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.2025.141060

Gesture Controlled Virtual Mouse with Voice Commands

Prajyot Milind Dhiware[1], Prof., Pravin I. Patil[2], Manoj V. Nikum*[3]

Student Of MCA, Shri Jaykumar Rawal Institute of Technology Dondaicha, KBC NMU JALGAON, Maharashtra, India[1]

Assistant Professor MCA Department, Shri Jaykumar Rawal Institute of Technology, Dondaicha, KBC NMU JALGAON, Maharashtra, India[2]

Assistant Professor & HOD MCA Department, Shri Jaykumar Rawal Institute of Technology, Dondaicha, KBC NMU JALGAON, Maharashtra, India*[3]

Abstract: Contactless human–computer interaction has become increasingly important for accessibility, hygiene, and natural user experience. This project presents a practical Gesture Controlled Virtual Mouse with Voice Commands framework that allows users to control the computer cursor and perform system actions using only a standard webcam and microphone. The proposed system leverages Google's Media Pipe for reliable hand landmark detection and OpenCV for image preprocessing and frame handling, while Speech Recognition (with local microphone input) maps spoken commands to system-level actions. The design emphasizes low computing requirements, real-time responsiveness, and user accessibility.

Extensive experimental evaluation under multiple lighting and background scenarios demonstrate an average gesture recognition accuracy of ~96%, voice command recognition accuracy of ~93%, an average latency of ~45 ms, and stable operating frame rates (25– 30 fps) on commodity hardware. The system is lightweight, platform flexible, and suitable for applications in healthcare, education, and assistive technologies.

Keywords: Gesture Recognition, Speech Recognition, MediaPipe, OpenCV, PyAutoGUI, Human–Computer Interaction, Virtual Mouse.

I. INTRODUCTION

Human–Computer Interaction (HCI) is shifting from traditional tactile input devices towards more natural, multimodal interfaces. Conventional Peripherals such as mice and keyboards are precise but require physical contact and are not always suitable for sterile or hands-busy environments. Gesture and voice-based interfaces enable intuitive, Touchless interactions that improve accessibility for users with motor impairments and reduce hygiene concerns in medical and industrial contexts.

This project aims to develop a hybrid system that fuses vision-based gesture recognition and audio-based voice commands to control pointer movement and perform common system actions without physical input devices. The implementation focuses on using open-source tools and standard hardware (webcam and microphone) to maintain low cost and ease of deployment. The solution integrates Media Pipe's pre-trained hand landmark model for robust hand pose estimation and leverages Python libraries for audio processing and action automation.

The primary objectives of the project are:

- To implement a real-time virtual mouse controlled by hand gestures using Media Pipe and OpenCV.
- To integrate voice commands that trigger application-level tasks and system controls.
- To ensure reliable performance across diverse lighting and background conditions with practical latency suitable for interactive use.

The rest of the paper is organized as follows: Section II reviews prior work, Section III discusses analysis and observations, Section IV presents the proposed system architecture, Section V details the methodology, Section VI reports experimental results, and Section VII concludes with future directions

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.2025.141060

II. LITERATURE REVIEW

Gesture and speech interfaces have been extensively explored in HCI research. Earlier approaches focused on simple computer vision techniques of skin color segmentation, contour detection, and convexity defects to detect hand shapes and gestures (e.g., basic finger counting and static gestures). These methods often suffered in variable lighting and complex backgrounds.

With the rise of robust landmark models, frameworks such as Media Pipe provide real-time hand key point detection (21 landmarks) with high stability on CPUs. Recent student and conference projects implemented virtual mouse systems using OpenCV and Media Pipe for cursor control and basic clicks; however, they often report limitations in low-light environments or lack voice integration.

Speech-controlled interfaces have matured through cloud and local APIs. Systems using Google's recognition offer high accuracy but may depend on internet connectivity. Offline solutions such as Vosk or on-device models provide privacy advantages but may require further tuning.

Hybrid or multimodal interfaces that combine vision and audio modalities have shown improved robustness and flexibility. Some hybrid models use gesture for spatial control (cursor movement) and voice for high-level commands (open application, volume control), but there is limited literature demonstrating a lightweight, low-resource implementation suitable for commodity hardware.

Table 1 — Summary of Related Work

Reference Approach	Hardware Key	Outcome Limitation
OpenCV-based virtual mouse (Student projects)	Contour + fingertip detection Webcam Basic	Cursor control Low-light instability
CNN-based gesture systems (research)	Trained CNN GPU High accuracy	Requires GPU C dataset
Media Pipe implementations (recent)	Landmark- based detection Webcam Stable,	Gesture-only, limited voice support
Speech Automation Systems	Cloud Speech APIs Mic Good Accuracy	Internet dependency
Hybrid prototypes	Gesture + Voice Webcam + Mic	Improved usability Implementation complexity

III. ANALYSIS AND DISCUSSION

A. Choice of Tools

Media Pipe is chosen for hand detection due to its optimized inference pipeline, which runs efficiently on CPUs and returns consistent 21-point landmarks. It provides a good balance between accuracy and Computational load.

