

# Gesture talk: an Integrated Multimodal AI Assistant (Gesture, Voice, and Conversational Intelligence)

Pawan gandhi<sup>1</sup>, Krina Masharu<sup>2</sup>

(B.Teach in Computer Science Engineering, Atmiya University, Rajkot, India

Email: [gandhipawan054@gmail.com](mailto:gandhipawan054@gmail.com))

(Faculty of Engineering and Technology, Atmiya University, Rajkot, India

Email: [krina.masharu@atmiyauni.ac.in](mailto:krina.masharu@atmiyauni.ac.in))

## Abstract:

Emerging trends in Human-Computer Interaction (HCI) emphasize multimodal input systems that combine visual gestures, voice commands, and dialogue-based AI. This work presents a Python-based assistant integrating MediaPipe/OpenCV, voice automation through SpeechRecognition and system subprocesses, and a Generative AI chatbot powered by Google's Gemini API. Inspired by prior multimodal studies and systems combining speech and gestures, our system enables real-time control of volume, brightness, media, applications, files, and AI chat—all running concurrently using multithreading for responsiveness. Evaluation demonstrates high accuracy and low latency, showing promise for intuitive, accessible multimodal interfaces.

*Index Term- Computer Interaction, Gesture Recognition, Voice Assistant, Generative AI, Multiomodal Interfaces, MediaPipe, Gemini, Automation*

## 1. Introduction

- While today traditional interfaces rely heavily on keyboards and mice, it requires the user to be limited to just one way to use their device(s). this Multi-modal system is designed to create an interactive relation between the user and the device. Apart from that it also allows one to be away from their computers or laptops and still use it although other similar modals exist in the market, Gesture Talk is completely free to anyone across the world and open to any customization. Gesture talk has three modules; AI, ChatBot and Gesture & Voice Control. This Multi-modal is designed to create an easy to use atmosphere for people new to technology and the physically disabled. It is simple and easy to use for anyone. Universal actions have been coded for elements such as brightness, volume and cursor control. Settings can also be accessed to with Gesture Talk

## 2.Related Work

Gesture-based Interfaces: Survey by Saravana et al. highlights challenges and applications of gesture control. - Multimodal Systems: Liu & Kavakli

(2010) explored integration of speech and gesture for games. - Voice-gesture Fusion: Studies show combining speech and vision improves recognition. - AI Chat Interfaces: Generative models enhance interaction flexibility and productivity.

## 3.System Architecture

Our system comprises three concurrent modules:1. Gesture Module – MediaPipe, PyAutoGUI, WMI, pycaw for system control. 2. Voice Module – SpeechRecognition and subprocess for file/app operations. 3. AI Chat Module – Google Gemini API for conversational intelligence. Synchronization is achieved through multithreading.

## 4. Implementation Details

Gesture Detection: Real-time tracking with MediaPipe, rules for volume/brightness, clicks, media, windows.Voice Commands: Continuous recognition with trigger phrases for apps. Conversational Chat: Gemini API chat integrated

into same interface. All modules run as daemon threads for smooth execution.

## 5. Evaluation & Discussion

Environment: Windows 10, Python 3.10.9. -  
 Gesture Control: >90% accuracy under lighting. -  
 Voice Control: Reliable recognition, latency<1s. -  
 AI Chat: Responsive, context-aware, non-blocking.  
 Multimodal integration improved usability over single-modality systems.

## 6. Conclusion & Future Work

This project demonstrates a functional multimodal assistant combining gestures, voice, and chatbot AI. Future-work: - Custom ML gesture recognition. -Offline-speech-models. -Cross-platform expansion. - Memory-aware AI with task execution.

## References

1. Saravana M. K., et al., 'A Comprehensive Survey on Gesture-Controlled Interfaces,' Int. J. Sci. Res. Sci. Technol., 2025.
2. J. Liu and M. Kavakli, 'A survey of speech-hand gesture recognition,' IEEE ICME, 2010.
3. 'Multimodal Interaction, Interfaces, and Communication: A Survey,' MDPI, 2025.
4. E. Ghaleb et al., 'Leveraging Speech for Gesture Detection,' arXiv, 2024.

Human–Computer Interaction (HCI) has historically revolved around keyboards, mice, and more recently, touchscreens. While these methods are effective, they impose a cognitive and physical load that can reduce efficiency in certain contexts. Emerging trends emphasize natural interaction, drawing inspiration from human communication that combines gestures, speech, and dialogue. For example, gesturing to adjust brightness or speaking to open an application mirrors real-world interaction models. Multimodal systems also have accessibility implications. Users with motor impairments may find gestures easier than typing, while speech-based interaction can assist those with visual limitations. The convergence of multimodal inputs in a single system can thus support inclusive design, making computing environments more universally usable. Our work contributes to this paradigm by demonstrating an

integrated assistant capable of performing daily computing tasks using gestures, voice commands, and conversational AI—all synchronized in real time. Unlike siloed systems that handle one modality at a time, our design emphasizes concurrency. We employ Python multithreading to ensure low-latency responses across all input channels, preventing the “blocking” effect common in sequential designs. This responsiveness is particularly vital for time-sensitive operations such as volume control during media playback or hands-free navigation while multitasking.

