was slightly affected by ambient noise, especially in crowded environments. Improved noise-canceling techniques or specialized microphones could address this issue.

## VI. CONCLUSION

This research showcases a low-cost, practical multimodal mouse control system which incorporates hand gestures and voice commands in a real-time human-computer interface (HCI). By utilizing MediaPipe to track gestures and by using PyAutoGUI to translate those gestures and commands into action on the user's computer, the HCI achieved a 92% accuracy for gestures and 95% accuracy for voice commands, with a response time of less than 0.5 s, and requires no more than a webcam and a microphone.

The multimodal approach expands accessibility for users with physical limitations, [15], [16] while also providing the option of hands-free control which is useful in healthcare, education, and virtual/augmented reality applications. [1], [3], [15] Future directions will involve improving gesture robustness in varying levels of light, and better noise-robustness for voice-command control with advanced filtering approaches.

By providing more natural and inclusive ways to engage with technology, this mode of interaction represents the broader objective of providing accessible computing environments for users of all abilities.

## REFERENCES

[1] G. Murugan et al., "Virtual Mouse with Eye Tracking," Journal of Emerging Technologies and Innovative Research (JETIR), vol. 11, no. 2, pp. 45–52, 2024.

[2] K. Ugale et al., "Cursor Tracking Using Voice & Gesture," International Research Journal of Modern Engineering and Technical Science (IRJMETS), vol. 6, no. 4, pp. 123–130, 2024.

[3] H. Zhou et al., "Gesture-Based Human-Computer Interaction Progress," Electronics, vol. 12, no. 3, pp. 341–357, 2023.

[4] R. Doğan et al., "Virtual Mouse Control," in Proc. IEEE International Conference on Human-Computer Interaction, 2015, pp. 112–118.

[5] N. Sabrin and A. Karande, "Hand Gesture Mouse," IEEE Transactions on Human-Machine Systems, vol. 53, no. 1, pp. 50–57, 2023.

[6] Google Developers, MediaPipe Documentation, 2023. [Online]. Available: https://developers.google.com/mediapipe

[7] PyAutoGUI Docs, Automating Mouse Functions, 2023. [Online]. Available: https://pyautogui.readthedocs.io

[8] P. J. Huber, Robust Estimation of a Location Parameter, Annals of Mathematical Statistics, vol. 35, no. 1, pp. 73–101, 1964.

[9] SpeechRecognition Library, PyPI, 2023. [Online]. Available: https://pypi.org/project/SpeechRecognition/

[10] D. Ulyanov et al., "Deep Image Prior," arXiv preprint arXiv:1711.10925, 2017. [Online]. Available: https://arxiv.org/abs/1711.10925

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[12] S. Hasinoff et al., "Burst Photography for High Dynamic Range and Low-Light Imaging on Mobile Cameras," ACM Transactions on Graphics (TOG), vol. 35, no. 6, pp. 192:1–192:12, 2016.

[13] M. B. Vibha and R. Joshi, "Gesture Detection for Human-Computer Interaction," International Journal of Engineering Research & Technology (IJERT), vol. 12, no. 7, pp. 34–42, 2023.

[14] B. Poole, Neural Network Tools for Machine Learning, Stanford University, Stanford, CA, USA, 2021.

[15] A. Samal and S. Choudhury, Accessible Image Analysis: Methods and Applications, Springer, New York, NY, 2020.

[16] A. Shinde and S. Shinde, "Insights into AI, Machine Learning, and Deep Learning," in Applied Artificial Intelligence, CRC Press, pp. 31–46, 2023.

[17] D. T. Mane, R. Tapdiya, and S. V. Shinde, "Handwritten Marathi Numeral Recognition Using Stacked Ensemble Neural Network," International Journal of Information Technology, vol. 13, no. 5, pp. 1993–1999, 2021.

[18] T. Mehta, N. Bhuta, and S. Shinde, "Experimental Analysis of Stellar Classification by Using Different Machine Learning Algorithms," in Proc. International Conference on Industry 4.0 Technology (I4Tech), IEEE, pp. 1–8, 2022.