OpenCV handles image capture and light preprocessing (histogram equalization and Gaussian blur) to improve hand detection in varied environments.

Speech Recognition with Google's backend provides high recognition accuracy for English commands during prototyping; in noisy or offline scenarios, alternatives such as Vosk or local models can replace it.
PyAutoGUI performs OS-level cursor movement and simulated clicks, supporting cross-platform automation.

B. System Constraints

Lighting Conditions: Visual detection is sensitive to extreme low light or strong backlight. Adaptive preprocessing mitigates some of these issues but cannot fully compensate for severely degraded inputs.

Noise: The voice component is affected by ambient noise; noise thresholding and short command vocabularies reduce misrecognition.

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.2025.141060

Hardware: The system targets commodity laptops (Intel i5, 8GB RAM). Achieving acceptable FPS and latency depends on CPU performance.

C. User Scenarios

Assistive Use: Users with limited fine motor control can navigate a desktop using larger gestures and voice commands.

Healthcare: Clinicians can view and navigate medical records in sterile environments without touching devices.
Presentations: Presenters can control slides with gestures and voice.

IV. PROPOSED SYSTEM

A. System Overview

The system is modular with three primary layers:

1. **Input Layer:** Webcam and microphone capture video frames and audio streams in real time.
2. **Processing Layer:** Gesture Module: Media Pipe extracts hand landmarks per frame. A gesture classifier interprets landmarks into actions (move, left-click, right-click, scroll, drag).

Voice Module: Continuous or event-driven microphone listener captures short utterances and converts them to text commands.

3. **Control Layer:** A central executor maps recognized gestures and voice commands to OS-level actions through PyAutoGUI, ensuring safe concurrency and command arbitration.

A multithreaded controller runs the gesture and voice modules concurrently. An arbitration policy resolves conflicts e.g., if a critical voice command arrives while a gesture action is in progress, a short debounce and confirmation logic prevent accidental overrides.

B. Functional Modules

1. **Frame Acquisition s Preprocessing:** Frames are resized and color- corrected. Histogram equalization improves contrast; median blur reduces noise.
2. **Landmark Extraction:** Media Pipe detects 21 hand landmarks. Normalized coordinates are converted to pixel coordinates and mapped to screen coordinates.
3. **Gesture Classification:** Gestures are recognized using rule-based distance and angle thresholds between key landmarks (index tip, thumb tip, middle fingertip). For example:

Cursor movement → Index finger extended alone.
Left click → Index + thumb pinch (distance < threshold). Right click →
Index + middle pinch.
Scroll → Two-finger vertical motion.

4. **Smoothing s Filtering:** Cursor jitter is reduced by applying exponential or moving-average smoothing to mapped coordinates.
5. **Voice Command Mapping:** Recognized phrases are matched against a dictionary of supported commands (open browser, close window, volume up, volume down, scroll to top). Confidence of thresholds must be exceeded to execute commands.
6. **Action Executor:** PyAutoGUI sends mouse/keyboard events. Safety checks prevent repeated unintended actions—e.g., requiring hold duration for drag.

C. Safety s Accessibility Features

Confirmation Mode: For destructive commands (e.g., delete, close), optional voice confirmation is enabled.
User Calibration: A short calibration routine maps the webcam region to screen coordinates to account for different camera positions and user distances.

V. METHODOLOGY

A. Implementation Stack

Programming Language: Python 3.10
Libraries: Media Pipe, OpenCV, PyAutoGUI, Speech Recognition, PyAudio (for live audio interface), NumPy, Math, Threading.
Hardware: Laptop (Intel i5, 8GB RAM), 720p webcam, built-in microphone.

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.2025.141060

Platform: Prototyped on Windows 10; code is portable to Linux with minor adjustments.

B. Development Steps

1. Environment Setup: Install dependencies via pip (media pipe, OpenCV- python, pyautogui, speech recognition, py audio).

2. Gesture Module:

Initialize Media Pipe Hands with detection of C tracking confidence settings.

- Capture frames using OpenCV; apply preprocessing.
- Extract normalized landmarks and convert coordinates.
- Implement rule-based classifiers for gesture recognition.

3. Voice Module:

Initialize microphone streams.
Implement short wake-word or hotkey mechanism for voice capture to reduce continuous false positives.
Use speech_recognition.Recognizer() with ambient noise adjustment before recognition.

4. Integration s Threading:

Run gesture detection loop and voice listener in separate threads.
Use a thread-safe queue to deliver recognized commands to the executor.

5. Action Executor s Safety:

Map commands to PyAutoGUI functions.
Implement rate-limiting and confirmation for critical actions.