Previous literature demonstrates a steady evolution of multimodal systems. Gesture-based interfaces, for instance, have been applied in domains ranging from gaming (e.g., Microsoft Kinect) to healthcare, where surgeons use touchless interfaces to interact with digital imaging during operations. Saravana et al. (2025) provided a comprehensive survey on gesture-controlled interfaces, identifying challenges such as lighting conditions, occlusion, and cultural variation in gestures. Speech-based systems have advanced through the proliferation of cloud-based APIs like Google Speech Recognition and offline models such as Vosk. Studies indicate that recognition rates exceed 95% under ideal conditions, though accents, background noise, and latency remain key challenges. Liu & Kavakli (2010) emphasized the synergy of combining speech and hand gestures, particularly in gaming environments. Conversational AI has progressed dramatically with the rise of generative models. Early chatbots relied on rule-based systems, but transformer-based architectures such as Gemini (Google), GPT (OpenAI), and LLaMA (Meta) now enable contextual, multi-turn dialogue. Recent surveys (MDPI, 2025) highlight their ability to adapt across professional, educational, and assistive contexts. Despite these advances, few systems integrate gestures, speech, and conversational intelligence into a cohesive architecture. Our work addresses this gap by developing a unified framework that combines these three modalities.

The proposed multimodal AI assistant comprises three independent yet synchronized modules: Gesture Control, Voice Assistant, and Conversational AI Chatbot. Each module processes a distinct input stream and executes corresponding actions in real time. 1. Gesture Control Module: Built using MediaPipe and OpenCV, this module captures real-time video frames, identifies hand landmarks, and interprets gestures. It controls

system features such as brightness and volume via WMI and pycaw, and enables application navigation using pyautogui. 2. Voice Assistant Module: Powered by the SpeechRecognition library, it listens continuously through a microphone. Recognized commands map to predefined system operations such as opening YouTube, launching applications, or manipulating files. It includes simple file editing capabilities via voice. 3. Conversational AI Module: Integrates Google's Gemini API to handle free-form dialogue. This chatbot offers context-aware responses, enabling users to query information or perform task-oriented conversations. The synchronization between these modules is achieved through multithreading. Each module runs in a daemon thread, ensuring responsiveness even when other modules are active. The architecture is designed with modularity, allowing future extensions such as gesture-speech fusion or integration with IoT devices.

Our prototype implementation was developed in Python 3.10.9 on Windows 10. The system leverages the following core libraries: - MediaPipe + OpenCV: For real-time hand tracking and gesture recognition. - pyautogui: For cursor, keyboard, and screen interaction. - WMI + pycaw: For brightness and audio volume manipulation. - SpeechRecognition + Google API: For real-time voice-to-text translation. - google.generativeai: For conversational dialogue with Gemini. - threading + queue: For concurrent execution and message passing. Gesture Module: The gesture detection rules are simple yet effective. For instance, detecting "all fingers up" on the right hand triggers volume adjustments. The relative wrist displacement determines whether the volume increases or decreases. Similarly, left-hand gestures control brightness levels. These intuitive mappings reduce learning curves for new users. Voice Module: Voice commands include "open YouTube," "close Chrome," "open calculator," and file-based commands such as "open new file," "replace word," and "close file." These demonstrate the assistant's utility as both a productivity tool and system manager. AI Chat Module: The chatbot leverages Google's Gemini-2.5-Pro model, providing human-like responses. The integration allows the user to seamlessly switch from task execution (e.g., opening applications) to conversational queries (e.g., "Explain quantum computing"). Code integration

emphasizes error handling to ensure robustness. For example, speech recognition failures trigger retries, and gesture control includes thresholding to minimize false positives caused by camera noise.

We conducted preliminary evaluation in a controlled environment with moderate lighting and minimal background noise. Key findings include: - Gesture Control: Achieved >90% accuracy in detecting open-hand gestures for volume and brightness control. Latency was <150 ms per frame, enabling real-time responsiveness. Limitations included reduced accuracy under low light and occasional misclassification when hands overlapped. - Voice Control: Recognition accuracy was reliable (>93%) for standard English accents. Latency remained below 1 second due to cloud-based processing. Limitations included challenges in noisy environments and difficulties with multilingual commands. - Conversational AI: The Gemini chatbot exhibited coherent, context-aware responses. In extended conversations, it maintained topic continuity for up to 10 exchanges, after which occasional drift occurred. The system's multithreading ensured that chatbot interactions did not block gesture or voice functions. Comparative Usability: In surveys with 5 test users, multimodal interaction was preferred over unimodal approaches. Users appreciated being able to "switch modalities" depending on convenience—for instance, using gestures while hands were free, or voice commands while away from the keyboard. Overall, the assistant shows promise as an intuitive, accessible, and responsive multimodal interface.

This paper presented a multimodal AI assistant that integrates gestures, voice commands, and conversational dialogue into a unified system. The prototype demonstrated high responsiveness and usability, outperforming unimodal alternatives. Its modular design allows for extensibility and customization across diverse application domains such as smart homes, education, and accessibility. Future work will explore the following directions: - Custom machine learning models to extend the gesture vocabulary beyond simple rules. - Offline voice recognition to reduce reliance on cloud services and improve privacy. - Integration with wearable devices and IoT for ubiquitous multimodal interaction. - Contextual memory for the chatbot to enable long-term personalized assistance. - Cross-platform deployment, extending beyond Windows to Linux, macOS, and mobile devices. By addressing these challenges, the system

can evolve into a robust, widely applicable multimodal assistant capable of transforming human-computer interaction.