6. Calibration s Settings UI:

Small GUI or console prompts for mapping camera area and adjusting thresholds.

C. Dataset s Testing

Unlike pure ML-based approaches, the system primarily relies on landmark detection and hand- engineered rules; therefore, no large gesture dataset is required. However, test sessions were recorded from 10 participants performing pre-defined tasks under varied conditions (bright, dim, cluttered background). Voice command accuracy was tested against a fixed set of commands spoken in natural conditions

VI. RESULTS AND DISCUSSION

A. Evaluation Metrics

Gesture Recognition Accuracy: Percentage of intended gestures correctly recognized.
Voice Command Accuracy: Percentage of intended voice commands correctly transcribed and mapped.
Latency: Average time between user action and system response (ms). Frame Rateb: Processing frames per second (fps).
User Satisfaction Rating: Subjective rating on a 1–5 scale.

B. Experimental Results

Table 2 — Summary of Results

Condition	Gesture Accuracy (%)	Voice Accuracy (%)	Latency (ms)	FPS	User Rating (/5)
Bright Light	97	94	40	30	4.8
Dim Light	93	91	53	25	4.4
Complex Background	95	90	47	27	4.6
Multi-User (different hands)	96	93	46	28	4.7
Average	95.25	92	46.5	27.5	4.62

C. Observations

Accuracy: Gesture accuracy remains high due to robust landmark detection even with simple rule- based classifiers. Voice accuracy is acceptable for defined command vocabulary.
Latency: End-to-end latency averages under 50 ms for gesture actions; voice-driven actions vary depending on recognition backend latency.

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.2025.141060

Stability: Smoothing significantly reduces cursor jitter; aggressive smoothing reduces responsiveness, so parameters require calibration.
User Feedback: Participants reported that gestures felt natural, and voice commands were convenient for tasks that require context (opening applications).

D. Failure Modes

Fast, blurred hand motion sometimes yields missed Landmarks.
Strong backlighting causes partial landmark loss.
Overlapping gestures (rapid successive gestures) may be misinterpreted; a short debounce of interval helps.

E. Comparative Note

While ML-trained gesture classifiers may achieve marginally higher accuracy under some conditions, the proposed approach offers pragmatic
trade-off: near real-time performance on standard hardware with minimal model training and data requirements.

VII. CONCLUSION AND FUTURE SCOPE

This project demonstrates a practical implementation of a Gesture Controlled Virtual Mouse with Voice Commands, offering a low-cost, real-time multimodal interface suitable for everyday use. The integration of Media Pipe for landmark detection and a straightforward voice recognition pipeline yields robust performance on commodity hardware. The system is particularly suited to scenarios demanding touchless interaction and improved accessibility.

Key Contributions:
A working, low-latency hybrid interface for cursor and system control using only webcams and microphones.
Practical calibration, smoothing, and arbitration mechanisms to make the system usable in realistic settings.
User validation shows high satisfaction and acceptable accuracy.

Future Work:

1. **ML Enhancements:** Add optional ML-based gesture classifiers (CNN or LSTM) trained on collected gesture datasets to handle complex gestures and personalization.
2. **Offline Speech Recognition:** Integrate offline models (Vosk, Whisper local) for privacy and reduced internet dependency.
3. **Multilingual Support:** Expand the voice command vocabulary to support regional languages.
4. **Edge Deployment:** Optimize the pipeline for Raspberry Pi or Jetson Nano to enable embedded applications.
5. **Adaptive Learning:** Implement user-specific calibration and reinforcement learning to adapt thresholds over time.
6. **Expanded Command Set:** Add gestures for multi-touch-like interactions (pinch-zoom, rotate) and richer voice-command syntax for scripting tasks.

REFERENCES

[1]. Google Media Pipe Hand Tracking — documentation and model references.
[2]. Brad ski, G. et. al., “Learning OpenCV: Computer Vision with the OpenCV Library,” O’Reilly, 2nd Edition, 2008.
[3]. Balakrishnan, R. et al., “Design and Implementation of a Virtual Mouse Using Hand Gestures,” Conference Proceedings, 2021.
[4]. Jagnade, G. et al., “Hand Gesture-based Virtual Mouse using OpenCV,” IEEE
[5]. IDCIoT, 2023.
[6]. Sen, A. et al., “Deep Learning-Based Hand Gesture Recognition,” IEEE, 2022.
[7]. Sharma, S., “Speech Recognition Approaches for Human Computer Interaction,” IJRTE, 2021.
[8]. Patel, J.C.N., Mehta, K., “Phishing detection and other hybrid systems,” Pattern Recognition
[9]. Letters, 2023. (used for style reference only)
[10]. PyAutoGUI documentation, cross-platform GUI automation with Python.
[11]. Vosk and Whisper project pages — offline speech recognition options.
[12]. User experience methodology references and HCI best practices.

